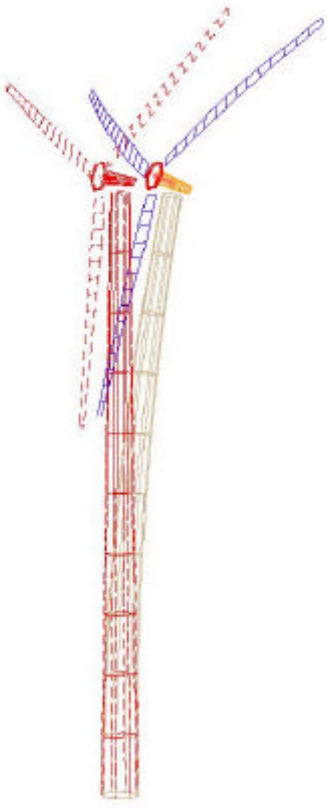
USER'S GUIDE

to the Computer Software Routines

AeroDyn Interface for ADAMS®



David J. Laino
A. Craig Hansen
Windward Engineering, LC
Salt Lake City, UT 84117
www.windwardengineering.com

Phone: 801-278-7852
Fax: 801-272-4132
email: dlaino@ windwardengineering.com
chansen@windwardengineering.com

Software date and version
ADAMS2AD 12.02, 06-Sep-2001

Prepared for the
National Renewable Energy Laboratory
under Subcontract No.
TCX-9-29209-01

Notice

This report was prepared as an account of work sponsored by the National Renewable Energy Laboratory, a Division of Midwest Research Institute, in support of its Contract No. DE-AC02-83-CH10093 with the United States Department of Energy. Neither the National Renewable Energy Laboratory, the United States Government, nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

The software described in this User's Guide is distributed for evaluation purposes only. Feedback concerning the validity of the software should be provided to the authors.

Table of Contents

List of Figures

List of Tables

USER'S GUIDE

to the Computer Software Routines

AeroDyn Interface for ADAMS®

About this Guide

This User's Guide is written to assist engineers with the preparation and use of the computer software routines for interfacing the wind turbine aerodynamics software AeroDyn with the commercial dynamics analysis program ADAMS®. ADAMS is available from Mechanical Dynamics, Incorporated (MDI) of Ann Arbor, Michigan. As of the writing of this guide, the current release version is ADAMS v11.0, but this guide was written while interfacing with ADAMS v10.1. See section 1.3 for interfacing with ADAMS 11.0.

This Guide was developed originally as part of the broader YawDyn/AeroDyn User's Guide. With the latest of the codes, YawDyn, AeroDyn and the ADAMS Interface all have separate User's Guides. In order to use the ADAMS Interface, AeroDyn and ADAMS are required. Refer to the AeroDyn User's Guide for information on the inputs and outputs that software requires and creates. It may also help to build a YawDyn model, and refer to that User's Guide as well, to understand how the new AeroDyn 12.3 interfaces with dynamics routines.

Other Guides in this Series

This guide is one in a set of three (as of the publication date on the cover), which include:

1. The AeroDyn 12.3 Users' Guide
2. The YawDyn 12.0 Users' Guide, and
3. The AeroDyn Interface for ADAMS 12.02 User's Guide (this guide).

Depending on which programs you intend to run, you may need to refer to other User's Guides in the series. If you plan to use ADAMS with AeroDyn, you will need guides 1 and 3 (this guide). If you plan to use YawDyn with AeroDyn, you will need guides 1 and 2.

1.0 Introduction

This guide describes the Fortran routines written to interface the ADAMS® program - a commercially available software package for Automatic Dynamic Analysis of Mechanical Systems written by Mechanical Dynamics, Inc. (MDI, Ann Arbor, MI) - with the AeroDyn routines for aerodynamic analysis of wind turbines. This interface was developed and is maintained with the support of the National Renewable Energy Laboratory (NREL) National Wind Technology Center (NWTC). These routines have been extensively rewritten since the last release (as part of YawDyn/AeroDyn v11.0), and subsequently extracted from that larger conglomerate of routines. The thrust behind this effort was the need to interface AeroDyn with a growing number of dynamics analysis codes. Taking a more universal approach to this interface makes maintaining and improving each code simpler in the long run.

This guide is written for the engineer/analyst who is familiar with wind turbine dynamics and aerodynamics as well as the ADAMS basics. Rudimentary knowledge of the application of user-written subroutines in ADAMS is also required. Information on ADAMS can be obtained from MDI's documentation or courses. Background information on the AeroDyn code and the aerodynamics analysis employed in AeroDyn can be found in a National Renewable Energy Laboratory (NREL) final report (Hansen, 1992).

This edition of the User's Guide is current as of the date shown on the cover page. It is applicable only to the specified version of the code. Since the software development is continuing, and significant changes are occasionally made to the codes, the reader should be certain the guide is appropriate to the version being used. Research is ongoing regarding the strengths and limitations of the codes. Users may wish to consult recent wind energy literature to improve their understanding of the codes and their accuracy. The change.log file maintains a summary of changes made to the ADAMS to AeroDyn Interface code.

1.1 Upgrading from version 11.0 of YawDyn/AeroDyn

The major way that this upgrade will affect users is that previous input and source code files from earlier versions are not compatible with this version. Be aware that upgrading is not a simple task, and you will have to upgrade AeroDyn as well to use this new interface. A conversion program (available with AeroDyn 12.3) is useful for upgrading an earlier yawdyn.ipt file to the new aerodyn.ipt file.. If you have a custom REQSUB.FOR, you will need to change it to use the new modules instead of the old include files. We recommend you keep your old version of YawDyn/AeroDyn while getting accustomed to the newer versions of the codes.

The main difference in the Interface from previous versions is in the the way AeroDyn and the Interface interact. The Interface to ADAMS makes a single call from GFOSUB to AeroDyn for an element aerodynamic force. AeroDyn in turn calls up to 4 routines in the Interface that provide the necessary information about the dynamics model to complete the task of calculating the aerodynamic forces. We feel this new method is a much 'neater' format than previous versions. A flowchart depicting this new method is provided in Appendix A.

1.2 Important changes since the last version

Aside from the major rewrite of the code itself, at least one change was made to improve code performance. That change is the addition of a smooth step function to ramp up the aerodynamic loads returned to ADAMS over the first 2 seconds of a simulation. By adding this feature, it is hoped startup problems experienced due to the sudden application of aerodynamic loads at time=0 will be eliminated. This became increasingly important with the new Generalized Dynamic Wake routine implemented in AeroDyn 12.3, which monitors the change in thrust on the rotor. Since the first several seconds of an ADAMS run usually must be discarded in any case due to startup transients, this step function should not impact the usable simulation results, and may even reduce the time required to damp out these startup transients.

