

CHAPTER III

IMPLEMENTATION DETAILS FOR FORWARD FLIGHT

As discussed in the earlier chapters, the objective of this research is to develop an efficient rotor flow solver that can accurately predict the aerodynamic loads in hover and forward flight, while accounting for the blade motion and aeroelastic deformation. The use of hybrid method allows an accurate and economical modeling of viscous features near the blades, and an accurate “non-diffusive” modeling of the shed wake in the far field.

The CFD calculations are limited to a single blade called the “reference blade” in forward flight as shown in Figure 3.1, rather than the entire rotor system as typically done in other wake-capturing CFD codes. The other blades are “seen” by the reference blade as a collection of bound and tip vortices. In the hover case, even the full Navier-Stokes solvers need to resolve only one blade and can use periodic boundary condition at azimuthal boundaries. In forward flight, the flow-field is no longer symmetric due to the spatially varying grid velocity and the complex pitching and flapping motion of the blade at different azimuthal positions. In conventional Navier-Stokes methods, all the rotor blades must be solved at each time step as illustrated in Figure 3.2. The grids for the

adjacent blades need to be patched carefully to model complex blade motions. Some methods use overset grids that require complex data structures and grid connectivity techniques, which may be computationally expensive. The present hybrid method avoids all these difficulties, since the flow field over other blades is not explicitly gridded and solved.

