Rotation Notation/Convention

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = [\Lambda] \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{bmatrix} \hat{x}^T \\ \hat{y}^T \\ \hat{z}^T \end{bmatrix} \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \text{ (from global to local)}$$

or equivalently:

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = [\Lambda]^T \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} == \begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

where *X*/*Y*/*Z* are global coordinates, *x*/*y*/*z* are local coordinates, Λ is the DCM from global to local, and $\hat{x}$ / $\hat{y}$ / $\hat{z}$ are the unit vectors of the local coordinate system expressed in the global coordinate system.

$$\begin{Bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix} = F^{EulerExtract}\left(\left[\Lambda\left(\theta_x,\theta_y,\theta_z\right)\right]\right)$$

where function $F^{EulerExtract}(\ )$ returns the 3 Euler angles of the *x*-*y*-*z* (1-2-3) rotation sequence used to form Λ (that is, first a rotation θ_x about the global *X* axis, followed by rotation θ_y about the *Y'* axis, followed by rotation θ_z about the *Z''* axis) defined as follows:

$$\Lambda\left(\theta_x,\theta_y,\theta_z\right) = \begin{bmatrix} COS(\theta_z) & SIN(\theta_z) & 0 \\ -SIN(\theta_z) & COS(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} COS(\theta_y) & 0 & -SIN(\theta_y) \\ 0 & 1 & 0 \\ SIN(\theta_y) & 0 & COS(\theta_y) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & COS(\theta_x) & SIN(\theta_x) \\ 0 & -SIN(\theta_x) & COS(\theta_x) \end{bmatrix}$$

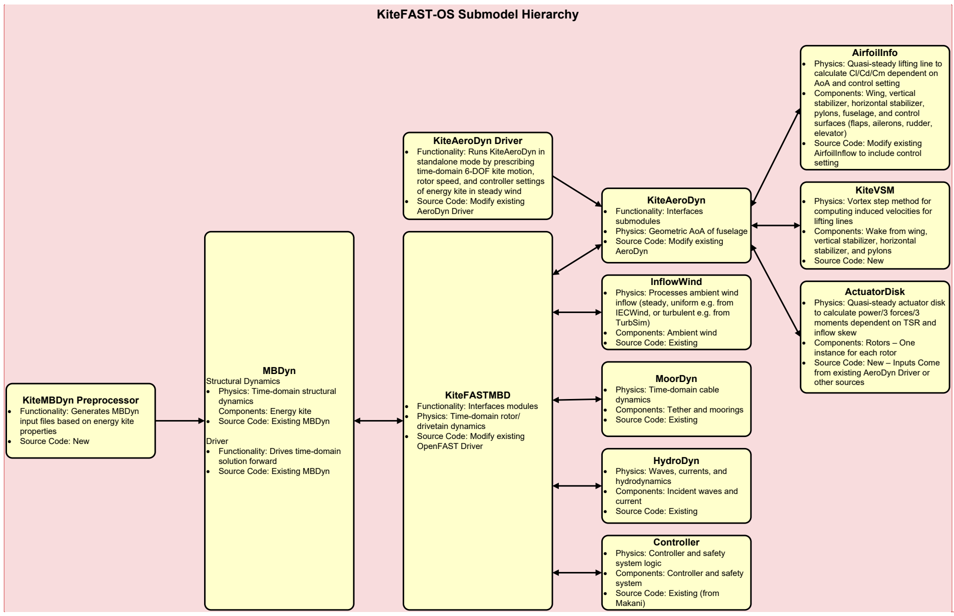
$$= \begin{bmatrix} COS(\theta_y)COS(\theta_z) & COS(\theta_x)SIN(\theta_z)+SIN(\theta_x)SIN(\theta_y)COS(\theta_z) & SIN(\theta_x)SIN(\theta_z)-COS(\theta_x)SIN(\theta_y)COS(\theta_z) \\ -COS(\theta_y)SIN(\theta_z) & COS(\theta_x)COS(\theta_z)-SIN(\theta_x)SIN(\theta_y)SIN(\theta_z) & SIN(\theta_x)COS(\theta_z)+COS(\theta_x)SIN(\theta_y)SIN(\theta_z) \\ SIN(\theta_y) & -SIN(\theta_x)COS(\theta_y) & COS(\theta_x)COS(\theta_y) \end{bmatrix}$$

Note the following simplifications:

$$\Lambda\left(0,\theta_y,\theta_z\right) = \begin{bmatrix} COS(\theta_y)COS(\theta_z) & SIN(\theta_z) & -SIN(\theta_y)COS(\theta_z) \\ -COS(\theta_y)SIN(\theta_z) & COS(\theta_z) & SIN(\theta_y)SIN(\theta_z) \\ SIN(\theta_y) & 0 & COS(\theta_y) \end{bmatrix}$$

$$\Lambda\left(\theta_x,0,\theta_z\right) = \begin{bmatrix} COS(\theta_z) & COS(\theta_x)SIN(\theta_z) & SIN(\theta_x)SIN(\theta_z) \\ -SIN(\theta_z) & COS(\theta_x)COS(\theta_z) & SIN(\theta_x)COS(\theta_z) \\ 0 & -SIN(\theta_x) & COS(\theta_x) \end{bmatrix}$$

$$\Lambda\left(\theta_x,\theta_y,0\right) = \begin{bmatrix} COS(\theta_y) & SIN(\theta_x)SIN(\theta_y) & -COS(\theta_x)SIN(\theta_y) \\ 0 & COS(\theta_x) & SIN(\theta_x) \\ SIN(\theta_y) & -SIN(\theta_x)COS(\theta_y) & COS(\theta_x)COS(\theta_y) \end{bmatrix}$$



Commented [JJ1]: We've split up KiteFASTMBD into KiteFASTMBD in C and KiteFASTMBD in Fortran. This plan is for KiteFASTMBD in Fortran.

KiteFASTMBD

Inputs	Outputs	States	Parameters
• $^{MBD}\vec{p}^{Ptfm}$ – Position of the floating platform (m) • $^{MBD}\Lambda^{Ptfm}$ – Rotation (absolute orientation) of the floating platform (-) • $^{MBD}\vec{v}^{Ptfm}$ – Translational velocity (absolute) of the floating platform (m/s) • $^{MBD}\vec{\omega}^{Ptfm}$ – Rotational velocity (absolute) of the floating platform (rad/s) • $^{MBD}\vec{a}^{Ptfm}$ – Translational acceleration (absolute) of the floating platform (m/s^2) • $^{MBD}\vec{\alpha}^{Ptfm}$ – Rotational acceleration (absolute) of the floating platform (rad/s^2) • $^{MBD}\vec{p}^{PtfmIMU}$ – Position of the floating platform IMU (m) • $^{MBD}\Lambda^{PtfmIMU}$ – Rotation	• $^{MBD}\vec{F}^{Ptfm}$ – Hydrodynamic, mooring, and tether applied concentrated forces on the floating platform (N) • $^{MBD}\vec{M}^{Ptfm}$ – Hydrodynamic, mooring, and tether applied concentrated moments on the floating platform (N-m) • $^{MBD}\vec{F}_j^{Fus}$ – Aerodynamic applied concentrated forces at the j th node of the fuselage mesh (N) • $^{MBD}\vec{M}_j^{Fus}$ – Aerodynamic applied concentrated moments at the j th node of the fuselage mesh (N-m) • $^{MBD}\vec{F}_j^{SWn}$ – Aerodynamic and tether applied	• $^{KAD}NewTime$ – Is this a new time step (in order to only call KiteAeroDyn once per step)? (flag) (other state) • $^{Ctrl}NewTime$ – Is this a new time step (in order to only call the controller once per step)? (flag) (other state) • $^{MBD}OtherStates$ – Inputs from MBDyn from the previous time step (stored as other states) • $^{MD[n_2]}OtherStates$ – Inputs to both instances of MoorDyn from the previous time step (stored as other states)	• Δt – MBDyn time step (s) • $^{KAD}\Delta t$ – KiteAeroDyn time step (s) • $InterpOrder$ – Interpolation order for input/output time history (-) {1=linear, 2=quadratic} • $N_{KAD/MBD}$ – Number of KiteAeroDyn time steps per MBDyn time step (-) • $N_{Ctrl/MBD}$ – Number of controller time steps per MBDyn time step (-) • N_{Flaps} – Number of flaps per wing (-) • N_{Pylons} – Number of pylons per wing (-) • $\Lambda^{FAST2Ctrl}$ – DCM conversion from the FAST ground system (X pointed

Commented [JJ2]: These are the data queried from the MBDyn model at t using GetXCur to be used within KiteFASTMBD. The outputs from MBDyn are inputs to KiteFASTMBD.

Commented [JJ3]: These are the data sent to the MBDyn model from KiteFASTMBD. The inputs to MBDyn are outputs from KiteFASTMBD.

Commented [JJ4]: Obvious parameters are not listed here.