1.3 A note about ADAMS version 11.0

The release version of this software is configured for use with ADAMS version 10.1. In trials with ADAMS 11.0, it was discovered that ADAMS no longer supported code included in the interface to handle unit conversions. The code to remedy this has been added to the interface routines, but is commented out in the release version. If you plan to use ADAMS 11.0, you must first activate these lines of code by removing the "!" at the beginning of each of these lines. You will find this code clearly marked in a single block near the top of subroutine UNITCHK in file GFOSub.F90. It reads:

```
 ! ======= ADAMS 11 code block; comment out for ADAMS 10.1============
!DOUBLE PRECISION   SCALES(4)  ! *** Use for ADAMS 11  SCALES array (1 to 4):…
!LOGICAL           EXIST      ! *** Use for ADAMS 11
!CHARACTER*2        UNITS(4)   ! *** Use for ADAMS 11

!CALL GTUNTS(EXIST,SCALES,UNITS)  ! *** Use for ADAMS 11
!FCONV = SCALES(3)                ! *** Use for ADAMS 11
 ! ======= End of ADAMS 11 code block ================================
```

Simply remove the first "!" from the 5 lines that start with this symbol, and compile the code as usual.

2.0 Requirements

To compile, link and run ADAMS with the aerodynamic subroutines the following items are required (these requirements pertain to the Windows NT/2000 operating system. Compilers and version numbers will vary for other platforms):

Purchased items:

1) ADAMS® version 10.1 and all hardware required to run ADAMS (Mechanical Dynamics, Inc., Ann Arbor, MI)

2) A FORTRAN 90 compiler (Compaq Visual Fortran, version 6.1).

Public domain items (These items are available from the NWTC web site):

3) The AeroDyn software files containing the AeroDyn source code (AeroSubs.F90, GenSubs.F90, AeroMods.F90).

4) The AeroDyn User's Guide.

5) The ADAMS to AeroDyn interface containing the source code (GFOSub.F90, ReqSub.F90, SenSub.F90).

6) This User's Guide.

User-supplied items:

7) Several data files that are created by the user. Up to six different types of input data files are required to use ADAMS with the AeroDyn aerodynamics routines: 1) ADAMS requires a data set that describes the structural model for ADAMS. 2) An ADAMS command (.acf) file can be used to control the simulation. The rest of the input files are required by AeroDyn (see the AeroDyn User's Guide for more information). These are: 3) A file named aerodyn.ipt that contains basic data such as blade aerodynamic characteristics and aerodynamic analysis control parameters. 4) File(s) that contain airfoil lift and drag data in addition to dynamic stall data. One of the last two items is required: 5) A hub-height wind file that contains time-varying parameters such as wind speed or blade pitch; or 6) Two full-field turbulence wind files that contain wind turbulence data. This guide will discuss what is required in the first of these files to interface ADAMS to AeroDyn. The AeroDyn files are addressed in the AeroDyn User's Guide.

The ADAMS data set and command file are not discussed in detail in this User's Guide. That information is available in the extensive MDI manuals. However, the aerodynamics routines place some requirements on the content of the data set. Three types of statements are required in the data set:

a) GFORCE statements that apply the aerodynamic forces.
b) A variety of MARKERS.
c) A SENSOR statement.

These requirements are detailed in the sections that follow.

3.0 Background

ADAMS simulates the dynamics of a structure, which is described using an "ADAMS data set". The turbine model can be quite complex and will analyze system dynamics such as coupled blade and tower vibrations. To model a wind turbine, ADAMS must obtain information about the aerodynamic forces on the blades. This is accomplished with the AeroDyn "user-written" subroutines that compile and link with ADAMS. These subroutines make it possible to model the aeroelastic interactions between the blade motions and the aerodynamic forces.

The AeroDyn software was originally developed as part of the YawDyn code, used for analysis of the blade and yaw loads and motions of a HAWT. It was later developed into a separate, complete subroutine package to model rotor aerodynamics. The package was developed at the University of Utah under an earlier contract with NREL and is in the public domain. YawDyn was the starting point in the development of the aerodynamics subroutines for ADAMS. The development and validation of the ADAMS wind energy package started in 1991 and is continuing. Thus the methods and software are constantly evolving as new information becomes available. It is important that users of the software provide feedback and suggestions to NREL and Windward Engineering to expedite the improvements to the code.

The evolutionary nature of this software and the need for constant comparison with other codes has led to a decision by the authors to maintain complete compatibility between several dynamics programs, such as ADAMS and YawDyn, and the AeroDyn subroutines. In earlier releases of the software, this compatibility was maintained through extensive duplication of code in the YawDyn, AeroDyn and ADAMS Interfacing routines. The latest release of the codes has sought to eliminate this duplication, making it not only simpler to maintain the codes, but more straight-forward to interface the dynamics codes – including ADAMS – to AeroDyn.

The AeroDyn User's Guide describes the subroutines required to interface AeroDyn with any dynamics analysis software. These routines can be found in the file GFOSub.F90. They provide AeroDyn with the status of the ADAMS model and the current blade element so AeroDyn can provide ADAMS with the correct aerodynamic forces it seeks.

The status of the dynamics models is easily obtained from ADAMS. The velocity or position vector of any location (indicated on the model with MARKERS) on the model (rotor hub, blade element, etc.) relative to any other location (yaw axis, ground, etc.) is available at any time from ADAMS (using INFFNC or INFARY subroutines). Obviously, the velocity can be the result of rotor rotation, structural vibration, or other rotor motions from causes such as yaw rate or tower bending. The relative wind velocity at a blade element is achieved through a vector combination of the wind velocity and element velocity, thus providing the relative velocity components required for aerodynamic analysis. The aerodynamic forces thereby become functions of the blade vibratory motions and the feedback loop between forces and motions is closed.

3.1 A Suggested Strategy for Modeling Wind Turbine Systems

ADAMS with AeroDyn is an extremely versatile and powerful tool. As with all powerful tools, there are many ways the analyst can approach the task of developing a complex model. We are convinced that the model development will proceed faster, with greater confidence, and more useful final results if the following general strategy is employed.

Step 1. Create a simple YawDyn model of the turbine. This may seem counterproductive if your ultimate goal is analysis of structural degrees of freedom that are not available in YawDyn. But it provides the opportunity to develop a basic understanding of the system, particularly the aerodynamic behavior. Little effort is wasted because the AeroDyn input files are also needed for the ADAMS model. YawDyn should be used until you are confident the mean loads and power output are correct. This usually requires "tweaking" the airfoil data files and checking all of the input variables. Doing this with YawDyn offers the advantages of simplicity and speed. *This is a very simple and useful step that is often ignored until the complete ADAMS model is running but giving mysterious results.*

Step 2. Evaluate the need for a more complex model. You may be surprised and discover that YawDyn is adequate for your needs. If this is the case you can save a great deal of time by using the simple model. If only a slightly more complex model is desired, consider the FAST_AD or SymDyn codes, which also interface to AeroDyn, are more versatile than YawDyn, but not so complex as ADAMS. If this is not the case, and you need more flexibility in modeling, you will need to develop an ADAMS model.