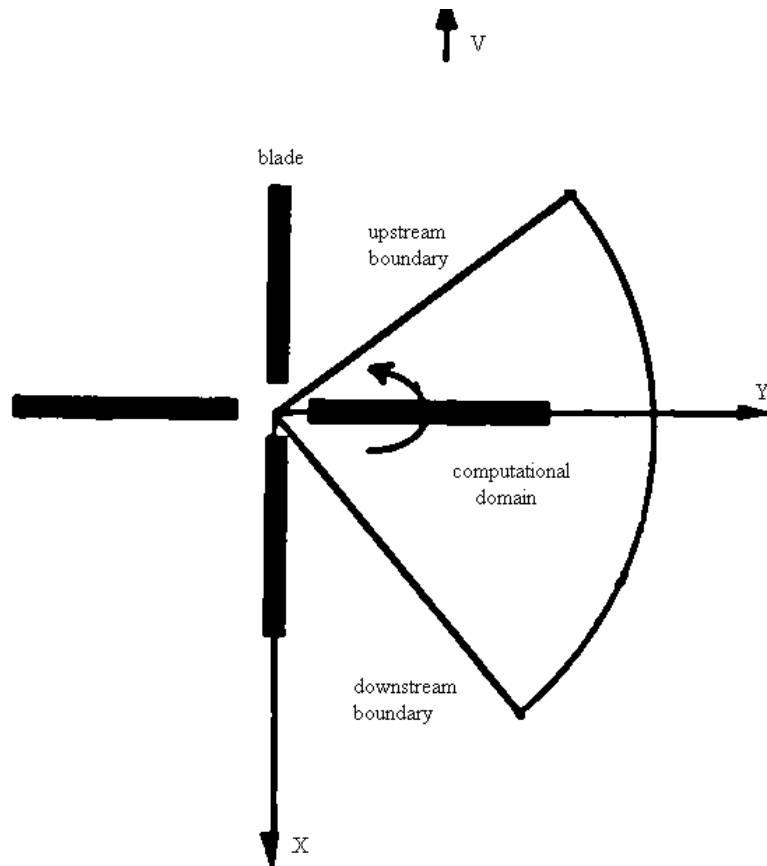


Figure 3.1: Computational Domain for the Hybrid Method

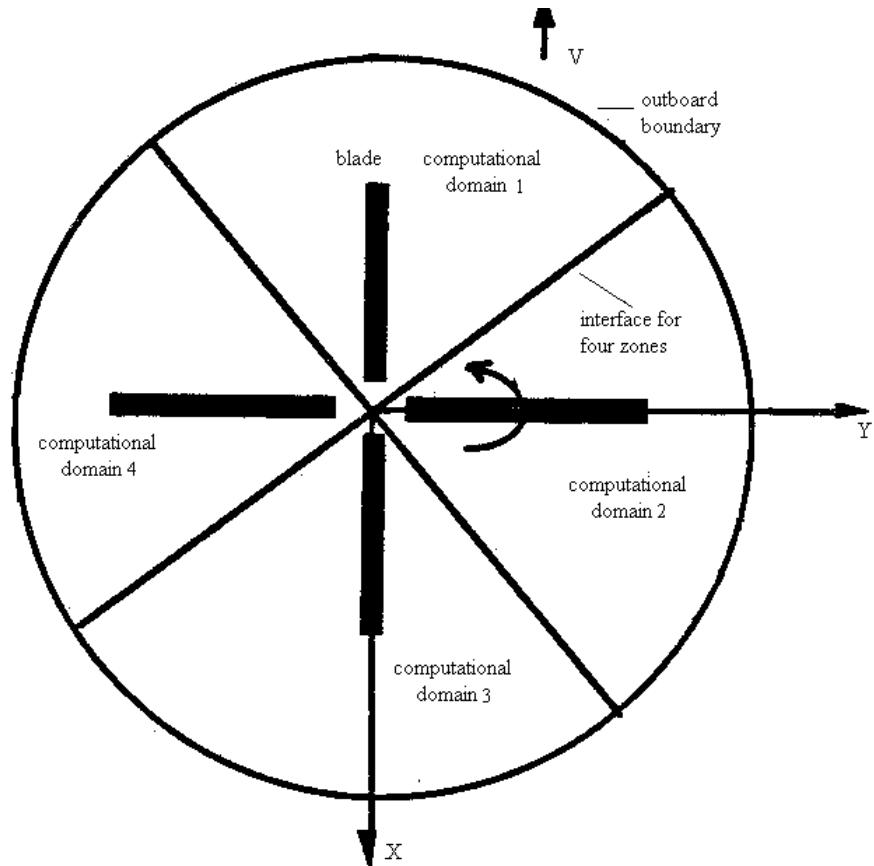


Figure 3.2: Computational Domain for a Conventional Navier-Stokes Solver

The accurate prediction of rotor blade aerodynamic loads in forward flight requires appropriate modeling of the unsteady, three-dimensional flow field at the rotor, the rotor wake geometry and strength, and the elastic and dynamic characteristics of the rotor in motion. The additional complexity that arises when coupling these diverse disciplines and phenomena together compounds the difficulty. The previous chapter described numerical models of the Navier-Stokes and full potential equations for the

unsteady, three-dimensional flow field. This chapter will concentrate on the blade dynamic motion, elastic deformations and the Lagrangean wake model.

3.1 Blade Dynamics

A module for handling blade motion has been developed to account for the rigid blade motions in flap and pitch, and the complex blade deformation due to aeroelastic effects. Eulerian angles [107] are a useful and convenient way of expressing the motion of rotating bodies in an inertial frame. The periodic blade motion in pitch and flap can be described by a Fourier series in terms of the blade azimuth. To keep the hybrid method stable for arbitrarily large blade motions, several ideas were tried to handle the complex blade motion.

3.1.1 Rigid Blade Motion

If only periodically steady state conditions (e.g. level forward flight, steady descent) are considered, the loads and motion repeat from cycle to cycle. The flap angle β and pitch angle θ can be written in forward flight as:

$$\begin{aligned}\beta &= \beta_0 + \beta_{1c} \cos \psi + \beta_{1s} \sin \psi + \beta_{2c} \cos 2\psi + \beta_{2s} \sin 2\psi + \dots \\ \theta &= \theta_0 + \theta_{1c} \cos \psi + \theta_{1s} \sin \psi + \theta_{2c} \cos 2\psi + \theta_{2s} \sin 2\psi + \dots\end{aligned}\tag{3.1}$$

In this approach, the blades are assumed to be rigid. The (x,y,z) positions in space at any instance in time may be related to (x,y,z) at an earlier time through simple transformation such as:

$$\begin{aligned} \mathbf{T} &= [\mathbf{A}][\mathbf{B}][\mathbf{C}] \\ \vec{x}_{new} &= \mathbf{T}\vec{x}_{old} \end{aligned} \quad (3.2)$$

The transformation matrix \mathbf{T} is given by the following equation in terms of the Eulerian angles [107]:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \quad (3.3)$$

$$\begin{aligned} t_{11} &= \cos \theta \cos \psi \\ t_{12} &= \cos \beta \sin \psi + \sin \beta \sin \theta \cos \psi \\ t_{13} &= \sin \psi \sin \beta - \cos \beta \sin \theta \cos \psi \\ t_{21} &= -\cos \theta \sin \psi \\ \text{where } t_{22} &= \cos \beta \cos \psi - \sin \beta \sin \theta \sin \psi \\ t_{23} &= \sin \beta \cos \psi + \cos \beta \sin \theta \sin \psi \\ t_{31} &= \sin \theta \\ t_{32} &= -\sin \beta \cos \theta \\ t_{33} &= \cos \beta \cos \theta \end{aligned} \quad (3.4)$$

3.1.2 Deforming Blade Motion

If the blades are not rigid, the grid motion should include additional rotations (in twist), and bending deformations. A number of strategies for including these effects have been considered.

3.1.2.1 Transpiration Velocity Approach

An efficient approach for including surface motions and deformations is to simulate the surface motion at the original, non-deformed blade through a transpiration boundary condition. All the validation case results presented in this work have been computed using the transpiration velocity approach. In this approach [108], a nonzero transpiration velocity component is specified to handle the local changes in the body deformation:

$$\left(\vec{v}_b - \vec{v}_{surface}\right) \cdot \vec{n} = \left(\Omega r + V_\infty \sin \psi\right) \left[\frac{d}{dx}(\Delta z)\right] \zeta_z \quad (3.5)$$

where Δz is the difference in the z coordinate in the blade surface between the rigid grid and the deformed grid at each time step.

This approach is efficient, does not require the calculation of metric terms at every time step, and avoids the complexity of the grid updating. However, when the amplitude of the blade deformation is large, this approach is not suitable and the boundary condition must be applied at the “exact” instantaneous blade surface.

3.1.2.2 Geometric Conservation Law

The Geometric Conservation Law (GCL) [93] states that a cell should not accumulate mass or momentum purely due to its motion. It may be written in integral form as:

$$\frac{\partial}{\partial t} \iiint dV + \oint \vec{V}_{grid} \cdot d\vec{s} = 0 \quad (3.6)$$

and in finite difference form as:

$$\frac{\partial}{\partial \tau} \left(\frac{1}{J} \right) + \frac{\partial}{\partial \xi} \left(\frac{\xi_t}{J} \right) + \frac{\partial}{\partial \eta} \left(\frac{\eta_t}{J} \right) + \frac{\partial}{\partial \zeta} \left(\frac{\zeta_t}{J} \right) = 0 \quad (3.7)$$

In the current studies, only the rigid blade motion is considered. Blade torsional and bending deformations are handled through transpiration boundary conditions. For this situation, the GCL is exactly satisfied. This is due to the fact that the Jacobian J (or cell volume), and the dot product $\vec{V}_{grid} \cdot d\vec{s}$ are independent of the blade orientation.