(absolute orientation) of the floating platform IMU (-) • ${}^{MBD}\vec{v}^{PtfmIMU}$ – Translational velocity (absolute) of the floating platform IMU (m/s) • ${}^{MBD}\vec{\omega}^{PtfmIMU}$ – Rotational velocity (absolute) of the floating platform IMU (rad/s) • ${}^{MBD}\vec{a}^{PtfmIMU}$ – Translational acceleration (absolute) of the floating platform IMU (m/s^2) • ${}^{MBD}\vec{p}^{BSRef}$ – Position of the floating platform base station (BS) reference point (m) • ${}^{MBD}\Lambda^{BSRef}$ – Rotation (absolute orientation) of the floating platform BS reference point (-) • ${}^{MBD}\vec{v}^{BSRef}$ – Translational velocity (absolute) of the floating platform BS reference point (m/s) • ${}^{MBD}\vec{\omega}^{BSRef}$ – Rotational velocity (absolute) of the floating platform BS reference point (rad/s) • ${}^{MBD}\vec{a}^{BSRef}$ – Translational acceleration (absolute) of the floating platform BS reference point (m/s^2) • ${}^{MBD}\vec{p}^{Wind}$ – Position of the base station where the wind measurement on the floating platform is taken (m) • ${}^{MBD}\vec{v}^{Wind}$ – Translational velocity (absolute) of the base station where the wind measurement on the floating platform is taken (m/s) • ${}^{MBD}\vec{p}^{FusO}$ – Position (origin) of the fuselage (m) • ${}^{MBD}\Lambda^{FusO}$ – Rotation (absolute orientation) of the	concentrated forces at the j th node of the starboard wing mesh (N) • ${}^{MBD}\vec{M}_j^{SWn}$ – Aerodynamic applied and tether concentrated moments at the j th node of the starboard wing mesh (N-m) • ${}^{MBD}\vec{F}_j^{PWn}$ – Aerodynamic and tether applied concentrated forces at the j th node of the port wing mesh (N) • ${}^{MBD}\vec{M}_j^{PWn}$ – Aerodynamic and tether applied concentrated moments at the j th node of the port wing mesh (N-m) • ${}^{MBD}\vec{F}_j^{VS}$ – Aerodynamic applied concentrated forces at the j th node of the vertical stabilizer mesh (N) • ${}^{MBD}\vec{M}_j^{VS}$ – Aerodynamic applied concentrated moments at the j th node of the vertical stabilizer mesh (N-m) • ${}^{MBD}\vec{F}_j^{SHS}$ – Aerodynamic applied concentrated forces at the j th node of the starboard horizontal stabilizer mesh (N) • ${}^{MBD}\vec{M}_j^{SHS}$ – Aerodynamic applied concentrated moments at the j th node of the starboard horizontal stabilizer mesh (N-m) • ${}^{MBD}\vec{F}_j^{PHS}$ – Aerodynamic applied concentrated forces at the j th node of the port horizontal stabilizer mesh (N)	• ${}^{MD[n_2]}x$ – MoorDyn continuous states for both instances (varied) • ${}^{HD}OtherStates^u$ – Inputs to HydroDyn from the previous time step (stored as other states) • ${}^{HD}x$ – HydroDyn continuous states (varied) • ${}^{HD}x^d$ – HydroDyn discrete-time states (varied) • ${}^{HD}OtherStates$ – HydroDyn other states (varied) • ${}^{KAD}z$ – KiteAeroDyn constraint states (varied) • ${}^{KAD}u(:)$ – Time history of KiteAeroDyn inputs (stored as other states) • ${}^{KAD}y(:)$ – Time history of KiteAeroDyn outputs (stored as other states) • ${}^{KAD}t(:)$ – Times associated with history of KiteAeroDyn inputs and outputs (stored as other states)	nominally downwind; Z pointed vertically opposite gravity; Y transverse) to the ground system used by the controller (X pointed nominally upwind; Z pointed vertically downward, Y transverse) (-) • ${}^{MBD}\vec{g}$ – Gravity vector expressed in the global inertial-frame coordinate system (m/s^2) • ρ – Air density (kg/m^3) • $\vec{p}^{BSRefR}$ – Undisplaced position in the floating platform of the BS reference point (m) • ${}^{MBD}m^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Mass of the top and bottom rotors/drivetrains on the pylons on the starboard wing mesh (kg) • ${}^{MBD}I_{Rot}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Rotational inertia about the shaft axis of the top and bottom rotors/drivetrains on the pylons on the starboard wing mesh (kg·m^2) • ${}^{MBD}I_{Tran}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Transverse inertia about the rotor reference point of the top and bottom rotors/drivetrains on the pylons on the starboard wing mesh (kg·m^2) • ${}^{MBD}x_{CM}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Distance along the shaft from the rotor reference point of the top and bottom rotors/drivetrains on the pylons on the starboard wing mesh to the center of mass of the rotor/drivetrain (positive along positive x) (m) • ${}^{MBD}m^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Mass of the top and bottom rotors/drivetrains

Commented [JJ6]: The first instance of MoorDyn is for the tether; the second instance of MoorDyn is for the mooring system.

Commented [JJ7]: The first instance of MoorDyn is for the tether; the second instance of MoorDyn is for the mooring system.

Commented [JJ8]: The first instance of MoorDyn is for the tether; the second instance of MoorDyn is for the mooring system.

Commented [JJ9]: The first instance of MoorDyn is for the tether; the second instance of MoorDyn is for the mooring system.

Commented [JJ5]: These points should move rigidly with the floating platform i.e. the orientation and rotational velocity are the same as that of the floating platform.

fuselage origin (-) • $^{MBD}\vec{v}^{FusO}$ – Translational velocity (absolute) of the fuselage origin (m/s) • $^{MBD}\vec{\omega}^{FusO}$ – Rotational velocity (absolute) of the fuselage origin (rad/s) • $^{MBD}\vec{a}^{FusO}$ – Translational acceleration (absolute) of the fuselage origin (m/s^2) • $^{MBD}\vec{\alpha}^{FusO}$ – Rotational acceleration (absolute) of the fuselage origin (rad/s^2) • $^{MBD}\vec{p}_j^{Fus}$ – Translational position (absolute) of the j th node of the fuselage mesh (m) • $^{MBD}\Lambda_j^{Fus}$ – Displaced rotation (absolute orientation) of the j th node of the fuselage mesh (-) • $^{MBD}\vec{v}_j^{Fus}$ – Translational velocity (absolute) of the j th node of the fuselage mesh (m/s) • $^{MBD}\vec{\omega}_j^{Fus}$ – Rotational velocity (absolute) of the j th node of the fuselage mesh (rad/s) • $^{MBD}\vec{a}_j^{Fus}$ – Translational acceleration (absolute) of the j th node of the fuselage mesh (m/s^2) • $^{MBD}\vec{FR}_j^{Fus}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the fuselage mesh (N) • $^{MBD}\vec{MR}_j^{Fus}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the fuselage mesh (N-m) • $^{MBD}\vec{p}^{SWnO}$ – Position (origin) of the starboard	• $^{MBD}\vec{M}_j^{PHS}$ – Aerodynamic applied concentrated moments at the j th node of the port horizontal stabilizer mesh (N-m) • $^{MBD}\vec{F}_j^{SPy}\left[n_{Pylons}\right]$ – Aerodynamic applied concentrated forces at the j th node of the pylons on the starboard wing mesh (N) • $^{MBD}\vec{M}_j^{SPy}\left[n_{Pylons}\right]$ – Aerodynamic applied concentrated moments at the j th node of pylons on the starboard wing mesh (N-m) • $^{MBD}\vec{F}_j^{PPy}\left[n_{Pylons}\right]$ – Aerodynamic applied concentrated forces at the j th node of the pylons on the port wing mesh (N) • $^{MBD}\vec{M}_j^{PPy}\left[n_{Pylons}\right]$ – Aerodynamic applied concentrated moments at the j th node of pylons on the port wing mesh (N-m) • $^{MBD}\vec{F}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Concentrated reaction forces at the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (N) • $^{MBD}\vec{M}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Concentrated reaction moments at the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (N-m) • $^{MBD}\vec{F}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Concentrated reaction forces at the top and bottom nacelles on the pylons on the port wing		on the pylons on the port wing mesh (kg) • $^{MBD}I_{Rot}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Rotational inertia about the shaft axis of the top and bottom rotors/drivetrains on the pylons on the port wing mesh (kg·m^2) • $^{MBD}I_{Tran}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Transverse inertia about the rotor reference point of the top and bottom rotors/drivetrains on the pylons on the port wing mesh (kg·m^2) • $^{MBD}x_{CM}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Distance along the shaft from the rotor reference point of the top and bottom rotors/drivetrains on the pylons on the port wing mesh to the center of mass of the rotor/drivetrain (positive along positive x) (m) • $^{MBD}\vec{p}_j^{FusR}$ – Reference position of the j th node of the fuselage mesh (m) • $^{MBD}\Lambda_j^{FusR}$ – Reference orientation of the j th node of the fuselage mesh (-) • $^{MBD}\vec{p}_j^{SWnR}$ – Reference position of the j th node of the starboard wing mesh (m) • $^{MBD}\Lambda_j^{SWnR}$ – Reference orientation of the j th node of the starboard wing mesh (-) • $^{MBD}\vec{p}_j^{PWnR}$ – Reference position of the j th node of the port wing mesh (m) • $^{MBD}\Lambda_j^{PWnR}$ – Reference orientation of the j th node of the port wing

wing (m) • ${}^{MBD}\vec{p}_j^{SWn}$ – Translational position (absolute) of the j th node of the starboard wing (m) • ${}^{MBD}\Lambda_j^{SWn}$ – Displaced rotation (absolute orientation) of the j th node of the starboard wing mesh (-) • ${}^{MBD}\vec{v}_j^{SWn}$ – Translational velocity (absolute) of the j th node of the starboard wing mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{SWn}$ – Rotational velocity (absolute) of the j th node of the starboard wing mesh (rad/s) • ${}^{MBD}\vec{a}_j^{SWn}$ – Translational acceleration (absolute) of the j th node of the starboard wing mesh (m/s^2) • ${}^{MBD}\vec{F}R_j^{SWn}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the starboard wing mesh (N) • ${}^{MBD}\vec{M}R_j^{SWn}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the starboard wing mesh (N-m) • ${}^{MBD}\vec{p}^{PWnO}$ – Position (origin) of the port wing (m) • ${}^{MBD}\vec{p}_j^{PWn}$ – Translational position (absolute) of the j th node of the port wing mesh (m) • ${}^{MBD}\Lambda_j^{PWn}$ – Displaced rotation (absolute orientation) of the j th node of the port wing mesh (-) • ${}^{MBD}\vec{v}_j^{PWn}$ – Translational	mesh at the rotor reference point (N) • ${}^{MBD}\vec{M}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Concentrated reaction moments at the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (N-m)		mesh (-) • ${}^{MBD}\vec{p}_j^{VSR}$ – Reference position of the j th node of the vertical stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{VSR}$ – Reference orientation of the j th node of the vertical stabilizer mesh (-) • ${}^{MBD}\vec{p}_j^{SHSR}$ – Reference position of the j th node of the starboard horizontal stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{SHSR}$ – Reference orientation of the j th node of the starboard horizontal stabilizer mesh (-) • ${}^{MBD}\vec{p}_j^{PHSR}$ – Reference position of the j th node of the port horizontal stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{PHSR}$ – Reference orientation of the j th node of the port horizontal stabilizer mesh (-) • ${}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right]$ – Reference position of the j th node of the pylons on the starboard wing mesh (m) • ${}^{MBD}\Lambda_j^{SPyR}\left[n_{Pylons}\right]$ – Reference orientation of the j th node of the pylons on the starboard wing mesh (-) • ${}^{MBD}\vec{p}_j^{PPyR}\left[n_{Pylons}\right]$ – Reference position of the j th node of the pylons on the port wing mesh (m) • ${}^{MBD}\Lambda_j^{PPyR}\left[n_{Pylons}\right]$ – Reference orientation of the j th node of the pylons on the port wing mesh (-)