Step 3. If this is your first ADAMS model, we suggest creating a turbine in ADAMS that is identical to the YawDyn model you have just been using. That is, create a rotor with a flapping hinge and rigid blade, a rigid tower, and fixed yaw. ADAMS WT® makes it quite easy to create such a model. Run this model to be certain you get very similar results to what you see in YawDyn. We have done this a number of times and consistently find the two programs agree within a few percent when the models are identical. This will help you gain confidence that you have correctly modeled the turbine in ADAMS and that you understand the marker and data input requirements. You will also gain experience running and debugging a complete ADAMS data set without the complexity of a complete system model. The power train can be modeled with a MOTION statement to specify a constant rotor rpm. Simulations will run quite fast and the results will be relatively easy to interpret.

Step 4. Modify the model from Step 3 to use flexible blades. This will allow you to simulate all blade degrees of freedom without the additional difficulties involved in modeling power train and tower dynamics. You can "tune" the structural model to match natural frequencies known from finite element analysis or modal testing of the blade. Your model now includes many blade motions that YawDyn cannot analyze. You have arrived at this point by a series of incremental improvements and have a relatively small area in which to search if your model requires debugging. You will probably need to model the power train using a torque-speed equation to avoid large accelerations during startup. You may need to change the integrator parameters in ADAMS to achieve accurate results.

Step 5. Update your model to include all degrees of freedom you feel may be important. Again, changes can be incremental to make debugging relatively easy. It is very important to remember that a good model will reflect and sharpen the analyst's understanding of the system. This is much more important than using all of the degrees of freedom that are possible, simply because the software gives you that capability.

4.0 Adams Markers

ADAMS provides information about the model (such as element position and velocity) by using an intrinsic INFARY or INFFNC subroutine in conjunction with markers. Markers indicate a specific point on the model and its orientation. Some markers are required by the aerodynamics routines and they must have specific marker numbers, locations and orientations. We have tried to minimize the number of "hard-wired" marker numbers in the code, but some are essential. This section describes the markers that are required in the ADAMS data set. If the user wishes, it is possible to change the marker numbers in the subroutines to match them with the ADAMS data set. However, this requires a thorough understanding of the subroutines and is not recommended.

We have been using a marker and part numbering scheme that has helped us navigate through our own models and, more important, makes it easier for others to understand a model. This numbering system is not required by the subroutines, but we do find it useful and include it in this guide for your information. Table 4.1 lists the number ranges and their associated turbine components.

The required markers are shown in Figure 4.1, and can be found in the Interface release code in module ADAMSMarkers at the top of file GFOSub.F90. They are discussed in the sections that follow. If you are using ADAMS WT to create your model, these markers should be created for you. These markers are also used in the sample ReqSub.F90 file that is distributed with the software. Other markers may be added to the data set for the purpose of requesting additional outputs (the outputs in the release ReqSub.F90 are very limited). Any such markers you add will obviously not be detailed in this Guide.

4.1 Ground marker, ID = 1

The ground marker is attached to the ground part and must have ID number 1. It is oriented with z vertical upwards and x in the nominal downwind direction (indicating the zero yaw and wind direction angle). It must be located at ground level at the origin of the GROUND local part reference frame. (The wind need not actually be in the x direction, but all wind components are specified in this ground marker coordinate system.)

Table 4.1 - Marker and Part Number Ranges (Suggested)

Part and Marker Numbers	Components
1 - 999	Ground, other markers fixed to ground
1,000 - 1,999	Tower and guy cables
2,000 - 2,999	Nacelle, yaw bearing, mainframe
3,000 - 3,999	Power train: Generator, gearbox, shafts, etc.
4,000 - 4,999	Hub, teeter bearing, teeter stops, etc.
11,000 - 11,999	Blade #1
21,000 - 21,999	Blade #2
31,000 - 31,999	Blade #3

4.2 Aerodynamic force (AERO) markers, IDs selected by user

These markers identify the point of application of the GFORCE aerodynamic forces. One marker is required for each GFORCE (or blade element, in the terminology of the rotor aerodynamicist). Each marker must be aligned with the chord line of the local element airfoil (see below) with the Z-axis pointed toward the leading edge of the blade element, as shown in Figure 4.2. Thus all AERO markers will be aligned with the chord line of the blade element at the location of the marker. The pitch angle of each blade element is obtained by requesting the angle between the AERO marker and the pitch reference marker on the hub (see below). The PITCH + TWIST values familiar to users of YawDyn or PROP are both reflected in the location of the AERO markers. The TWIST values from the aerodyn.ipt file are ignored by AeroDyn when running ADAMS. This change was implemented in version 8.1 and is quite

different from earlier versions. The Y-axis is oriented nominally upwind such that the Y component of the aerodynamic force is usually negative. The X-axis will be directed either inward or outward along the span of the blade, depending upon the direction of rotor rotation. If the blade rotates clockwise (counterclockwise) when viewed looking downwind, the X-axis will be pointed outward (inward) along the blade span.