3.1.2.3 Exact Approach

The other approach for modeling surface motions and deformations is to deform the whole grid system, thus exactly following the blade deformations. The boundary conditions are also applied at the instantaneous boundary surface. At each time step, the grid point coordinates and the velocities of the new deformed grid need to be computed. Usually a line that runs from the root to the tip of the blade is chosen as a node line along which the deflections are zero. The torsional deformation and the flapping deformation (bending deformations which are generally very small) are specified with respect to the node line. At each of the spanwise locations, the shape of the node line and the amplitude of pitching and bending deformations can be prescribed from experimental data or from a Computational Structure Dynamics (CSD) code. These data are stored as a table and then interpolated, linearly in spanwise direction and using a Fourier series fit over the azimuth, to get the values at the computational spanwise stations at different azimuthal positions.

At any time level, the new instantaneous grid coordinates \vec{x}_d may be computed in terms of the rigid blade position \vec{x}_r as:

$$\vec{x}_d = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \vec{x}_r + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \quad (3.8)$$

where θ is the pitching deformation angle and (dx, dy, dz) are the bending deformations.

In order to decrease the grid irregularities, a variable mesh deformation was used to keep the exact deformation along the blade surface and no deformation at the far field

boundaries. This was implemented by using a factor ω that slowly varied from a value of 1 near the blade to 0 near the external boundaries.

Calculations were done without solving the Geometric Conservation Law equation. The implementation of this law can improve the accuracy of the results for moving grids since it maintains conservation of the governing equations. If the time steps used for computations with a moving grid are very small, the error due to geometric non-conservativeness can be considered to be negligible for most practical purposes [72].

Once the grid coordinates are known, the grid velocities can be computed analytically. An alternate way used in current calculation is computing the grid velocities numerically as:

$$\vec{V}_d = \frac{\vec{x}_d^{n+1} - \vec{x}_d^n}{\Delta t} \quad (3.9)$$

If the time step is large, then the finite difference formulation may be less accurate than the analytical estimate.

In the code validation studies, this “exact” approach was coded and found to be stiff, requiring very small time steps compared to the transpiration velocity approach, resulting in an increase of the overall solution time. The time step for the exact approach was at most one half the Δt that could be used with the transpiration velocity. This led to 160 hours of CPU time in Silicon Graphic O2 system. Therefore, this approach was not pursued further in the validation studies. The stiffness problem may be due to the fact that the present study computes time-accurate solution in a geometrically non-

conservative form. A Newton sub-iteration scheme may improve the stability of the solution. In this scheme, the calculation is stopped at an azimuthal angle and several sub-iterations are performed to numerically converge the pseudo-steady solution, then the calculation is advanced to the next azimuthal position.

3.2 Wake Model

In low speed operations, the rotor flow environment is strongly modified by the interaction between the rotor blade and the vortices shed from the neighbor blade. The ability to predict this wake is an important problem, especially for blade-vortex interactions (BVI) in forward flight. The requirement that the vortex core be resolved without dissipation can make direct CFD wake methods very expensive. In the hybrid method, the velocity decomposition in the potential flow equation coupled with Lagrangean wake convection is employed to preserve the shed vorticity and the associated rotational field. The ability to convect the vortices without numerical dissipation allows the use of fairly sparse grids. The grid resolution just needs to be adequate to resolve the boundary layer. There is no need for a globally fine grid capable of capturing the vorticity, as in conventional Navier-Stokes solvers.

Vortex-lattice and lifting-line based methods are examples of methods where a Lagrangean representation of the vortex is used. These methods have been studied for a long time, and are widely used because they are computationally efficient. These models make some assumptions regarding the location of wake and about the small-scale

features such as the core size. In hover, the wake models have been tested by Berkman [81] in conjunction with a hybrid method. In forward flight, the helical vortices shed from the blade tip are carried downward by the induced velocity component normal to the disk and rearward by the freestream velocity. In the present study, two different of wake models are implemented: a prescribed wake model and a free wake model.

3.2.1 Prescribed Wake Model

The tip vortex is modeled as a connected series of straight line segments. The connecting points are called “markers” as illustrated in Figure 3.3. The initiation point for the first vortex element marker is at the downstream interface between Navier-Stokes zone and the potential flow zone. The downward drift of the tip vortex from the blade tip to the interface due to the induced velocity is also considered. After each vortex element is created, it is convected at the local potential flow velocity of the flow field plus a velocity component induced by the wake itself. The prescribed wake model used in this work is an undistorted helical geometry shown in Figure 3.4. All the wake elements are convected with the same mean velocity. The tip vortex position is described by its age ϕ_w , which is the current blade azimuthal angle position minus the azimuthal angle of the shedding initiation point. The wake geometry is described in the inertial coordinate system as:

$$\begin{aligned}
 x &= r \cos(\psi - \phi) + R\mu_x\phi + R\lambda_x\phi \\
 y &= r \sin(\psi - \phi) + R\mu_y\phi + R\lambda_y\phi \\
 z &= z_0 + R\mu_z\phi - R\lambda_z\phi
 \end{aligned}
 \tag{3.10}$$

Here the ψ is the current azimuthal angle of the blade. μ_x , μ_y and μ_z are the nondimensional freestream velocities in the streamwise, lateral (sideslip) and normal (descent, climb) direction; λ_x , λ_y and λ_z are the induced flow field computed using Glauert theory. Also, z_0 is the spatial z - location where the vortex is released from the blade tip.