velocity (absolute) of the j th node of the port wing mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{PWn}$ – Rotational velocity (absolute) of the j th node of the port wing mesh (rad/s) • ${}^{MBD}\vec{a}_j^{PWn}$ – Translational acceleration (absolute) of the j th node of the port wing mesh (m/s²) • ${}^{MBD}\vec{F}R_j^{PWn}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the port wing mesh (N) • ${}^{MBD}\vec{M}R_j^{PWn}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the port wing mesh (N-m) • ${}^{MBD}\vec{p}^{VSO}$ – Position (origin) of the vertical stabilizer (m) • ${}^{MBD}\vec{p}_j^{VS}$ – Translational position (absolute) of the j th node of the vertical stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{VS}$ – Displaced rotation (absolute orientation) of the j th node of the vertical stabilizer mesh (-) • ${}^{MBD}\vec{v}_j^{VS}$ – Translational velocity (absolute) of the j th node of the vertical stabilizer mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{VS}$ – Rotational velocity (absolute) of the j th node of the vertical stabilizer mesh (rad/s) • ${}^{MBD}\vec{a}_j^{VS}$ – Translational acceleration (absolute) of the j th node of the vertical stabilizer mesh (m/s²)			• ${}^{MBD}\vec{p}^{SPyRtrR}\left[n_{Pylons},n_2\right]$ – Reference positions (origins) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (m) • ${}^{MBD}\Lambda^{SPyRtrR}\left[n_{Pylons},n_2\right]$ – Reference orientations of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (-) • ${}^{MBD}\vec{p}^{PPyRtrR}\left[n_{Pylons},n_2\right]$ – Reference positions (origins) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (m) • ${}^{MBD}\Lambda^{PPyRtrR}\left[n_{Pylons},n_2\right]$ – Reference orientations of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (-)

• ${}^{MBD}\vec{F}R_j^{VS}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the vertical stabilizer mesh (N) • ${}^{MBD}\vec{M}R_j^{VS}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the vertical stabilizer mesh (N-m) • ${}^{MBD}\vec{p}^{SHSO}$ – Position (origin) of the starboard horizontal stabilizer (m) • ${}^{MBD}\vec{p}_j^{SHS}$ – Translational position (absolute) of the j th node of the starboard horizontal stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{SHS}$ – Displaced rotation (absolute orientation) of the j th node of the starboard horizontal stabilizer mesh (-) • ${}^{MBD}\vec{v}_j^{SHS}$ – Translational velocity (absolute) of the j th node of the starboard horizontal stabilizer mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{SHS}$ – Rotational velocity (absolute) of the j th node of the starboard horizontal stabilizer mesh (rad/s) • ${}^{MBD}\vec{a}_j^{SHS}$ – Translational acceleration (absolute) of the j th node of the starboard horizontal stabilizer mesh (m/s^2) • ${}^{MBD}\vec{F}R_j^{SHS}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the starboard horizontal stabilizer mesh (N)			

• ${}^{MBD}\vec{M}R_j^{SHS}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the starboard horizontal stabilizer mesh (N-m) • ${}^{MBD}\vec{p}^{PHSO}$ – Position (origin) of the port horizontal stabilizer (m) • ${}^{MBD}\vec{p}_j^{PHS}$ – Translational position (absolute) of the j th node of the port horizontal stabilizer mesh (m) • ${}^{MBD}\Lambda_j^{PHS}$ – Displaced rotation (absolute orientation) of the j th node of the port horizontal stabilizer mesh (-) • ${}^{MBD}\vec{v}_j^{PHS}$ – Translational velocity (absolute) of the j th node of the port horizontal stabilizer mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{PHS}$ – Rotational velocity (absolute) of the j th node of the port horizontal stabilizer mesh (rad/s) • ${}^{MBD}\vec{a}_j^{PHS}$ – Translational acceleration (absolute) of the j th node of the port horizontal stabilizer mesh (m/s^2) • ${}^{MBD}\vec{F}R_j^{PHS}$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the port horizontal stabilizer mesh (N) • ${}^{MBD}\vec{M}R_j^{PHS}$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the port horizontal stabilizer mesh (N-m)			

• ${}^{MBD}\vec{p}^{SPyO}\left[n_{Pylons}\right]$ – Positions (origins) of pylons on the starboard wing (m) • ${}^{MBD}\vec{p}_j^{SPy}\left[n_{Pylons}\right]$ – Translational position (absolute) of the j th node of the pylons on the starboard wing mesh (m) • ${}^{MBD}\Lambda_j^{SPy}\left[n_{Pylons}\right]$ – Displaced rotation (absolute orientation) of the j th node of the pylons on the starboard wing mesh (-) • ${}^{MBD}\vec{v}_j^{SPy}\left[n_{Pylons}\right]$ – Translational velocity (absolute) of the j th node of the pylons on the starboard wing mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{SPy}\left[n_{Pylons}\right]$ – Rotational velocity (absolute) of the j th node of the pylons on the starboard wing mesh (rad/s) • ${}^{MBD}\vec{a}_j^{SPy}\left[n_{Pylons}\right]$ – Translational acceleration (absolute) of the j th node of the pylons on the starboard wing mesh (m/s^2) • ${}^{MBD}\vec{F}R_j^{SPy}\left[n_{Pylons}\right]$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the pylons on the starboard wing mesh (N) • ${}^{MBD}\vec{M}R_j^{SPy}\left[n_{Pylons}\right]$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the pylons on the starboard wing mesh (N-m) • ${}^{MBD}\vec{p}^{PPyO}\left[n_{Pylons}\right]$ – Positions (origins) of pylons on the port wing			

(m) • ${}^{MBD}\vec{p}_j^{PPy}\left[n_{Pylons}\right]$ – Translational position (absolute) of the j th node of the pylons on the port wing mesh (m) • ${}^{MBD}\Lambda_j^{PPy}\left[n_{Pylons}\right]$ – Displaced rotation (absolute orientation) of the j th node of the pylons on the port wing mesh (-) • ${}^{MBD}\vec{v}_j^{PPy}\left[n_{Pylons}\right]$ – Translational velocity (absolute) of the j th node of the pylons on the port wing mesh (m/s) • ${}^{MBD}\vec{\omega}_j^{PPy}\left[n_{Pylons}\right]$ – Rotational velocity (absolute) of the j th node of the pylons on the port wing mesh (rad/s) • ${}^{MBD}\vec{a}_j^{PPy}\left[n_{Pylons}\right]$ – Translational acceleration (absolute) of the j th node of the pylons on the port wing mesh (m/s^2) • ${}^{MBD}\vec{F}R_j^{PPy}\left[n_{Pylons}\right]$ – Reaction force (expressed in the local coordinate system) at the j th Gauss point of the pylons on the port wing mesh (N) • ${}^{MBD}\vec{M}R_j^{PPy}\left[n_{Pylons}\right]$ – Reaction moment (expressed in the local coordinate system) at the j th Gauss point of the pylons on the port wing mesh (N-m) • ${}^{MBD}\vec{p}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Translational position (absolute) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (m)			

• ${}^{MBD}\Lambda^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Displaced rotation (absolute orientation) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (-) • ${}^{MBD}\vec{v}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Translational velocity (absolute) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (m/s) • ${}^{MBD}\vec{\omega}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Rotational velocity (absolute) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (rad/s) • ${}^{MBD}\vec{a}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Translational acceleration (absolute) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (m/s^2) • ${}^{MBD}\vec{\alpha}^{SPyRtr}\left[n_{Pylons}, n_2\right]$ – Rotational acceleration (absolute) of the top and bottom nacelles on the pylons on the starboard wing mesh at the rotor reference point (rad/s^2) • ${}^{MBD}\vec{p}^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Translational position (absolute) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (m) • ${}^{MBD}\Lambda^{PPyRtr}\left[n_{Pylons}, n_2\right]$ – Displaced rotation (absolute orientation) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (-)			

• $^{MBD}\vec{v}^{PPyRtr}\left[n_{Pylons},n_2\right]$ – Translational velocity (absolute) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (m/s) $^{MBD}\vec{\omega}^{PPyRtr}\left[n_{Pylons},n_2\right]$ – Rotational velocity (absolute) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (rad/s) • $^{MBD}\vec{a}^{PPyRtr}\left[n_{Pylons},n_2\right]$ – Translational acceleration (absolute) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (m/s^2) • $^{MBD}\vec{\alpha}^{PPyRtr}\left[n_{Pylons},n_2\right]$ – Rotational acceleration (absolute) of the top and bottom nacelles on the pylons on the port wing mesh at the rotor reference point (rad/s^2)			

MiscVars: ^{Ctrl}y, ^{HD}y, $^{MD[n_2]}y$, ^{IfW}y, ^{KAD}y, ^{MBD}u, ^{KAD}u, $^{MD[n_2]}u$, ^{HD}u, $^{MD[n_2]}x^{Copy}$, $^{HD}x^{Copy}$, $^{HD}x^{dCopy}$, $^{HD}OtherStates^{Copy}$

Commented [JJ10]: The outputs of each module at time t (as calculated by their respective CalcOutput() routines) are stored as MiscVars in KiteFASTMBD.

Commented [JJ11]: The inputs from MBDyn and inputs to MoorDyn and HydroDyn at time t and the extrapolated inputs to KiteAeroDyn at t+KAD^dt are stored as MiscVars in KiteFASTMBD.