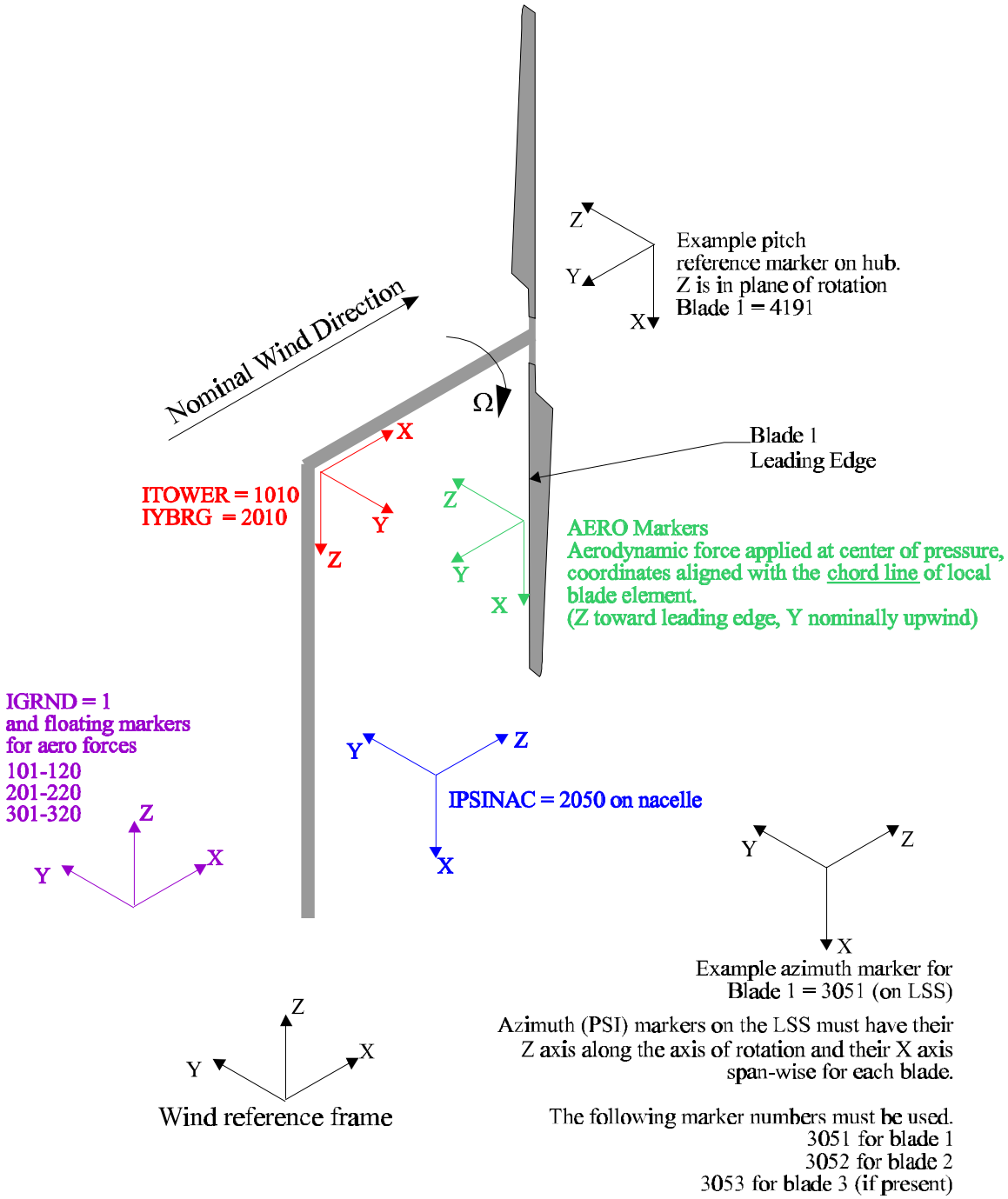


Figure 4.1 Required marker identification and orientation

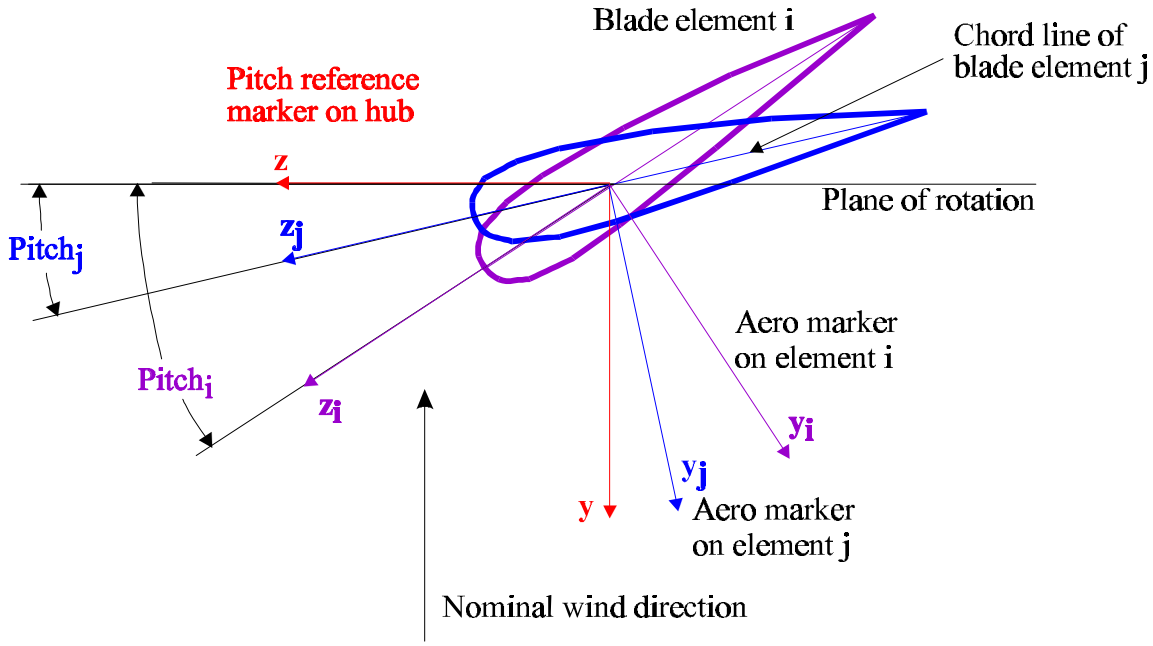


Figure 4.2 Definitions of aerodynamic pitch. The pitch angle determined by ADAMS for each blade element. The angles are shown in their positive sense. The **y** and **z** coordinates show the GFORCE marker orientations for two different blade elements i and j

Twist of the structural elements of the blade is defined using BEAM or other similar markers in the ADAMS data set. This structural twist must not be confused with the aerodynamic twist mentioned above.

An unlimited number of blade elements can be defined for an unlimited number of blades. Each element must have an aerodynamic force marker, a floating marker on the ground (see section 4.3 below), and a GFORCE statement that references the two marker ID's. For example, the following statements might appear in the data set to identify one GFORCE. The MARKER/101 definition given in section 4.3 must also be in the data set.

```
MARKER/11010, PART=10000, QP=0.74, 0.0, 0.0, REU = 0.0D, 90.0D, 0.0D

GFORCE/11010, I = 11010, JFLOAT = 101, RM = 11010,
FUNCTION=USER(1,1,0,11010)
```

This pair of statements applies one aerodynamic force vector to marker 11010 at a location that is 0.74 length units from the origin of the local part reference frame of blade part number 10000. The blade is rotating clockwise when viewed looking downwind. The x (spanwise) component of the applied force is always zero. The blade element part is pitched (and/or twisted) in the ADAMS data set, and the GFORCE marker is parallel to the part.