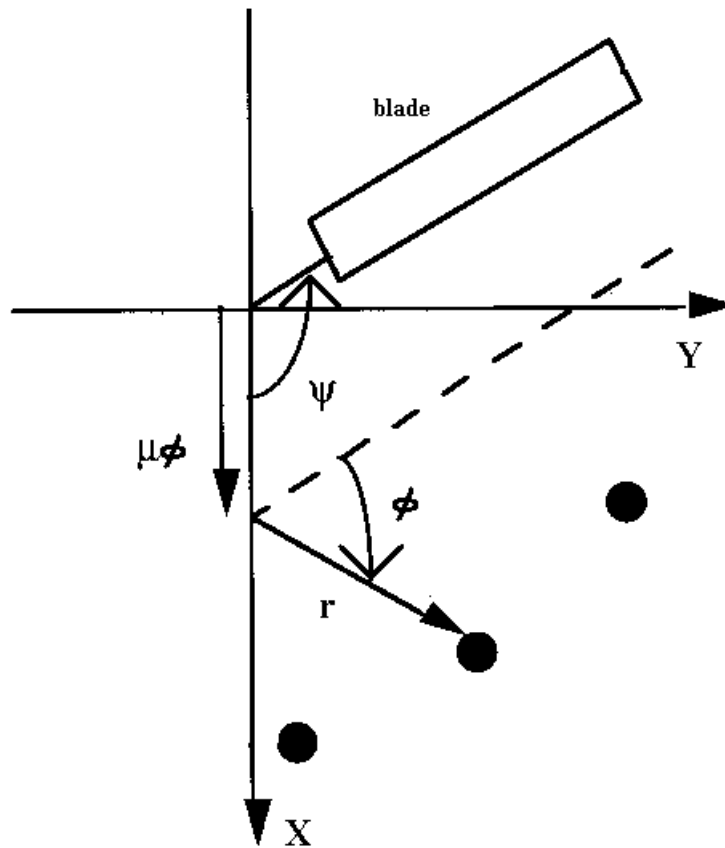


Figure 3.3: Wake Markers for an Advancing Rotor

The Glauert inflow model is used to estimate the inflow components λ_x , λ_y and λ_z . From momentum theory for forward flight, the solution for the induced velocity in dimensionless form is:

$$\lambda = \mu \tan \alpha + \lambda_i = \mu \tan \alpha + \frac{c_T}{2\sqrt{\mu^2 + \lambda^2}} \quad (3.11)$$

A Newton-Raphson method is employed to solve this nonlinear equation, starting from the value:

$$\lambda = \mu \tan \alpha + \frac{c_T}{2\sqrt{\mu^2 + c_T/2}} \quad (3.12)$$

Classical non-uniform inflow models in forward flight such as

$$v = v_0(1 + k_x r \cos \psi + k_y r \sin \psi) \quad (3.13)$$

can also be implemented, but have not been tested.

The prescribed wake model requires a negligible amount of computation, but can not model the wake distortion.

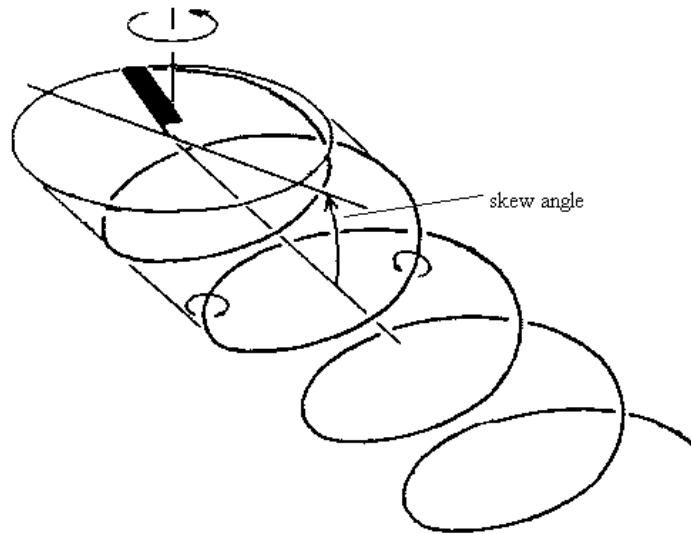


Figure 3.4: Structure of a Non-distorted (rigid) Wake

3.2.2 Free Wake Model

If the wake from the preceding blade remains close to the resolved blade, the distortion of the wake geometry due to self-induced velocities can have a large effect on the loading. In the free wake model, each wake element is convected at the local flow velocity, including the velocity induced by the wake on itself:

$$\vec{V} = \vec{V}_\phi + \vec{V}_w + \vec{V}_{Flight} \quad (3.14)$$

Free wake methods can capture the distortion from the basic helix as illustrated in Figure 3.5. One of the ways of computing velocity \vec{V} at the wake markers is to search for the computational cell in which each wake marker lies and calculate the local flow velocity for the marker by interpolating the velocity components from the surrounding grid nodes. The search algorithm is similar to that used in hover by Berkman [81]. A trilinear interpolation routine coded by Hariharan [49] is used to calculate the local velocity for the markers. This method has been implemented in hybrid solution, but is computationally very expensive. Sometimes the search for the computational cell fails in forward flight. In forward flight, especially at high advance ratios, the wake markers moving at the freestream velocity, exit the computational domain after one revolution of wake markers.