Commented [JJ12]: The temporary states of MoorDyn and HydroDyn are stored as MiscVars. In KiteFASTMBD.

Mapping of Outputs to Inputs in KiteFASTMBD

Output depends on Input (Y/N)		Inputs					
		MBDyn	KiteAeroDyn	InflowWind	MoorDyn	HydroDyn	Controller
Outputs	MBDyn		N	N	N	Y	Y
	KiteAeroDyn	Y					Y
	InflowWind		Y				Y
	MoorDyn	Y					Y
	HydroDyn	Y					
	Controller	N	N				

Commented [JJ13]: This may technically not be true, but we can only call the Controller once anyway, so, we'll assume no.

Data Flow (stopping when reaching "N")

MBDyn	HydroDyn Controller	MBDyn…		
KiteAeroDyn	MBDyn	HydroDyn Controller	MBDyn…	
	Controller			
InflowWind	KiteAeroDyn	MBDyn	HydroDyn Controller	MBDyn…

		Controller	
	Controller		
MoorDyn	MBDyn	HydroDyn	MBDyn…
		Controller	
	Controller		
HydroDyn	MBDyn	HydroDyn…	
		Controller	
Controller			

Thus, no nonlinear solves are required except between MBDyn and HyroDyn. But instead of doing the nonlinear solve, we'll use the predictor-corrector solve built into MBDyn directly.
Order of calls: MBDyn, Controller, MoorDyn, HydroDyn, InflowWind, KiteAeroDyn

Constructor

This routine initializes KiteFASTMBD at $t = 0$:

- Sets parameters
- Initializes states
- Calls module Init routines
- Opens the write output file
- Opens and writes the summary file

Commented [JJ14]: t=0 outputs are not set here, except for the Controller

Query the MBDyn model to access the inputs at $t = 0$.
Query the MBDyn model to access the names of the KiteAeroDyn, InflowWind, and MoorDyn primary input files

Commented [JJ15]: The names of the KiteAeroDyn input file etc., along with switches for enabling/disabling each module, must be queried from the MBDyn model. I haven't specifically included logic below to enable/disable modules, but this should implemented.

Set the parameters from inputs (Δt, $InterpOrder$, N_{Flaps}, N_{Pylons}, ${}^{MBD}\vec{g}$, ${}^{MBD}m^{SPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}I_{Rot}^{SPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}I_{Tran}^{SPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}x_{CM}^{SPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}m^{PPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}I_{Rot}^{PPyRtr}\left[n_{Pylons}, n_2\right]$, ${}^{MBD}I_{Tran}^{PPyRtr}\left[n_{Pylons}, n_2\right]$, and ${}^{MBD}x_{CM}^{PPyRtr}\left[n_{Pylons}, n_2\right]$). Trigger a fatal error if ${}^{MBD}m^{SPyRtr}\left[n_{Pylons}, n_2\right] < 0$, ${}^{MBD}I_{Rot}^{SPyRtr}\left[n_{Pylons}, n_2\right] < 0$, ${}^{MBD}I_{Tran}^{SPyRtr}\left[n_{Pylons}, n_2\right] - {}^{MBD}m^{SPyRtr}\left[n_{Pylons}, n_2\right]\left({}^{MBD}x_{CM}^{SPyRtr}\left[n_{Pylons}, n_2\right]\right)^2 < 0$, ${}^{MBD}m^{PPyRtr}\left[n_{Pylons}, n_2\right] < 0$, ${}^{MBD}I_{Rot}^{PPyRtr}\left[n_{Pylons}, n_2\right] < 0$, or ${}^{MBD}I_{Tran}^{PPyRtr}\left[n_{Pylons}, n_2\right] - {}^{MBD}m^{PPyRtr}\left[n_{Pylons}, n_2\right]\left({}^{MBD}x_{CM}^{PPyRtr}\left[n_{Pylons}, n_2\right]\right)^2 < 0$. Note that:

Commented [JJ16]: These must be queried from the MBDyn model.

- The flap indices: $n_{Flaps} = \left\{1, 2, \ldots, N_{Flaps}\right\}$
- The pylon indices: $n_{Pylons} = \left\{1, 2, \ldots, N_{Pylons}\right\}$
- And: $n_2 = \left\{1, 2\right\}$

Set the DCM conversion parameter from the FAST ground system (X pointed nominally downwind; Z pointed vertically opposite gravity; Y transverse) to the ground system used by the controller (X pointed nominally upwind; Z pointed vertically downward, Y transverse):

$$\Lambda^{FAST2Ctrl} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Set the reference positions (origins) needed as initialization inputs to KiteAeroDyn:

$$^{KAD}\vec{p}^{SWnOR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SWnO} - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{PWnOR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PWnO} - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{VSOR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{VSO} - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{SHSOR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SHSO} - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{PHSOR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PHSO} - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{SPyOR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SPyO}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{PPyOR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PPyO}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{SPyRtrR}\left[n_{Pylons}, n_2\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SPyRtr}\left[n_{Pylons}, n_2\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$
$$^{KAD}\vec{p}^{PPyRtrR}\left[n_{Pylons}, n_2\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PPyRtr}\left[n_{Pylons}, n_2\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$

Call KiteAeroDyn_Init()

Calculate the number of KiteAeroDyn time steps per MBDyn time step: $N_{KAD/MBD} = NINT\left(\frac{^{KAD}\Delta t}{\Delta t}\right)$

Trigger a fatal error if the KiteAeroDyn time step is not an integer multiple of the MBDyn time step i.e. if $N_{KAD/MBD}\Delta t - {}^{KAD}\Delta t \neq 0$

$^{KAD}NewTime = TRUE$

Set the air density for future reference: $\rho = {}^{KAD}\rho$

Determine the number of points where wind will be accessed within InflowWind by summing up the nodes on the AeroDyn input meshes, plus one for the fuselage origin and one for the base station:

$$\begin{aligned} ^{IfW}NumWindPoints = 2 \\ &+ {}^{KAD}NumFusNds \\ &+ {}^{KAD}NumSWnNds \\ &+ {}^{KAD}NumPWnNds \\ &+ {}^{KAD}NumVSNds \\ &+ {}^{KAD}NumSHSNds \\ &+ {}^{KAD}NumPHSNds \\ &+ {}^{KAD}NumPylNds\left(2N_{Pylons}\right) \\ &+ 4N_{Pylons} \end{aligned}$$

Call InflowWind_Init()

Set the initialization inputs to HydroDyn:

$$^{HD}Gravity = \left\|{}^{MBD}\vec{g}\right\|_2$$
$$^{HD}UseInputFile = TRUE$$

$^{HD}TMax =$

$^{HD}hasIce = 0$

$^{HD}PtfmLocationX = 0$

$^{HD}PtfmLocationY = 0$

Call HydroDyn_Init()

Trigger a fatal error if $\left({}^{HD}\Delta t \neq \Delta t \right)$

Set the initialization inputs to MoorDyn for the tether:

$^{MD[1]}g = \left\| {}^{MBD}\vec{g} \right\|_2$

$^{MD[1]}rhoW = \rho$

$^{MD[1]}WtrDepth = 0$

$$^{MD[1]}PtfmInit(1) = \begin{Bmatrix} {}^{MBD}\vec{p}^{FusO} \\ {}^{MBD}\Lambda^{FusO} \end{Bmatrix}$$

$$^{MD[1]}PtfmInit(2) = \begin{Bmatrix} {}^{MBD}\vec{p}^{Ptfm} \\ {}^{MBD}\Lambda^{Ptfm} \end{Bmatrix}$$

Call MoorDyn_Init()

Trigger a fatal error if $\left({}^{MD[1]}\Delta t \neq \Delta t \right)$

Set the initialization inputs to MoorDyn for the mooring system:

$^{MD[2]}g = \left\| {}^{MBD}\vec{g} \right\|_2$

$^{MD[2]}rhoW = {}^{HD}WtrDens$ (from HydroDyn initialization output)

$^{MD[2]}WtrDepth = {}^{HD}WtrDpth$ (from HydroDyn initialization output)

$$^{MD[2]}PtfmInit = \begin{Bmatrix} {}^{MBD}\vec{p}^{Ptfm} \\ {}^{MBD}\Lambda^{Ptfm} \end{Bmatrix}$$

Call MoorDyn_Init()

Trigger a fatal error if $\left({}^{MD[2]}\Delta t \neq \Delta t \right)$

Call Controller_Init()

Calculate the number of controller time steps per MBDyn time step: $N_{Ctrl/MBD} = NINT\left(\frac{{}^{Ctrl}\Delta t}{\Delta t} \right)$

Trigger a fatal error if the controller time step is not an integer multiple of the MBDyn time step i.e. if $N_{Ctrl/MBD}\Delta t - {}^{Ctrl}\Delta t \neq 0$

$^{Ctrl}NewTime = FALSE$

Set the undisplaced reference position parameter of the BS reference point:

Commented [JJ17]: Hopefully this can be accessed from the MBDyn input file?

Commented [JJ18R17]: TMax is passed from MBDyn to KiteFASTMBD at initialization.

Commented [JJ19]: We need to make a change to MoorDyn to allow for two separate bodies (or generalized for N bodies; N=2 for the tether). Each body will have its own set of fairleads (VESSEL nodes) and its own input and output point meshes. The number of bodies should be set at initialization based on making PtfmInit array of size N. In the MoorDyn input file, which fairleads correspond to which body can be distinguished by specifying VESSEL1 or VESSEL2 (or VESSELN) in place of VESSEL. For the tether, KiteFASTMBD assumes that VESSEL1 is the energy kite and VESSEL2 is the platform.

Commented [JJ20]: This PtfmInit is not an array of size 2, so, there is only one body (the floating platform) for the mooring system.

Commented [JJ21]: Note: the Controller_Init() call initializes the controller states and returns the initial controller outputs.