The arguments of the GFORCE USER function are all integer values that are defined as follows:

Argument #	Definition
1	The blade number (1, 2, 3…)
2	The blade element number (1 through NELM, with 1 being inboard and NELM being outboard)
3	Value of the VARVAL identifier that determines the location on the multiple airfoil table metric. When a single airfoil data table (not file) is used, enter 0. When multiple tables are used, enter the proper VARVAL index. This is a new, undocumented aspect of AeroDyn that is still under development. Users not wishing to use the control capabilities of ADAMS should always enter 0.
4	The ID number of the marker to which this aerodynamic force is applied (must be the same value as I and RM)

The blade element number provides the link between the ADAMS data set and the aerodyn.ipt file. The first blade element in the aerodyn.ipt file is element #1. There must be the same number of blade elements on each blade in the ADAMS model as in the aerodyn.ipt file, and each blade must have the same element layout (location, chord, and airfoil). The pitch angle can be different for each blade to simulate pitch imbalance. Though the numbering of the aerodynamic force markers is arbitrary, we strongly suggest you use a consistent numbering system. The system we have used seems to work well. The inboard element of blade one is numbered 11010, the second element is 11020, the tenth element is 11100, etc. The inboard elements of blades two and three are 21010 and 31010 respectively.

The AeroDyn software produces an output file that is useful for checking model inputs. The file is named GFOSUB.OPT. It echoes many input parameters and includes a table of blade element data that can be used to verify the location and orientation of each of the AERO markers at time=0. ADAMS calculates the blade element pitch in the same manner that is used during the simulation. This provides a thorough check of the blade marker orientations and locations.

4.3 Floating ground markers, Suggested IDs = 101-120, 201-220, 301-320

These markers are required for the GFORCE aerodynamic forces. One marker is required for each GFORCE. Each marker must be attached to ground and designated a floating marker. The orientation and location of the markers are arbitrary. Thus a typical ADAMS data set line would be:

```
MARKER/101, PART = 1, FLOATING
```

The markers associated with the blade elements on blade #1 have suggested numbers 101, 102, 103, up to the number of forces used on that blade. The markers for the aerodynamic forces on blade #2 are numbered 201,202, etc. These marker numbers are not required, but selection of these numbers will aid others users of the data sets.

4.4 Tower marker, ID = 1010

A marker must be attached to the topmost tower part and oriented with its Z-axis vertical downward as shown in Figure 4.1. This marker id is called ITOWER in the subroutines. This marker is used in conjunction with the yaw bearing marker (IYBRG=2010) to establish the yaw angle and yaw rate of the nacelle.

If the tower experiences torsional deflections (relative to ground), the yaw angle will err by an amount equal to the torsional motion. This is a result of defining the yaw angle with respect to the tower top rather than with respect to ground (as should be done to more precisely model the aerodynamics of the rotor). It is expected that this error will be negligible for most towers, but the user should be aware of this limitation.

4.5 Yaw Bearing marker, ID = 2010

This marker must coincide with the tower marker, but it is attached to the nacelle part. The yaw angle of the rotor is the angle of rotation about the Z-axis of marker 2010 relative to marker 1010. This angle is used in the aerodynamics calculations. It is suggested that markers 2010 and 1010 be the same markers that establish the yaw bearing. It is essential that structural deflections of any kind not cause relative motion between these markers that could introduce any artificial rotor yaw. The simplest way to ensure this is to connect the parts upon which these markers are placed with a revolute joint, and to place the markers at the location of the revolute joint. This allows only one rotational degree of freedom between the markers.

4.6 Nacelle marker, ID = 2050

This marker is attached to the nacelle and must be located on the low-speed shaft (LSS or mainshaft) centerline. It is used in conjunction with markers on the low-speed shaft to obtain blade azimuth angle, rotor angular velocity, and the radial position of each blade element. Since the instantaneous radial position of each blade element is determined by ADAMS as the radial distance between the AERO marker and the nacelle marker, it is imperative that marker 2050 be located on the LSS centerline. The marker is oriented such that the Z-axis is parallel to the shaft axis of rotation and x is nominally vertical downward. If the rotor axis is tilted, the x-axis will deviate from vertical by an amount equal to the tilt angle.

4.7 Low-speed shaft markers, ID = 3051, 3052, 3053…

One marker is required on the LSS for each blade. Each marker has its Z-axis parallel to the shaft axis of rotation and its X-axis directed radially outward in the plane formed by the blade span and the low-speed shaft. Thus, each marker has a Z-axis that is parallel to the z axis of marker 2050. The angle between the x axes of marker 2050 and 305n is the azimuth angle of blade n. Zero azimuth is defined when the blade is at the 6:00 position.

If the LSS is composed of beam or field elements, it is important that these LSS markers are located at the revolute joint that supports the LSS. If the markers are located at a point on the part that is not constrained to pure rotation (relative to the nacelle) then the Euler angles that are returned by ADAMS will give misleading information. This results from the fact that an infinitesimal amount of bending in the LSS will tilt the z-axis of these markers relative to the z-axis of the nacelle marker. When this happens the Euler angles between these markers, though correct, will not yield the blade azimuth angle as required by the GFOSUB.

4.8 Pitch reference markers, ID = 4191, 4291, 4391…

A marker is required on the hub at the root of each blade. The markers are used as a reference from which the pitch angle of any blade element can be determined. Each marker must be oriented such that its z-axis is in the plane of rotation and its x-axis is parallel to the x-axes of the (undeflected) AERO markers. Thus, the x-axes of the pitch reference markers will project along the (undeflected) blade span, outwards for a rotor that spins clockwise when looking downwind and inwards for a rotor that spins counterclockwise. The markers will also be coned to match the rotor coning. Marker 4191 is also used in conjunction with the yaw bearing marker (ID 2010) to estimate the rotor hub sling. This is used to determine the rotor hub velocity due to yaw rate requested by AeroDyn from subroutine GetRotorParams of the interface.

5.0 Adams Sensor Statement

ADAMS integrates the equations of motion using a scheme that adjusts the size of the time-step to achieve the desired solution accuracy. When a time step fails to meet convergence criteria, the integrator may step backwards in time and try a smaller step. Thus, the aerodynamics routines must be informed when a successful forward time step has occurred, and they must not update "old" values until a successful time step has been achieved. The SENSUB user-written subroutine provides a convenient method for determining when a successful time step has been completed. SENSUB also checks whether a time period of DTAERO (see the AeroDyn User's Guide) has elapsed since the last aerodynamic calculation, indicating it is time to update the aerodynamic forces.

To invoke the SENSUB subroutine, a SENSOR statement MUST appear in your ADAMS data set. Without this statement, the time-based operations such as dynamic stall calculations and inflow averaging cannot be completed correctly. The subroutines will not always generate an error message if the SENSOR is not present. Therefore, it is essential that users take care to verify the operation of the SENSOR in their model. (Starting in version 10.0 there is one segment of code that tests for the presence of a working SENSOR statement. But this test will not work for very short simulations and has not been thoroughly tested and found reliable in other conditions.)

A typical SENSOR statement appears below. The VALUE and FUNCTION statements are required. The value of VALUE is irrelevant, but the FUNCTION must equal USER with one parameter as an argument. The value of this argument, however, is also inconsequential.

```
SENSOR/11111, VALUE=1., FUNCTION=USER(1.0)
```

6.0 Restrictions

The current version of the subroutines has the following restrictions. Our plan is to eliminate these restrictions over time. But this will require significant changes in the code and will not be done until we have validated the basic operation more fully.