To avoid these problems, it was decided to directly use the Biot-Savart Law to evaluate the self-induced velocity at every marker in the wake, and neglect $\vec{V}\phi$. The convection velocities are numerically integrated to obtain the spatial positions of the markers at every time step.

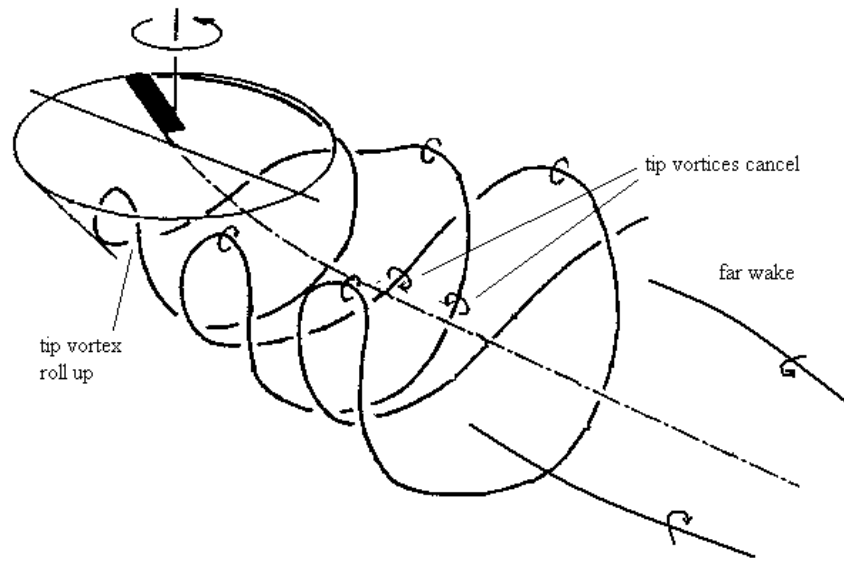


Figure 3.5: Structure of a Distorted (Free) Wake

In the present implementation, the free wake distortion begins with a prescribed wake geometry. The number of revolutions of the wake can be chosen. In forward flight, 3 to 5 wake revolutions are chosen, depending on the advance ratio. In the hybrid method, the wake strength and geometry are assumed to vary periodically with ψ , thus each blade's wake undergoes the same variation with azimuth. New wake filaments are added at the downstream interface between Navier-Stokes zone and the full potential zone as the rotor is advanced in the azimuthal direction. To keep the fixed number of wake elements small, the oldest age elements are dropped from the end of the wake. The induced velocities and wake geometry distortion are updated for all wake elements each time new wake filaments are shed. Also, in order to reduce the computational cost, the

frequency of updating the wake distortion can be controlled, permitting velocity updates after a multiple wake elements are shed, instead of after every wake element is shed. Free wake methods provide more generality with a minimum dependence on experimental data. However the large number of free wake elements leads to increased computational cost compared to prescribed wake methods.

3.3 Vortex Induced Velocity

3.3.1 Biot-Savart Law Approach

The velocity at an arbitrary grid point induced by a finite length, constant strength vortex line element is given by the Biot-Savart Law:

$$\Delta \vec{v} = -\frac{\Gamma}{4\pi} \int \frac{\vec{r} \times d\vec{l}}{|\vec{r}|^3} \quad (3.15)$$

Here $d\vec{l}$ is the elementary vector in the direction of the vortex filament and \vec{r} is the position vector of the grid point in question with respect to the vortex line element.

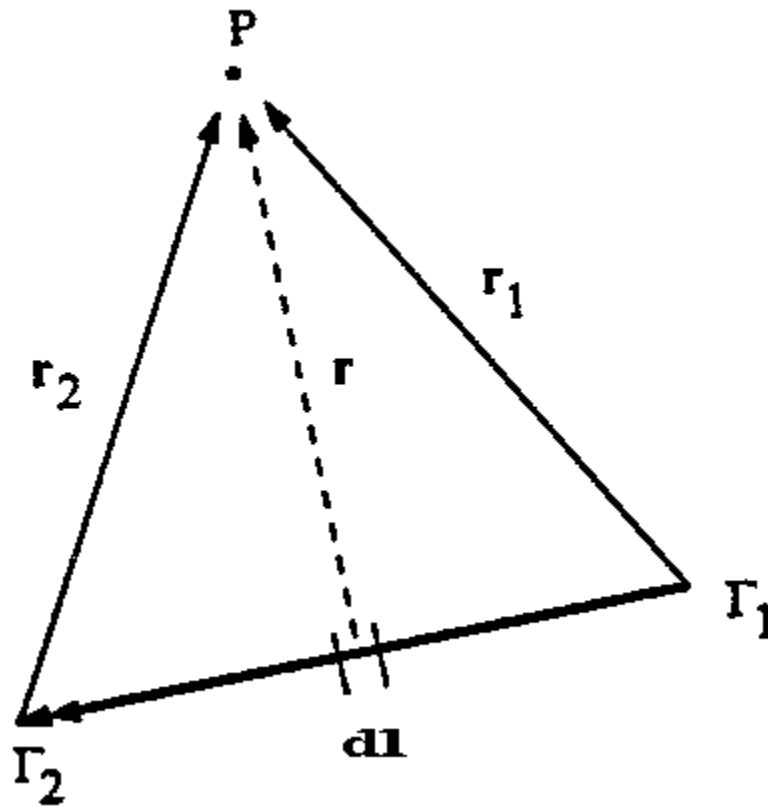


Figure 3.6: Induced Velocity due to a Finite Length Vortex Segment

Consider the straight vortex line segment shown in Figure 3.6. The vectors \vec{r}_1 and \vec{r}_2 are the position vectors from the ends of the line segment to the point P at which the induced velocity is to be computed. For a short vortex segment, the induced velocity is formulated as [6]:

$$\Delta \vec{v} = \frac{\Gamma}{4\pi} \vec{r}_1 \times \vec{r}_2 \left(\frac{1}{r_1} + \frac{1}{r_2} \right) \frac{1}{r_1 r_2 + \vec{r}_1 \cdot \vec{r}_2} \quad (3.16)$$

The vortex core model by Scully [22] is used to eliminate the singularity that occurs when the vortex line segment is close to the point P. Thus the velocity induced by the finite-length vortex line element is:

$$\Delta \vec{v} = \frac{\Gamma}{4\pi} \vec{r}_1 \times \vec{r}_2 \frac{(r_1 + r_2)(1 - \vec{r}_1 \cdot \vec{r}_2 / r_1 r_2)}{r_1^2 r_2^2 - (\vec{r}_1 \cdot \vec{r}_2)^2 + r_c^2 (r_1^2 + r_2^2 - 2\vec{r}_1 \cdot \vec{r}_2)} \quad (3.17)$$

where r_c is the vortex core radius.

In the full potential solver, the induced velocity must be computed at every grid node in the outer zone because the velocity decomposition approach is used in solving potential equations (discussed in Chapter II). Such a calculation is computationally intensive, but it is still cheaper than solving the Navier-Stokes equations at these nodes.

3.3.2 Clebsch Variables Approach

Another scheme implemented in the hybrid method is based on the vorticity embedding technique by Steinhoff et al [18]. In the full potential solver, the rotational components \vec{v}_w is computed using Clebsch variables:

$$\vec{v}_w = \Gamma_{i,j,k} \vec{\nabla} \lambda_{i,j,k} \quad (3.18)$$

which express \bar{v}_w as the product of a magnitude term Γ and a normal spatial distribution term $\nabla\lambda$. The shape parameter $\lambda_{i,j,k}$ is chosen to be a half-sine wave:

$$\begin{aligned}\lambda_{i,j,k} &= \frac{1}{2} \sin \frac{\pi\bar{\lambda}}{2D} \\ \bar{\lambda}_{i,j,k} &= \frac{\sum S_n W_{i,j,k}}{\sum W_{i,j,k}} \\ \Gamma_{i,j,k} &= \frac{\sum \Gamma W_{i,j,k}}{\sum W_{i,j,k}}\end{aligned}\tag{3.19}$$

where $W_{i,j,k}$ is a weighting function given by:

$$W_{i,j,k} = 1 - \frac{\Delta s^2}{D^2}\tag{3.20}$$

Here, D is the vorticity spreading distance, Δs is the distance of wake node from the grid point, and S_n is the signed normal distance of a grid node from a wake panel. Γ and λ are treated as a weighted average from all wake markers within a selected region of influence.

This model was implemented in the hybrid method but generated a very weak induced velocity field compared to that computed by Biot-Savart Law. This approach was therefore not pursued further.

3.4 Tip Vortex Strength

In forward flight, the aerodynamic loads vary with the azimuth. The strength of the tip vortex therefore varies with the vortex age, whereas it is a constant value in hover. The strength of the vortex element is equal to the peak bound circulation of the rotor blade at the instance in time when the vortex element was created. This means that the circulation strengths of the tip vortex are simply the time history of the blade peak bound circulation. An intricate procedure has been developed to prescribe an initial wake geometry and proper tip vortex strength, and to store and update the geometry and vortex strength at different azimuthal position. At the start of the simulation, the prescribed wake geometry discussed above is assumed, and the tip vortex circulation initialized using the model discussed by Mello et al [109]. The blade bound vorticity distribution is described as the following function of the non-dimensional radial location and azimuthal angle:

$$\Gamma_b(\bar{r}, \psi) = \Gamma_0 \bar{r} \sqrt{1 - \bar{r}^2} \frac{1}{1 + 1.5k_T \mu \sin \psi} \quad (3.21)$$

where k_T has the value of 0.982.

Applying the Kutta-Joukowski theorem, the relation between the constant Γ_0 and the thrust coefficient C_T is defined as:

$$\bar{\Gamma}_0 = \frac{\Gamma_0}{R(\Omega R)} = C_T \frac{\pi}{N_b} \frac{\sqrt{1-a^2}}{\frac{\pi}{16} + \frac{4}{9k_T}(\sqrt{1-a^2} - 1)} \quad (3.22)$$

where $a=1.5k_T\mu$. The tip vortex circulation at any given azimuthal angle is set to be the maximum bound circulation at that azimuth.