Commented [JJ22]: Note: the controller will trigger a fatal error if $N_{Flaps} \neq 3$ (to match the current controller interface), $N_{Pylons} \neq 2$ (to match the current controller interface)

Commented [JJ23]: If the controller takes larger steps than MBDyn, then we'll need to smooth the controller output to ensure that it is continuous (at least for the rotor velocity and acceleration). That is, the controller would have to be implemented like KiteAeroDyn.

$$\vec{p}^{BSRefR} = {}^{MBD}\Lambda^{Ptfm}\left\{{}^{MBD}\vec{p}^{BSRef} - {}^{MBD}\vec{p}^{Ptfm}\right\}$$

Set the reference positions and orientations of the line2 and point meshes from the inputs:

$${}^{MBD}\vec{p}^{PtfmR} = \vec{0}$$

$${}^{MBD}\Lambda^{PtfmR} = I$$

$${}^{MBD}\vec{p}_j^{FusR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{Fus} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumFusNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{FusR} = {}^{MBD}\Lambda_j^{Fus}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumFusNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{SWnR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{SWn} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumSWnNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{SWnR} = {}^{MBD}\Lambda_j^{SWn}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumSWnNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{PWnR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{PWn} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPWnNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{PWnR} = {}^{MBD}\Lambda_j^{PWn}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPWnNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{VSR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{VS} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumVSNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{VSR} = {}^{MBD}\Lambda_j^{VS}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumVSNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{SHSR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{SHS} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumSHSNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{SHSR} = {}^{MBD}\Lambda_j^{SHS}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumSHSNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{PHSR} = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{PHS} - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPHSNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{PHSR} = {}^{MBD}\Lambda_j^{PHS}\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPHSNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{SPy}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{SPyR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda_j^{SPy}\left[n_{Pylons}\right]\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$${}^{MBD}\vec{p}_j^{PPyR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}_j^{PPy}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}^{FusO}\right\} \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$${}^{MBD}\Lambda_j^{PPyR}\left[n_{Pylons}\right] = {}^{MBD}\Lambda_j^{PPy}\left[n_{Pylons}\right]\left[{}^{MBD}\Lambda^{FusO}\right]^T \qquad \text{(for } j=\left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$${}^{MBD}\vec{p}^{SPyRtrR}\left[n_{Pylons},n_2\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SPyRtr}\left[n_{Pylons},n_2\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$

$${}^{MBD}\Lambda^{SPyRtrR}\left[n_{Pylons},n_2\right] = I$$

$${}^{MBD}\vec{p}^{PPyRtrR}\left[n_{Pylons},n_2\right] = {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PPyRtr}\left[n_{Pylons},n_2\right] - {}^{MBD}\vec{p}^{FusO}\right\}$$

$${}^{MBD}\Lambda^{PPyRtrR}\left[n_{Pylons},n_2\right] = I$$

Set mesh-mappings between KiteFASTMBD-KiteAeroDyn, KiteFASTMBD-HydroDyn, KiteFASTMBD-MoorDyn for the tether-wing connection, KiteFASTMBD-MoorDyn for the tether-platform connection, and KiteFASTMBD-MoorDyn for the mooring system.

Open the write Output File

Open and write a summary file (if SumPrint = TRUE)

Commented [JJ24]: Note: the motion meshes are line2 meshes (except for the rotors, which are point meshes), but the load meshes are point meshes.

Commented [JJ25]: The mesh-mapping routines can only handle one source and one destination mesh. To do this mapping, the MBDyn meshes for the starboard and port wings (SWn and PWn) have to be copied into a single mesh using a one-to-one transfer of reference positions, reference orientations, and fields (which I label as Wn in the mesh-mappings below).

Commented [JJ26]: SumPrint must be queried from the MBDyn model

Commented [JJ27]: I'm only hand waving here because the implementation should be obvious (similar to other OpenFAST summary files)

KiteFASTMBD Summary File

Predictions were generated on DATE at TIME using KiteFASTMBD (VERSION, DATE)
compiled with
NWTC Subroutine Library (VERSION, DATE)
KiteAeroDyn (VERSION, DATE)
InflowWind (VERSION, DATE) for OpenFAST (VERSION DATE)
MoorDyn (VERSION, DATE)
HydroDyn (VERSION, DATE)
Controller Wrapper (VERSION, DATE)
Controller (VERSION, DATE)
MBDyn (VERSION, DATE)

Commented [JJ28]: (VERSION,DATE) has been replaced with the a git hash

Description from the MDyn input file: TITLE

Commented [JJ29]: Probably not needed if TITLE is not easily accessible within the MBDyn user element.

Time Step:

Component	Time Step
(-)	(s)
MBDyn	Δt
KiteAeroDyn	${}^{KAD}\Delta t$
MoorDyn	Δt
HydroDyn	Δt
Controller	${}^{Ctrl}\Delta t$

Reference Points, MBDyn Finite-Element Nodes, and MBDyn Gauss Points

Component	Type	Number	Output Number	x	y	z
(-)	(-)	(-)	(-)	(m)	(m)	(m)
Platform	Reference point	-	-	0	0	0
BS Reference	Reference point	-	-	$\vec{p}^{BSRefR}$		
Fuselage	Reference point	-	-	0	0	0
Fuselage	Finite-element node	j	$\begin{cases} Fus\langle\beta\rangle & for\left(FusOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_j^{FusR}$		
Fuselage	Gauss point	j	$\begin{cases} Fus\langle\beta\rangle & for\left(FusOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{FusR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{FusR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{FusR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{FusR} & otherwise \end{cases}$		
Starboard wing	Reference point	-	-	${}^{KAD}\vec{p}^{SWnOR}$		

Starboard wing	Finite-element node	j	$\begin{cases} SWn\langle\beta\rangle & for\left(SWnOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_{j}^{\,SWnR}$
Starboard wing	Gauss point	j	$\begin{cases} SWn\langle\beta\rangle & for\left(SWnOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,SWnR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,SWnR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,SWnR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,SWnR} & otherwise \end{cases}$
Port wing	Reference point	-	-	${}^{KAD}\vec{p}^{\,PWnOR}$
Port wing	Finite-element node	j	$\begin{cases} PWn\langle\beta\rangle & for\left(PWnOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_{j}^{\,PWnR}$
Port wing	Gauss point	j	$\begin{cases} PWn\langle\beta\rangle & for\left(PWnOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,PWnR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,PWnR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,PWnR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,PWnR} & otherwise \end{cases}$
Vertical stabilizer	Reference point	-	-	${}^{KAD}\vec{p}^{\,VSOR}$
Vertical stabilizer	Finite-element node	j	$\begin{cases} VS\langle\beta\rangle & for\left(VSOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_{j}^{\,VSR}$
Vertical stabilizer	Gauss point	j	$\begin{cases} VS\langle\beta\rangle & for\left(VSOutNd\left[\beta\right]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,VSR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,VSR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{\,VSR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j}^{\,VSR} & otherwise \end{cases}$
Starboard horizontal stabilizer	Reference point	-	-	${}^{KAD}\vec{p}^{\,SHSOR}$

Starboard horizontal stabilizer	Finite-element node	j	$\begin{cases} SHS\langle\beta\rangle & for\left(SHSOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_j^{SHSR}$
Starboard horizontal stabilizer	Gauss point	j	$\begin{cases} SHS\langle\beta\rangle & for\left(SHSOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{SHSR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{SHSR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{SHSR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{SHSR} & otherwise \end{cases}$
Port horizontal stabilizer	Reference point	-	-	${}^{KAD}\vec{p}^{PHSOR}$
Port horizontal stabilizer	Finite-element node	j	$\begin{cases} PHS\langle\beta\rangle & for\left(PHSOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_j^{PHSR}$
Port horizontal stabilizer	Gauss point	j	$\begin{cases} PHS\langle\beta\rangle & for\left(PHSOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{PHSR}+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{PHSR} & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{PHSR}+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{PHSR} & otherwise \end{cases}$
Starboard pylon n_{Pylons}	Reference point	-	-	${}^{KAD}\vec{p}^{SPyOR}\left[n_{Pylons}\right]$
Starboard pylon n_{Pylons}	Finite-element node	j	$\begin{cases} SP\langle n_{Pylons}\rangle\langle\beta\rangle & for\left(PylOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	${}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right]$
Starboard pylon n_{Pylons}	Gauss point	j	$\begin{cases} SP\langle n_{Pylons}\rangle\langle\beta\rangle & for\left(PylOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$	$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{SPyR}\left[n_{Pylons}\right]+\left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right] & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_{j+1}^{SPyR}\left[n_{Pylons}\right]+\left(1-\frac{\sqrt{3}}{3}\right){}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right] & otherwise \end{cases}$
Port pylon n_{Pylons}	Reference point	-	-	${}^{KAD}\vec{p}^{PPyOR}\left[n_{Pylons}\right]$

Port pylon n_{Pylons} Finite-element node j $\begin{cases} PP\langle n_{Pylons}\rangle\langle\beta\rangle & for\left(PylOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$

$^{MBD}\vec{p}_j^{PPyR}\left[n_{Pylons}\right]$

Port pylon n_{Pylons} Gauss point j $\begin{cases} PP\langle n_{Pylons}\rangle\langle\beta\rangle & for\left(PylOutNd[\beta]=j\right) \\ - & otherwise \end{cases}$

$$\begin{cases} \left(1-\frac{\sqrt{3}}{3}\right)^{MBD}\vec{p}_{j+1}^{PPyR}\left[n_{Pylons}\right]+\left(\frac{\sqrt{3}}{3}\right)^{MBD}\vec{p}_{j}^{PPyR}\left[n_{Pylons}\right] & for\left(Mod\left(j,2\right)=1\right) \\ \left(\frac{\sqrt{3}}{3}\right)^{MBD}\vec{p}_{j+1}^{PPyR}\left[n_{Pylons}\right]+\left(1-\frac{\sqrt{3}}{3}\right)^{MBD}\vec{p}_{j}^{PPyR}\left[n_{Pylons}\right] & otherwise \end{cases}$$

Top rotor on starboard pylon n_{Pylons} Reference point - -

$^{KAD}\vec{p}^{SPyRtrR}\left[n_{Pylons},1\right]$

Bottom rotor on starboard pylon n_{Pylons} Reference point - -

$^{KAD}\vec{p}^{SPyRtrR}\left[n_{Pylons},2\right]$

Top rotor on port pylon n_{Pylons} Reference point - -

$^{KAD}\vec{p}^{PPyRtrR}\left[n_{Pylons},1\right]$

Bottom rotor on port pylon n_{Pylons} Reference point - -

$^{KAD}\vec{p}^{PPyRtrR}\left[n_{Pylons},2\right]$

Requested Channels in KiteFASTMBD Output Files: NUMBER

Number	Name	Units	Generated by
0	Time	(s)	KiteFASTMBD
NUMBER	NAME	UNITS	(KiteFASTMBD, KiteAeroDyn, InflowWind, MoorDyn, HydroDyn, or Controller Wrapper)

Deconstructor

This routine ends KiteFASTMBD:

- Calls module End routines
- Deallocates memory
- Closes the write output file

AssRes

This routine accesses inputs at t (from GetXCur) (including $t=0$) for both the prediction and correction steps of each MBD time step, temporarily updates states from $t-\Delta t$ to t, and calculates outputs at t:

- Calls module UpdateStates and Controller_Step routines except at $t=0$
- Calls module CalcOutput routines

Set the discrete-time counter:

$$n=\frac{t}{\Delta t}-1$$

Query the MBDyn model to access the inputs at t (from GetXCur) i.e. ^{MBD}u.