1) The yaw angle is used to estimate the blade position in the skewed wake calculation. It is also used to calculate wind components relative to the rotor plane. This means ADAMS must have the yaw marker and tower-top marker shown in Figure 4.1.

2) The restriction on live twist in earlier versions of the routines has been eliminated. The routines now account for the live pitch of the blade whether the pitch is changing due to elastic twist or control inputs. However, the pitch angle is no longer obtained from the aerodyn.ipt file or the hub-height wind file. This makes it impossible to specify time-varying pitch using either of these files. If you want to specify a time-history of pitch angle you must use statements in the ADAMS data set to generate the desired motion.

7.0 The Request Subroutine (ReqSub.F90)

The distribution files include a sample "Request" subroutine that is useful for obtaining outputs from ADAMS that are in a form that is relatively easy to use and customize. We expect that each user will want to generate outputs that are unique and specific to the particular model. For example, there might be reason for particular interest in the internal bending moments at a specific blade station. The request subroutine can be used to determine those loads of interest and send them to an output file. The sample subroutine obtains yaw moment and a variety of other system parameters such as RPM, tower-top thrust, simulated time, blade azimuth and yaw angle. Any data that can be obtained from SYSARY or SYSFNC (or INFARY and INFFNC) subroutines in ADAMS can easily be accessed using the Request subroutine. This example subroutine is written to work with any ADAMS model that uses the required marker numbers described in this guide, which limits the outputs it creates greatly. The user should make the changes necessary to get the desired outputs from specific markers in their data set.

The data files created by the REQSub.F90 suroutines are written in ASCII format with tab characters delimiting the columns. This makes it easy to import the results into a plotting or spreadsheet application for further processing or display.

This subroutine is invoked by including a statement of the following form in the ADAMS data set and linking the compiled REQSUB.OBJ file with ADAMS:

```
REQUEST/11111, FUNCTION = USER(1,0,0,0,0)
```

Separate REQUEST files based upon blade number are no longer created by the Request subroutines. One file is created named reqsub1.plt, and it will contain all the outputs the user programs in the subroutines. If you desire data from several different blades, write the proper statements in the subroutines. The arguments of the USER function are defined in Table 7.1. These definitions have changed significantly since the last release of the interface. The first 4 parameters, which are not used in the release file, can be used at the discretion of the user for any purpose they wish to program into the code.

NREL has released a general purpose post-processor (GPP) that can be used to extract results from ADAMS output files. This permits use of standard REQUEST statements in your ADAMS data set and eliminates the need for the REQSUB subroutine. It also provides capability for in-depth statistical analysis of the simulation results. NREL can provide information about the availability and use of GPP. For our purposes, however, we prefer the flexibility that tab-delimited output from ReqSub.F90 affords.

Table 7.1 - Arguments of the User-Written Request Function

Argument	Definition
1	No longer used (enter 0).
2	No longer used (enter 0).
3	No longer used (enter 0).
4	No longer used (enter 0).
5	The time in seconds at which you prefer the output file to begin recording data. This is a useful feature that allows start-up transients to be ignored.

8.0 Questions And Answers

This section contains miscellaneous hints and tips for use of ADAMS with the wind turbine aerodynamics subroutines.

8.1 Units

ADAMS is quite versatile in the units it allows. But the aerodynamics subroutines can be used with SI or English units only (unless you want to change the source code). Just be consistent in units between ADAMS and AeroDyn. You may use kips (kN) in the ADAMS data set, and the interface will convert back and forth between lbs and kips (N and kN), but you cannot mix SI and English units.

8.2 Time steps for ADAMS integration

We typically use 120 time steps per second when running ADAMS. Of course, the time step and the error tolerance for the integrator work together to determine the efficiency and accuracy of the solution. We have found that a very tight tolerance does not always lead to a more accurate solution, and it will slow the simulation considerably. A value of approximately 0.01 generally seems adequate. These values may not be appropriate for your model however, so we recommend you try a few different values and verify that your results do not depend strongly upon the size of the time step or the error tolerance. Recent models have been forced to use time steps as small as 0.002 sec. We have also found that run time is much shorter if care is taken to be certain that the output time step is an integer multiple of the HMAX integration time step. This prevents needless interpolation to generate the requested output step.

8.3 Startup problems and solutions.

In the past, starting an ADAMS simulation of a complex, flexible model caused serious problems that often culminated in a failure of the model to run. A variety of ways and a great amount of effort went into remedies for this problem. This seems to be less of a problem with newer versions of ADAMS, and some simple methods have been found to reduce the effects of startup transients on the simulation.

Smoother starts of the rotor speed, using step functions, are still recommended over sudden jumps from 0 rpm to full speed. This seems to help when tower degrees of freedom are active. Abrupt rotor acceleration causes lateral tower bending that damps out slowly in some models. When the tower is rigid, a smooth start takes about the same time as an abrupt start to reach a steady-state solution. A sample step function in the ADAMS data set for rotor speed might read:

```
!                            adams_view_name='rotor_motion'
MOTION/3302
, ROTATIONAL
, VELOCITY
, JOINT = 3302
, FUNCTION = STEP(TIME, 0, 0, 10, 2.1 )
```

This will ramp up the rotor speed from zero to 2.1 radians per second over 10 seconds. This is a useful technique not just for constant rpm models, but for variable speed or synchronous generator models as well. For these models the motion statement can be deactivated using the technique described below.

When you simulate free yaw (or other joint) motion, it may be important to have the startup transients die out before the free motion begins. To accomplish this we have used statements to ACTIVATE and DEACTIVATE joints or other dataset elements in our command files. The technique we use most often to start a free-yaw simulation is to define two yaw joints in your dataset, one a revolute and the other a fixed joint at the same location. In your ADAMS command file you first deactivate the revolute joint and run a simulation for about five seconds. This allows the rotor to establish a "trim solution" in fixed yaw. You then deactivate the fixed joint and activate the revolute joint and run the simulation for the desired time of free yaw motion. For example, if we have a fixed yaw joint with ID # 2021 and a revolute yaw joint with ID # 2022, we turn off the fixed joint and turn on the revolute joint after 10 seconds using our control (acf) file with the statements:

```
DEACTIVATE/JOINT, ID = 2022
ACTIVATE/JOINT, ID = 2021
INTEGRATOR/GSTIFF, INTERPOLATE=ON, HMAX=0.005, KMAX=6, ERROR=0.001
SIMULATE/DYNAMIC, end=10., dtout=0.05
DEACTIVATE/JOINT, ID = 2021
ACTIVATE/JOINT, ID = 2022
INTEGRATOR/GSTIFF, INTERPOLATE=ON, HMAX=0.005, KMAX=6, ERROR=0.001
SIMULATE/DYNAMIC, end=60., dtout=0.05
stop
```

See the ADAMS manuals for more information on this powerful and versatile technique.