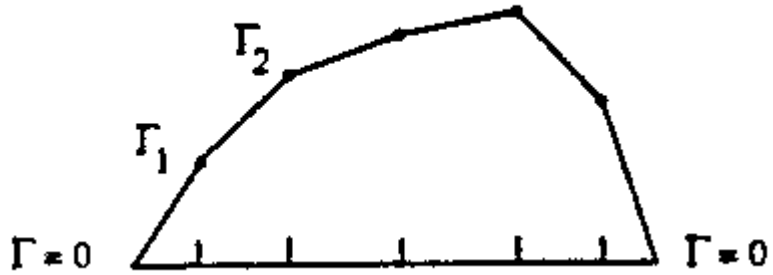
As the blade rotates to a different azimuthal position, the governing equations are integrated to yield the flow field at the next time step. The radial distribution of the sectional lift coefficient can be calculated from the surface pressure distribution. Then the bound vortex circulation is computed by Kutta-Joukowski theorem:

$$\rho U_T \Gamma = \frac{1}{2} \rho U_T^2 c C_l \quad (3.23)$$

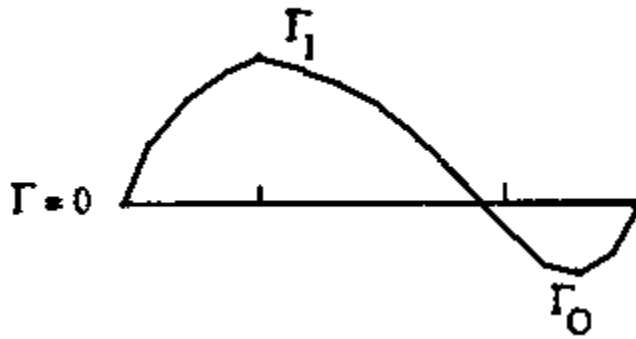
Conservation of vorticity requires that the radial changes to bound circulation cause trailing vortices to be shed into the wake. Usually the lift and the bound circulation are higher at the tip of the rotor blade, and the vortex sheet outboard of peak circulation quickly rolls up into a concentrated tip vortex, which has a strength equal to the peak bound circulation.

In low speed forward flight, the bound circulation is usually positive over the entire span of the blade. However, in some cases during high speed forward flight, the bound circulation can be negative at the blade tip for a range of azimuths on the advancing side, as shown in Figure 3.7. The negative loading is a consequence of moment balance on a flapping rotor which has a large negative twist. Typically, in such

cases there is a small negative bound circulation peak near the tip, and a large positive bound circulation peak inboard. So just searching for the maximum value of bound circulation along spanwise may lead to a result that has a wrong sign and too large a magnitude. In this instance, the value of bound circulation near the tip is chosen as the tip vortex strength. More complicated radial distributions of bound circulation can also be encountered. In the present implementation, the algorithm first searches the maximum value of Γ , then compares the radial location, r_c , of the maximum bound circulation (Γ_{\max}) with a user input value of radial location r_{input} (usually at 75%R). If r_c is greater than r_{input} , then the positive bound circulation distribution is assumed and Γ_{\max} is the tip vortex strength. Otherwise, the negative maximum value of Γ beyond r_{input} is located and chosen as the tip vortex strength. If all the above procedures fail, the value of bound circulation at a prescribed location near the blade tip (user can choose the radial location such as 97%R) is used as the tip vortex strength. This implementation was used in the validation case study and appeared to work well. In the case with negative bound circulation distribution, it may be more appropriate to use a weighted value of the positive and negative peaks of bound circulation as the tip vortex strength, or use multiple trailing vortices.



(a) Positive Circulation Distribution



(b) Negative Circulation Distribution near the Tip

Figure 3.7: Typical Circulation Distribution over a Rotor Blade

3.5 Rotor Trim

Comparisons with flight tests and experiments are only possible if the rotor is operating at the same thrust level in the CFD simulations and experiments. This requires

a trimming procedure. At a particular flight speed, the desired state is achieved through an adjustment of the rotor collective and cyclic pitch:

$$\begin{aligned}
 c_T &= c_T(\theta_0, \theta_{1c}, \theta_{1s}) \\
 c_{PM} &= c_{PM}(\theta_0, \theta_{1c}, \theta_{1s}) \\
 c_{RM} &= c_{RM}(\theta_0, \theta_{1c}, \theta_{1s})
 \end{aligned}
 \tag{3.24}$$

where c_T , c_{PM} , c_{RM} represent the specified coefficients of thrust, pitching moment and rolling moment. The blade pitch angle is defined as follows:

$$\theta(\psi) = \theta_0 + \theta_{1c} \cos \psi + \theta_{1s} \sin \psi
 \tag{3.25}$$

Since the relationship between the rotor aerodynamic parameters c_T , c_{PM} and c_{RM} and the blade pitch angle is non-linear, an iterative technique is necessary for convergence of the trim procedure. A Newton-Raphson iterative method is employed here to trim the rotor automatically. Given an initial guess for the control setting $\theta_0^0, \theta_{1c}^0, \theta_{1s}^0$, the flow solver is first run to get the initial rotor performance parameters $c_T^0, c_{PM}^0, c_{RM}^0$. Given desired trim state $c_T^d, c_{PM}^d, c_{RM}^d$, a new estimate for the control setting can be obtained using the iterative method:

$$\begin{Bmatrix} c_T \\ c_{PM} \\ c_{RM} \end{Bmatrix}^{(0)} + \begin{bmatrix} \frac{\partial c_T}{\partial \theta_0} & \frac{\partial c_T}{\partial \theta_{1c}} & \frac{\partial c_T}{\partial \theta_{1s}} \\ \frac{\partial c_{PM}}{\partial \theta_0} & \frac{\partial c_{PM}}{\partial \theta_{1c}} & \frac{\partial c_{PM}}{\partial \theta_{1s}} \\ \frac{\partial c_{RM}}{\partial \theta_0} & \frac{\partial c_{RM}}{\partial \theta_{1c}} & \frac{\partial c_{RM}}{\partial \theta_{1s}} \end{bmatrix}^{(0)} \begin{Bmatrix} \Delta \theta_0 \\ \Delta \theta_{1c} \\ \Delta \theta_{1s} \end{Bmatrix}^{(0)} = \begin{Bmatrix} c_T \\ c_{PM} \\ c_{RM} \end{Bmatrix}^{(d)} \quad (3.26)$$