Commented [JJ30]: AssRes could access inputs at t-dt (from GetXPrev), but we save the previous inputs as OtherStates instead.

Commented [JJ31]: Note: the module UpdateStates and Controller_Step routines are not called at t=0 (except for KiteAeroDyn) because the states have already been initialized through the Init calls.

Commented [JJ32]: This is necessary because in OpenFAST, UpdateStates shifts from t to t+dt whereas AssRes shifts from t-dt to t.

Calculate the translation displacements (relative) of the MBDyn input meshes at t:

$$^{MBD}\vec{u}^{Ptfm} = {}^{MBD}\vec{p}^{Ptfm} - {}^{MBD}\vec{p}^{PtfmR}$$

$$^{MBD}\vec{u}_j^{Fus} = {}^{MBD}\vec{p}_j^{Fus} - {}^{MBD}\vec{p}_j^{FusR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumFusNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{SWn} = {}^{MBD}\vec{p}_j^{SWn} - {}^{MBD}\vec{p}_j^{SWnR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumSWnNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{PWn} = {}^{MBD}\vec{p}_j^{PWn} - {}^{MBD}\vec{p}_j^{PWnR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumPWnNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{VS} = {}^{MBD}\vec{p}_j^{VS} - {}^{MBD}\vec{p}_j^{VSR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumVSNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{SHS} = {}^{MBD}\vec{p}_j^{SHS} - {}^{MBD}\vec{p}_j^{SHSR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumSHSNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{PHS} = {}^{MBD}\vec{p}_j^{PHS} - {}^{MBD}\vec{p}_j^{PHSR} \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumPHSNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{SPy}\left[n_{Pylons}\right] = {}^{MBD}\vec{p}_j^{SPy}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}_j^{SPyR}\left[n_{Pylons}\right] \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$$^{MBD}\vec{u}_j^{PPy}\left[n_{Pylons}\right] = {}^{MBD}\vec{p}_j^{PPy}\left[n_{Pylons}\right] - {}^{MBD}\vec{p}_j^{PPyR}\left[n_{Pylons}\right] \quad \text{(for } j = \left\{1,2,\ldots,{}^{MBD}NumPylNds\right\}\text{)}$$

$$^{MBD}\vec{u}^{SPyRtr}\left[n_{Pylons},n_2\right] = {}^{MBD}\vec{p}^{SPyRtr}\left[n_{Pylons},n_2\right] - {}^{MBD}\vec{p}^{SPyRtrR}\left[n_{Pylons},n_2\right]$$

$$^{MBD}\vec{u}^{PPyRtr}\left[n_{Pylons},n_2\right] = {}^{MBD}\vec{p}^{PPyRtr}\left[n_{Pylons},n_2\right] - {}^{MBD}\vec{p}^{PPyRtrR}\left[n_{Pylons},n_2\right]$$

Advance the controller only once per controller time step, updating the states to, and obtaining the controller outputs at, t:

IF $\left({}^{Ctrl}NewTime\right)$ THEN

First, calculate the InflowWind outputs at the base station and fuselage using the most converged inputs from MBDyn (as data stored in ${}^{MBD}OtherStates$ from the previous step):

$$^{IfW}PositionXYZ\left(:,1\right) = {}^{MBD}\vec{p}^{Wind}$$

$$^{IfW}PositionXYZ\left(:,2\right) = {}^{MBD}\vec{p}^{FusO}$$

Call InflowWind_CalcOutput()

Set inputs to Controller using the most converged inputs from MBDyn and the outputs from KiteAeroDyn, InflowWind, and MoorDyn (as data stored in ${}^{MBD}OtherStates$, ${}^{KAD}y$, and ${}^{MD}y$ from the previous step):

$$^{Ctrl}dcm\_g2b = {}^{MBD}\Lambda^{FusO}\left[\Lambda^{FAST2Ctrl}\right]^T$$

$$^{Ctrl}pqr = {}^{MBD}\Lambda^{FusO}\,{}^{MBD}\vec{\omega}^{FusO}$$

$$^{Ctrl}acc\_norm = \left\|{}^{MBD}\vec{a}^{FusO}\right\|_2$$

$$^{Ctrl}Xg = \Lambda^{FAST2Ctrl}\left\{{}^{MBD}\vec{p}^{FusO} - \vec{p}^{BSRefR}\right\}$$

$$^{Ctrl}Vg = \Lambda^{FAST2Ctrl}\,{}^{MBD}\vec{v}^{FusO}$$

$$^{Ctrl}Vb = {}^{MBD}\Lambda^{FusO}\,{}^{MBD}\vec{v}^{FusO}$$

$$^{Ctrl}Ag = \Lambda^{FAST2Ctrl}\,{}^{MBD}\vec{a}^{FusO}$$

$$^{Ctrl}Ab = {}^{MBD}\Lambda^{FusO}\,{}^{MBD}\vec{a}^{FusO}$$

$$^{Ctrl}rho = \rho$$

$$^{Ctrl}apparent\_wind = \Lambda^{FAST2Ctrl}\left\{{}^{IfW}VelocityUVW\left(:,2\right) - {}^{MBD}\vec{v}^{FusO}\right\}$$

Commented [JJ33]: One can call InflowWind_CalcOutput() with fewer than ${}^{IfW}NumWindPoints$.

Commented [JJ34]: All filtered values (_f) are identical to the unfiltered values.

$$^{Ctrl}tether\_force\_b = {}^{MBD}\Lambda^{FusO}\left\{\sum_{i=1}^{NFairs(1)} {}^{MD[1]}PtFairleadLoad(1)\%Force(:,i)\right\}$$

Commented [JJ35]: We are approximating this input to the controller as the vector sum of the fairlead tensions.

$$^{Ctrl}wind\_g = \Lambda^{FAST2Ctrl}\left\{{}^{IfW}VelocityUVW(:,1) - {}^{MBD}\vec{v}^{Wind}\right\}$$

$$^{Ctrl}aero\_torque^{SPyRtr}\left[n_{Pylons},n_2\right] = \left\{{}^{MBD}\hat{x}^{SPyRtr}\left[n_{Pylons},n_2\right]\right\}^T {}^{KAD}\vec{M}^{SPyRtr}\left[n_{Pylons},n_2\right]$$

$$^{Ctrl}aero\_torque^{PPyRtr}\left[n_{Pylons},n_2\right] = \left\{{}^{MBD}\hat{x}^{PPyRtr}\left[n_{Pylons},n_2\right]\right\}^T {}^{KAD}\vec{M}^{PPyRtr}\left[n_{Pylons},n_2\right]$$

Commented [JJ36]: These were added to the original controller inputs so that the controller could calculate the rotor/drivetrain acceleration and resulting generator speed and torque.

We should also ensure that the controller is using the same rotor/drivetrain rotational inertia.

$$^{Ctrl}BuoyIMU\_Position = \Lambda^{FAST2Ctrl}\left\{{}^{MBD}\vec{p}^{PtfmIMU} - \vec{p}^{BSRefR}\right\}$$

$$^{Ctrl}BuoyIMU\_Velocity = \Lambda^{FAST2Ctrl}\,{}^{MBD}\vec{v}^{PtfmIMU}$$

$$^{Ctrl}BuoyIMU\_Acceleration = \Lambda^{FAST2Ctrl}\,{}^{MBD}\vec{a}^{PtfmIMU}$$

$$^{Ctrl}BuoyIMU\_AngularVelocity = \Lambda^{FAST2Ctrl}\,{}^{MBD}\vec{\omega}^{PtfmIMU}$$

$$^{Ctrl}BuoyIMU\_Attitude = \frac{180}{\pi}F^{EulerExtract}\left({}^{MBD}\Lambda^{PtfmIMU}\right)$$

$$^{Ctrl}Wave =$$

$$^{Ctrl}Current =$$

Commented [JJ37]: We need clarification from Ruth what the controller needs for these.

Commented [JJ38]: I'm not sure what variable names are used by the controller for these.

- Call Controller_Step()

Ensure that we only call the controller once per the controller time step:

$$^{Ctrl}NewTime = FALSE$$

END

Store a copy of the MoorDyn current states at $t - \Delta t$ for the tether:

$$^{MD[1]}x^{Copy} = {}^{MD[1]}x$$

Set inputs to MoorDyn at t from MBDyn for the tether:

$$^{MD[1]}PtFairleadDisplacement(1) = M_u^{L2P}\left({}^{MBD}\vec{u}_j^{Wn}, {}^{MBD}\Lambda_j^{Wn}\right)$$

Commented [JJ39]: See earlier comment about mesh mapping with Wn above.

$$^{MD[1]}PtFairleadDisplacement(2) = M_u^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\Lambda^{Ptfm}\right)$$

Advance MoorDyn for the tether:

IF $(t > 0)$ Call MoorDyn_UpdateStates()

Call MoorDyn_CalcOutput()

Commented [JJ40]: Input the time at t-dt in this call.