We have also found it useful for some models to "loosen up" error tolerances on startup, even for just the first half second, when the model tends to "relax" from its undeflected state due to the instantaneous application of gravity, and other suddenly applied forces.

8.4 Comment lines in ADAMS.

ADAMS has two ways to mark a comment line in the data set. You can place an exclamation mark in the first column or you can leave the first five (or more) columns blank. This second method has given us some surprises. We have spent too much time debugging data sets only to find a line that appeared to have no errors was ignored because it had five or more leading spaces.

8.5 Products of inertia.

Alan Wright of NREL pointed out that ADAMS solutions seem to proceed more smoothly when the PARTS have non-zero products of inertia (terms such as I_{xy}). A teetering rotor model was developed with all the products of inertia set to zero. When it ran, the integrator occasionally had to rotate parts to avoid singularities. When all of the product of inertia terms were set to non-zero, but negligibly small values, the part rotations were avoided with a subsequent saving of CPU time and no change in the final results. This is not fully understood, but it seems to help the simulations.

8.6 Additional Debugging Techniques.

It is not uncommon to have problems running a new model because of marker misalignment, pitch angle errors, or similar simple but fatal errors. These are some ideas you might use to isolate and identify the problem.

1. Examine the message (.msg) and GFOSUB.OPT files very closely. Errors or warnings generated by the aerodynamics routines are written to the screen and to the .msg file (as well as the error.log file). It is easy to miss a warning written to the screen when it is scrolling, so the .msg provides a permanent record. The GFOSUB.OPT file contains information about your inputs. The blade element pitch angles are calculated by ADAMS using the same markers and methods that are employed in the aerodynamics calculations. This file is the key to determining whether your aerodynamic force markers are located and oriented correctly.

2. Turn off the wake calculations, and turn on the dynamic stall option. The wake calculations are time consuming and can increase the likelihood of numerical problems. If you are having problems running a new model, you can set the WAKE flag to NONE in the aerodyn.ipt file to speed the calculations. Without the correct induced velocities your loads and power output will not be correct, but they should be adequate for debugging. (Be sure to turn the wake calculations back on when you finish debugging the model.)

Dynamic stall usually increases the aerodynamic damping and stabilizes the models. When you are calculating the induced velocities, the dynamic inflow will also tend to provide aerodynamic damping. Since the aerodynamic damping is the largest contribution to total damping of a blade, these increases can be significant. Run your model in low winds to increase the aerodynamic damping. All of these steps will help get your model through the startup transients. After the model is debugged by running conditions with high damping, you should have little trouble running it in conditions with lighter damping.

3. Turn on the element file option (to create the reqelem.plt file) in the aerodyn.ipt file (see AeroDyn User's Guide). Examination of the wind velocities and aerodynamics parameters in this file may give insight to problems with the model.

9.0 SAMPLE ADAMS DATA SET

A sample ADAMS data set is included on the distribution disk. It may be helpful to examine this model to see some of the details that are not covered in this manual.

References

Hansen, A. C., 1992, "Yaw Dynamics of Horizontal Axis Wind Turbines: Final Report." National Renewable Energy Laboratory, NREL Technical Report, TP 442-4822.

Hansen, A. C., C. P. Butterfield and X. Cui, 1990, "Yaw Loads and Motions of a Horizontal Axis Wind Turbine." Journal of Solar Energy Engineering. Vol. 112, No. 4. pp. 310-314.

Hansen, A. C. and X. Cui, 1989, "Analyses and Observations of Wind Turbine Yaw Dynamics." ASME Journal of Solar Energy Engineering. Vol. 111, No. 4. pp. 367-371.

Hansen, A. C. and A. D. Wright, 1991, "A Comparison of Combined Experiment Flap Load Predictions by the FLAP and YawDyn Codes." Tenth ASME Wind Energy Symposium, Houston, TX.

Hansen, A. C. 1993. "Structural Response to Unsteady and Stall Aerodynamics." 1993 DOE Wind Energy Program Review Meeting, Golden, CO.

Hansen, A. C., Butterfield, C. P., 1993. Aerodynamics of Horizontal-Axis Wind Turbines. Annual Review of Fluid Mechanics, 1993. Palo Alto, Annual Reviews, Inc.

Laino, D., Butterfield, C. P. 1994. "Using YawDyn to Model Turbines with Aerodynamic Control Systems." ASME Wind Energy Conference, New Orleans, LA.

Leishman, J. G., Beddoes, T. S., 1989. A Semi-Empirical Model for Dynamic Stall. Journal of the American Helicopter Society. 34(3): 3-17.

Malcolm, D. J., Wright, A. D. 1994. "The Use of ADAMS to Model the AWT-26 Prototype." ASME Wind Energy Symposium, New Orleans.

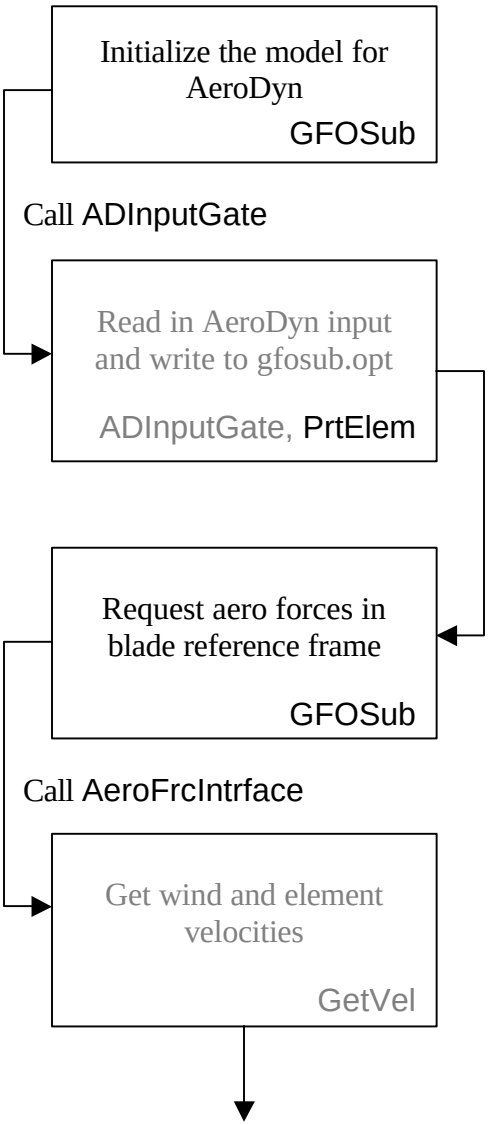
Peters, D. A., HaQuang, N., 1988. Dynamic Inflow for Practical Applications. Journal of the American Helicopter Society. 33(4): 64-68.