Generally, $c_T^d = c_T^{\text{supplied}}$, $c_{PM}^d = 0$, $c_{RM}^d = 0$.

The trim Jacobian matrix is determined through a small perturbation of the control setting. Given a small perturbation $\Delta \theta_0^0$ to θ_0^0 , the flow solver can be run to get the values $c_T^0 + \Delta c_T^0$, $c_{PM}^0 + \Delta c_{PM}^0$, $c_{RM}^0 + \Delta c_{RM}^0$. Then

$$\begin{bmatrix} \frac{\partial c_T}{\partial \theta_0} \end{bmatrix}^{(0)} \approx \frac{\Delta c_T^0}{\Delta \theta_0^0}, \quad \begin{bmatrix} \frac{\partial c_{PM}}{\partial \theta_0} \end{bmatrix}^{(0)} \approx \frac{\Delta c_{PM}^0}{\Delta \theta_0^0}, \quad \begin{bmatrix} \frac{\partial c_{RM}}{\partial \theta_0} \end{bmatrix}^{(0)} \approx \frac{\Delta c_{RM}^0}{\Delta \theta_0^0} \quad (3.27)$$

Similarly, the other derivatives can be obtained. Once the Jacobian matrix has been computed, change to the control setting may be computed and a new guess for the setting is obtained:

$$\begin{bmatrix} \theta_0 \\ \theta_{1c} \\ \theta_{1s} \end{bmatrix}^{(1)} = \begin{bmatrix} \theta_0 \\ \theta_{1c} \\ \theta_{1s} \end{bmatrix}^{(0)} + \begin{bmatrix} \Delta \theta_0 \\ \Delta \theta_{1c} \\ \Delta \theta_{1s} \end{bmatrix}^{(0)} \quad (3.28)$$

The above iterative procedure continues until the rotor performance parameters reach the desired value. Applying this procedure to the hybrid analyses is too expensive

to be practical. It has been implemented to form the outmost trimming loop (illustrated in Figure 3.8) in a full potential code, and appears to work well.

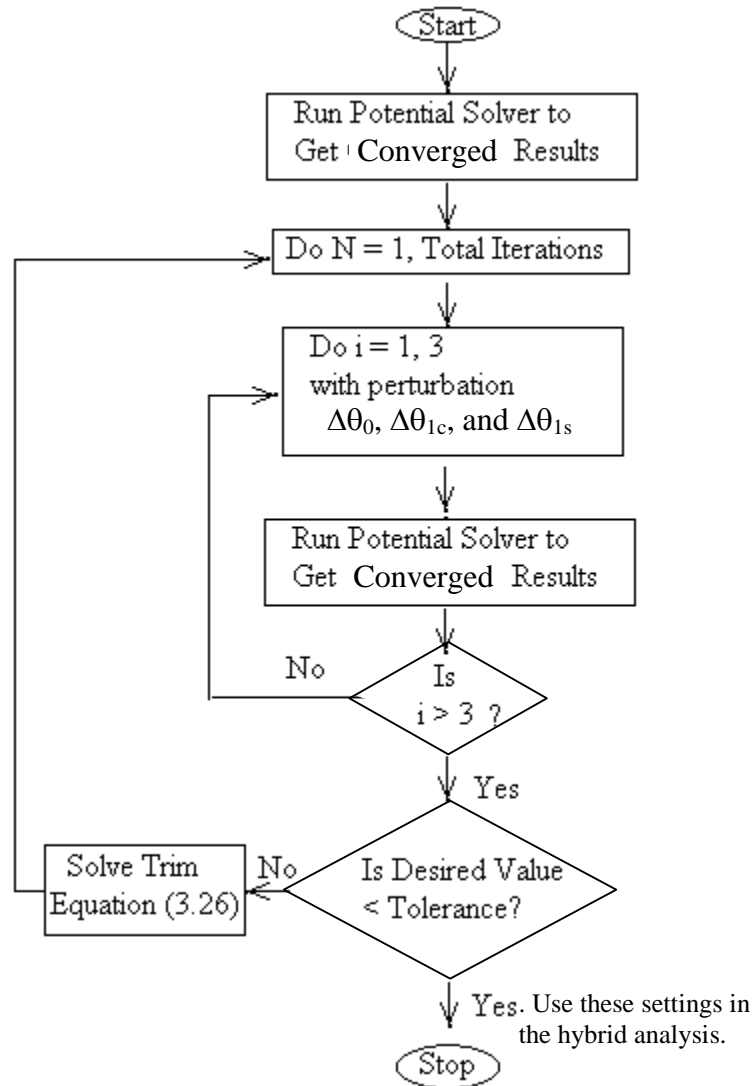


Figure 3.8: Flow Chart for the Outmost Trimming Loop