The input at t-dt comes from $^{MD[1]}OtherStates$

Store a copy of the MoorDyn current states at $t - \Delta t$ for the mooring system:

$$^{MD[2]}x^{Copy} = {}^{MD[2]}x$$

Set inputs to MoorDyn at t from MBDyn for the mooring system:

$$^{MD[2]}PtFairleadDisplacement = M_u^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\Lambda^{Ptfm}\right)$$

Advance MoorDyn for the mooring system:

IF $(t > 0)$ Call MoorDyn_UpdateStates()

Call MoorDyn_CalcOutput()

Commented [JJ41]: Input the time at t-dt in this call.

The input at t-dt comes from $^{MD[2]}OtherStates$

Store a copy of the HydroDyn current states at $t-\Delta t$:

$$^{HD}x^{Copy} = {}^{HD}x$$

$$^{HD}x^{dCopy} = {}^{HD}x^{d}$$

$$^{HD}OtherStates^{Copy} = {}^{HD}OtherStates$$

Set inputs to HydroDyn at t from MBDyn:

$$^{HD}Morison\%DistribMesh\%TranslationDisp(:,:) = M_{u}^{P2L}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Morison\%DistribMesh\%Orientation(:,:) = M_{\Lambda}^{P2L}\left({}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Morison\%DistribMesh\%TranslationVel(:,:) = M_{v}^{P2L}\left({}^{HD}Morison\%DistribMesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{v}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Morison\%DistribMesh\%RotationVel(:,:) = M_{\omega}^{P2L}\left({}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Morison\%DistribMesh\%TranslationAcc(:,:) = M_{a}^{P2L}\left({}^{HD}Morison\%DistribMesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}, {}^{MBD}\vec{a}^{Ptfm}, {}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

$$^{HD}Morison\%DistribMesh\%RotationAcc(:,:) = M_{\alpha}^{P2L}\left({}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%TranslationDisp(:,:) = M_{u}^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%Orientation(:,:) = M_{\Lambda}^{P2P}\left({}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%TranslationVel(:,:) = M_{v}^{P2P}\left({}^{HD}Morison\%LumpedMesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{v}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%RotationVel(:,:) = M_{\omega}^{P2P}\left({}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%TranslationAcc(:,:) = M_{a}^{P2P}\left({}^{HD}Morison\%LumpedMesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}, {}^{MBD}\vec{a}^{Ptfm}, {}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

$$^{HD}Morison\%LumpedMesh\%RotationAcc(:,:) = M_{\alpha}^{P2P}\left({}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

$$^{HD}Mesh\%TranslationDisp(:,:) = M_{u}^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Mesh\%Orientation(:,:) = M_{\Lambda}^{P2P}\left({}^{MBD}\Lambda^{Ptfm}\right)$$

$$^{HD}Mesh\%TranslationVel(:,:) = M_{v}^{P2P}\left({}^{HD}Mesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{v}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Mesh\%RotationVel(:,:) = M_{\omega}^{P2P}\left({}^{MBD}\vec{\omega}^{Ptfm}\right)$$

$$^{HD}Mesh\%TranslationAcc(:,:) = M_{a}^{P2P}\left({}^{HD}Mesh\%TranslationDisp(:,:), {}^{MBD}\vec{u}^{Ptfm}, {}^{MBD}\vec{\omega}^{Ptfm}, {}^{MBD}\vec{a}^{Ptfm}, {}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

$$^{HD}Mesh\%RotationAcc(:,:) = M_{\alpha}^{P2P}\left({}^{MBD}\vec{\alpha}^{Ptfm}\right)$$

Advance HydroDyn:

IF $(t > 0)$ Call HydoDyn_UpdateStates()

Call HydroDyn_CalcOutput()

Advance KiteAeroDyn only once per KiteAeroDyn time step, interpolate the KiteAeroDyn outputs otherwise.

IF $\left({}^{KAD}NewTime\right)$ THEN

Shift the KiteAeroDyn input history:

IF $(t > 0)$

IF $(InterpOrder == 1)$ THEN

$$^{KAD}u(2) = {}^{KAD}u(1)$$

Commented [JJ42]: Because the platform reference position and orientation of the MBD point mesh and the HydroDyn WAMIT mesh are the same, these could by equivalence instead of via mesh mapping.

Commented [JJ43]: Input the time at t-dt in this call.

The input at t-dt comes from $^{HD}OtherStates^{u}$

ELSEIF ! $\left(InterpOrder == 2\right)$

$${}^{KAD}u\left(3\right) = {}^{KAD}u\left(2\right)$$

$${}^{KAD}u\left(2\right) = {}^{KAD}u\left(1\right)$$

END IF

END IF

Set inputs to KiteAeroDyn—stored in ${}^{KAD}u\left(1\right)$—from Controller at t:

$${}^{KAD}Ctrl^{SFlp}\left[n_{Flaps}\right] = \begin{cases} {}^{Ctrl}kFlapA5 & for\left(n_{Flaps} = 1\right) \\ {}^{Ctrl}kFlapA7 & for\left(n_{Flaps} = 2\right) \\ {}^{Ctrl}kFlapA8 & for\left(n_{Flaps} = 3\right) \end{cases}$$

$${}^{KAD}Ctrl^{PFlp}\left[n_{Flaps}\right] = \begin{cases} {}^{Ctrl}kFlapA4 & for\left(n_{Flaps} = 1\right) \\ {}^{Ctrl}kFlapA2 & for\left(n_{Flaps} = 2\right) \\ {}^{Ctrl}kFlapA1 & for\left(n_{Flaps} = 3\right) \end{cases}$$

$${}^{KAD}Ctrl^{Rudr}\left[n_2\right] = {}^{Ctrl}kFlapA10$$

$${}^{KAD}Ctrl^{SElv}\left[n_2\right] = {}^{Ctrl}kFlapA9$$

$${}^{KAD}Ctrl^{PElv}\left[n_2\right] = {}^{Ctrl}kFlapA9$$

$${}^{KAD}\Omega^{SPyRtr}\left[n_{Pylons}, n_2\right] = {}^{Ctrl}\Omega^{SPyRtr}\left[n_{Pylons}, n_2\right]$$

$${}^{KAD}\Omega^{PPyRtr}\left[n_{Pylons}, n_2\right] = {}^{Ctrl}\Omega^{PPyRtr}\left[n_{Pylons}, n_2\right]$$

$${}^{KAD}\theta^{SPyRtr}\left[n_{Pylons}, n_2\right] = 0$$

$${}^{KAD}\theta^{PPyRtr}\left[n_{Pylons}, n_2\right] = 0$$

Commented [JJ44]: Different controller documentation use kFlapRud in place of kFlapA10

Commented [JJ45]: Different controller documentation use kFlapEle in place of kFlapA9

Commented [JJ46]: These were added to the original controller outputs so that the controller could calculate the rotor/drivetrain acceleration and resulting generator speed and torque.

Commented [JJ47]: The rotor-collective pitch angles are not currently commanded from the controller; assume zero for now.

Set inputs to KiteAeroDyn—stored in ${}^{KAD}u\left(1\right)$—from MBDyn at t based on mesh-mapping:

$${}^{KAD}\vec{u}^{FusO} = {}^{MBD}\vec{p}^{FusO}$$

$${}^{KAD}\vec{u}_j^{Fus} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{Fus}, {}^{MBD}\Lambda_j^{Fus}\right)$$

$${}^{KAD}\Lambda_j^{Fus} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{Fus}\right)$$

$${}^{KAD}\vec{v}_j^{Fus} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{Fus}, {}^{MBD}\vec{u}_j^{Fus}, {}^{MBD}\vec{v}_j^{Fus}, {}^{MBD}\vec{\omega}_j^{Fus}\right)$$

$${}^{KAD}\vec{u}_j^{SWn} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{SWn}, {}^{MBD}\Lambda_j^{SWn}\right)$$

$${}^{KAD}\Lambda_j^{SWn} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{SWn}\right)$$

$${}^{KAD}\vec{v}_j^{SWn} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{SWn}, {}^{MBD}\vec{u}_j^{SWn}, {}^{MBD}\vec{v}_j^{SWn}, {}^{MBD}\vec{\omega}_j^{SWn}\right)$$

$${}^{KAD}\vec{u}_j^{PWn} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{PWn}, {}^{MBD}\Lambda_j^{PWn}\right)$$

$${}^{KAD}\Lambda_j^{PWn} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{PWn}\right)$$

$${}^{KAD}\vec{v}_j^{PWn} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{PWn}, {}^{MBD}\vec{u}_j^{PWn}, {}^{MBD}\vec{v}_j^{PWn}, {}^{MBD}\vec{\omega}_j^{PWn}\right)$$

Commented [JJ48]: You could use P2P mappings here, but there is no point, because the reference (0,0,0) is the same in both KiteAeroDyn and MBDyn.