Appendix A. Flow Chart of the ADAMS to AeroDyn Interface

This appendix provides a simplified flow chart to illustrate how the ADAMS to AeroDyn interface operates. The purpose of the chart is to familiarize the user with the method that is used and some of the key assumptions. The chart does not map the flow of the entire code, nor does it use the format of traditional software flowcharts. The chart focuses on the operations and interactions between the codes, rather than the code or subroutine structure. Each box of the chart does, however, indicate the name of the subroutine(s) in which the procedure is performed (if known) in the lower, right-hand corner of the box, in Arial font. This is intended to assist users who wish to examine the details within the subroutines.

This chart deals only with the GFOSub part of the interface. The ReqSub and SenSub are not addressed. Tasks and subroutines that belong to the Interface are indicated in black, while those that belong to AeroDyn are depicted in grey.

Flow Chart of the ADAMS to AeroDyn Interface
June, 2001



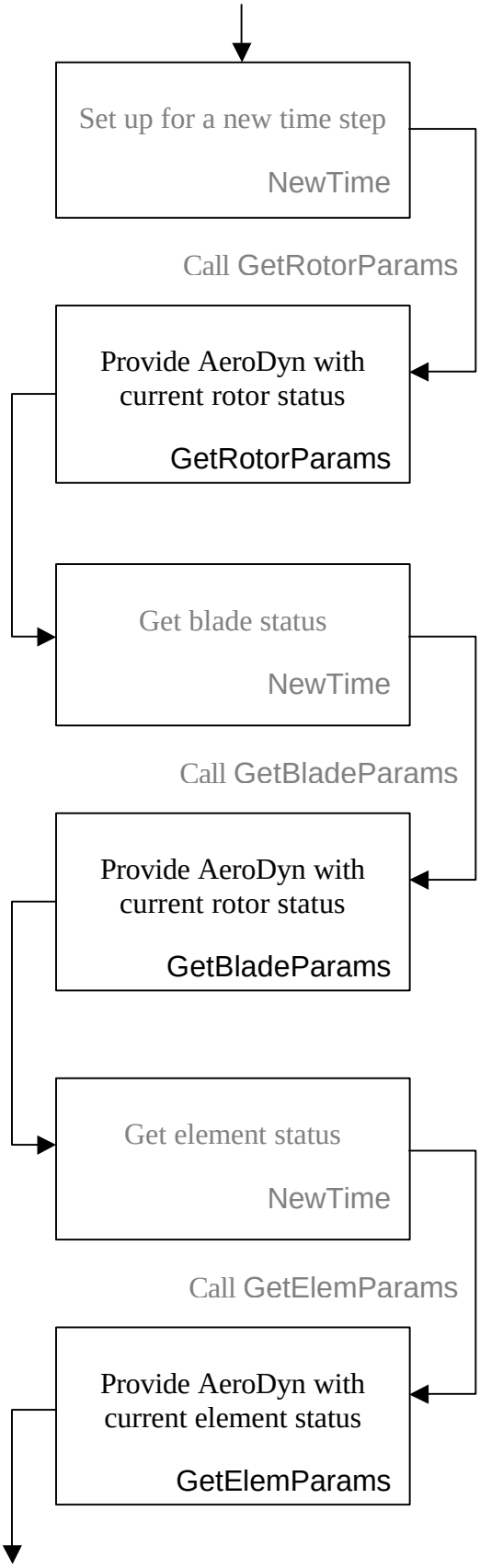
- The first NB (number of blade) times NELM (number of element) calls to GFOSub are used to determine the number of blades in the model.

- Get aerodynamics model parameters
- Write out AeroDyn input parameters
- Write out blade geometry data from ADAMS

- After forward time step and DTAERO seconds have passed since last aero calculation, get aero forces from AeroDyn

- This is the gateway into AeroDyn to get the element aerodynamic forces. Control is turned over to AeroDyn at this point.

- Get the velocities of the wind and element.



- At new time step, initialize parameters that change once per time step

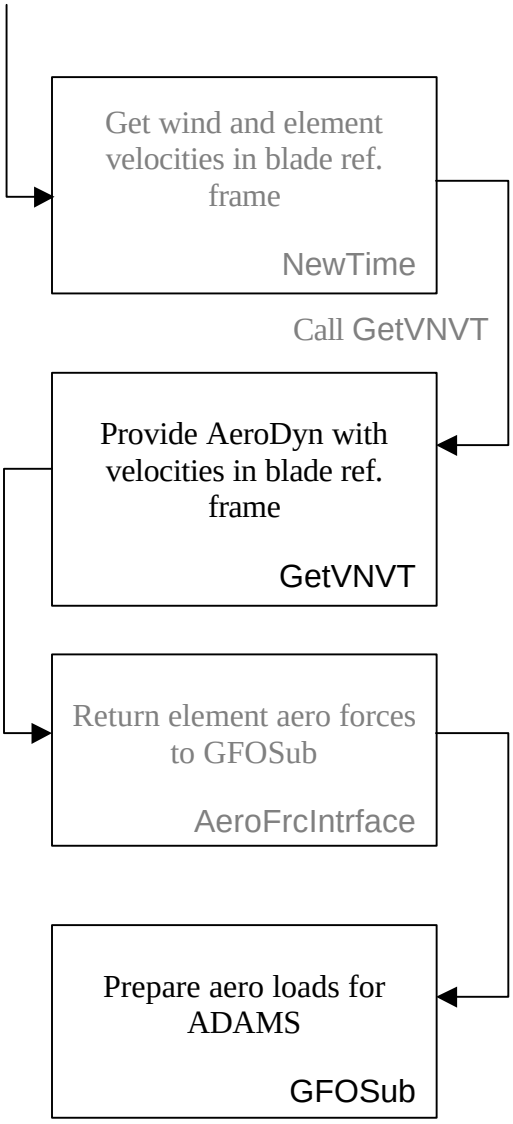
- Get the rotor status:
  - Rotor speed
  - Yaw angle
  - Hub velocity due to yaw rate
  - Nacelle tilt angle

- If call is for first element on a blade, initialize parameters that change for each blade

- Get parameters unique to each blade:
  - Blade azimuth angle

- For each element, initialize parameters for that element

- Get element status:
  - Multiple airfoil table location
  - Element pitch angle
  - Element radius
  - Element location in XYZ frame



- Get the wind velocity in the ground reference frame at the element position.

- Get element velocities and wind velocities in the blade reference frame (normal and tangential)

- Calculate the aerodynamic forces on the element

- Return the element forces to GFOSub in the blade reference frame.
- Return program control to ADAMS Interface

- Transform the aerodynamic loads from the blade to the element reference frame.
- Convert loads to proper units and return them to ADAMS.

INDEX

M

P

R

S

T

U

W

Y