$$^{KAD}\vec{u}_j^{VS} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{VS}, {}^{MBD}\Lambda_j^{VS} \right)$$
$$^{KAD}\Lambda_j^{VS} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{VS} \right)$$
$$^{KAD}\vec{v}_j^{VS} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{VS}, {}^{MBD}\vec{u}_j^{VS}, {}^{MBD}\vec{v}_j^{VS}, {}^{MBD}\vec{\omega}_j^{VS} \right)$$
$$^{KAD}\vec{u}_j^{SHS} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{SHS}, {}^{MBD}\Lambda_j^{SHS} \right)$$
$$^{KAD}\Lambda_j^{SHS} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{SHS} \right)$$
$$^{KAD}\vec{v}_j^{SHS} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{SHS}, {}^{MBD}\vec{u}_j^{SHS}, {}^{MBD}\vec{v}_j^{SHS}, {}^{MBD}\vec{\omega}_j^{SHS} \right)$$
$$^{KAD}\vec{u}_j^{PHS} = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{PHS}, {}^{MBD}\Lambda_j^{PHS} \right)$$
$$^{KAD}\Lambda_j^{PHS} = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{PHS} \right)$$
$$^{KAD}\vec{v}_j^{PHS} = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{PHS}, {}^{MBD}\vec{u}_j^{PHS}, {}^{MBD}\vec{v}_j^{PHS}, {}^{MBD}\vec{\omega}_j^{PHS} \right)$$
$$^{KAD}\vec{u}_j^{SPy}\left[n_{Pylons}\right] = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{SPy}\left[n_{Pylons}\right], {}^{MBD}\Lambda_j^{SPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\Lambda_j^{SPy}\left[n_{Pylons}\right] = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{SPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\vec{v}_j^{SPy}\left[n_{Pylons}\right] = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{SPy}\left[n_{Pylons}\right], {}^{MBD}\vec{u}_j^{SPy}\left[n_{Pylons}\right], {}^{MBD}\vec{v}_j^{SPy}\left[n_{Pylons}\right], {}^{MBD}\vec{\omega}_j^{SPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\vec{u}_j^{PPy}\left[n_{Pylons}\right] = M_u^{L2L}\left({}^{MBD}\vec{u}_j^{PPy}\left[n_{Pylons}\right], {}^{MBD}\Lambda_j^{PPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\Lambda_j^{PPy}\left[n_{Pylons}\right] = M_\Lambda^{L2L}\left({}^{MBD}\Lambda_j^{PPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\vec{v}_j^{PPy}\left[n_{Pylons}\right] = M_v^{L2L}\left({}^{KAD}\vec{u}_j^{PPy}\left[n_{Pylons}\right], {}^{MBD}\vec{u}_j^{PPy}\left[n_{Pylons}\right], {}^{MBD}\vec{v}_j^{PPy}\left[n_{Pylons}\right], {}^{MBD}\vec{\omega}_j^{PPy}\left[n_{Pylons}\right]\right)$$
$$^{KAD}\vec{u}^{SPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\vec{u}^{SPyRtr}\left[n_{Pylons}, n_2\right]$$
$$^{KAD}\Lambda^{SPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\Lambda^{SPyRtr}\left[n_{Pylons}, n_2\right]$$
$$^{KAD}\vec{v}^{SPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\vec{v}^{SPyRtr}\left[n_{Pylons}, n_2\right]$$
$$^{KAD}\vec{u}^{PPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\vec{u}^{PPyRtr}\left[n_{Pylons}, n_2\right]$$
$$^{KAD}\Lambda^{PPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\Lambda^{PPyRtr}\left[n_{Pylons}, n_2\right]$$
$$^{KAD}\vec{v}^{PPyRtr}\left[n_{Pylons}, n_2\right] = {}^{MBD}\vec{v}^{PPyRtr}\left[n_{Pylons}, n_2\right]$$

Commented [JJ49]: You could use P2P mappings here, but there is no point, because the references are the same in both KiteAeroDyn and MBDyn.

Commented [JJ50]: You could use P2P mappings here, but there is no point, because the references are the same in both KiteAeroDyn and MBDyn.

Set inputs to InflowWind at t based on the KiteAeroDyn inputs—stored in $^{KAD}u\left(1\right)$:

$$^{IfW}PositionXYZ\left(:,1\right) = {}^{MBD}\vec{p}^{Wind}$$
$$^{IfW}PositionXYZ\left(:,2\right) = {}^{MBD}\vec{p}^{FusO}$$
$$^{IfW}PositionXYZ\left(:,j+2\right) = {}^{KAD\,In}\vec{p}_j^{FusR} + {}^{KAD}\vec{u}_j^{Fus}$$ (for $j=\left\{1,2,\ldots,{}^{KAD}NumFusNds\right\}$)

$$^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ + {}^{KAD}NumFusNds \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{SWnR} + {}^{KAD}\vec{u}_j^{SWn}$$ (for $j=\left\{1,2,\ldots,{}^{KAD}NumSWnNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,PWnR} + {}^{KAD}\vec{u}_j^{\,PWn}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPWnNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,VSR} + {}^{KAD}\vec{u}_j^{\,VS}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumVSNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,SHSR} + {}^{KAD}\vec{u}_j^{\,SHS}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumSHSNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,PHSR} + {}^{KAD}\vec{u}_j^{\,PHS}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPHSNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(n_{Pylons} - 1\right) \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,SPyR}\left[n_{Pylons}\right] + {}^{KAD}\vec{u}_j^{\,SPy}\left[n_{Pylons}\right]
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPylNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} \vdots, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(N_{Pylons}\right) \\ \quad + {}^{KAD}NumPylNds\left(n_{Pylons}-1\right) \end{pmatrix} = {}^{KAD\,In}\vec{p}_j^{\,PPyR}\left[n_{Pylons}\right] + {}^{KAD}\vec{u}_j^{\,PPy}\left[n_{Pylons}\right]
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPylNds\right\}$)

$$
{}^{IfW}PositionXYZ\begin{pmatrix} \vdots, n_2+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(2N_{Pylons}\right) \\ \quad + 2\left(n_{Pylons}-1\right) \end{pmatrix} = {}^{KAD}\vec{p}_j^{\,SPyRtrR}\left[n_{Pylons}, n_2\right] + {}^{KAD}\vec{u}^{\,SPyRtr}\left[n_{Pylons}, n_2\right]
$$

$$
{}^{IfW}PositionXYZ\begin{pmatrix} \vdots, n_2+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(2N_{Pylons}\right) \\ \quad + 2\left(N_{Pylons}\right) \\ \quad + 2\left(n_{Pylons}-1\right) \end{pmatrix} = {}^{KAD}\vec{p}_j^{\,PPyRtrR}\left[n_{Pylons}, n_2\right] + {}^{KAD}\vec{u}^{\,PPyRtr}\left[n_{Pylons}, n_2\right]
$$

Call InflowWind_CalcOutput()

Commented [JJ51]: Input the time at t in this call.

Set inputs to KiteAeroDyn—stored in ${}^{KAD}u\left(1\right)$—from InflowWind at t:

$$
{}^{KAD}\vec{V}_j^{\,Fus} = {}^{IfW}VelocityUVW\left(\vdots, j+2\right)
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumFusNds\right\}$)

$$ {}^{KAD}\vec{V}_j^{SWn} = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \end{pmatrix} $$ (for $j = \left\{1, 2, \ldots, {}^{KAD}NumSWnNds\right\}$)

$$ {}^{KAD}\vec{V}_j^{PWn} = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \end{pmatrix} $$ (for $j = \left\{1, 2, \ldots, {}^{KAD}NumPWnNds\right\}$)

$$ {}^{KAD}\vec{V}_j^{VS} = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \end{pmatrix} $$ (for $j = \left\{1, 2, \ldots, {}^{KAD}NumVSNds\right\}$)

$$ {}^{KAD}\vec{V}_j^{SHS} = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \end{pmatrix} $$ (for $j = \left\{1, 2, \ldots, {}^{KAD}NumSHSNds\right\}$)

$$ {}^{KAD}\vec{V}_j^{PHS} = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \end{pmatrix} $$ (for $j = \left\{1, 2, \ldots, {}^{KAD}NumPHSNds\right\}$)

$$
{}^{KAD}\vec{V}_j^{SPy}\left[n_{Pylons}\right] = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(n_{Pylons}-1\right) \end{pmatrix}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPylNds\right\}$)

$$
{}^{KAD}\vec{V}_j^{PPy}\left[n_{Pylons}\right] = {}^{IfW}VelocityUVW\begin{pmatrix} :, j+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(N_{Pylons}\right) \\ \quad + {}^{KAD}NumPylNds\left(n_{Pylons}-1\right) \end{pmatrix}
$$

(for $j = \left\{1, 2, \ldots, {}^{KAD}NumPylNds\right\}$)

$$
{}^{KAD}\vec{V}^{SPyRtr}\left[n_{Pylons}, n_2\right] = {}^{IfW}VelocityUVW\begin{pmatrix} :, n_2+2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(2N_{Pylons}\right) \\ \quad + 2\left(n_{Pylons}-1\right) \end{pmatrix}
$$

$$
{}^{KAD}\vec{V}^{PPyRtr}\left[n_{Pylons}, n_2\right] = {}^{IfW}VelocityUVW\left(\begin{array}{l} :, n_2 + 2 \\ \quad + {}^{KAD}NumFusNds \\ \quad + {}^{KAD}NumSWnNds \\ \quad + {}^{KAD}NumPWnNds \\ \quad + {}^{KAD}NumVSNds \\ \quad + {}^{KAD}NumSHSNds \\ \quad + {}^{KAD}NumPHSNds \\ \quad + {}^{KAD}NumPylNds\left(2N_{Pylons}\right) \\ \quad + 2\left(N_{Pylons}\right) \\ \quad + 2\left(n_{Pylons} - 1\right) \end{array}\right)
$$

Initialize the KiteAeroDyn input history at $t = 0$:

IF $\left(t == 0\right)$ THEN

IF $\left(InterpOrder == 1\right)$ THEN

$${}^{KAD}u\left(2\right) = {}^{KAD}u\left(1\right)$$

$${}^{KAD}t\left(2\right) = -{}^{KAD}\Delta t$$

$${}^{KAD}t\left(1\right) = 0$$

ELSEIF ! $\left(InterpOrder == 2\right)$

$${}^{KAD}u\left(3\right) = {}^{KAD}u\left(1\right)$$

$${}^{KAD}u\left(2\right) = {}^{KAD}u\left(1\right)$$

$${}^{KAD}t\left(3\right) = -2\,{}^{KAD}\Delta t$$

$${}^{KAD}t\left(2\right) = {}^{KAD}\Delta t$$

$${}^{KAD}t\left(1\right) = 0$$

END IF

END IF

Advance KiteAeroDyn to $t + {}^{KAD}\Delta t$:

Call KiteAeroDyn_Input_ExtrapInterp(${}^{KAD}u\left(:\right)$, ${}^{KAD}t\left(:\right)$, ${}^{KAD}u$, $t + {}^{KAD}\Delta t$)

Call KiteAeroDyn_UpdateStates()

Call KiteAeroDyn_CalcOutput()

Commented [JJ52]: Input the time at t in this call

Commented [JJ53]: Input the time at t+KAD^dt in ths call.

Shift the KiteAeroDyn output history:

IF $\left(t > 0\right)$ THEN

IF $\left(InterpOrder == 1\right)$ THEN

$${}^{KAD}y\left(2\right) = {}^{KAD}y\left(1\right)$$

$${}^{KAD}y\left(1\right) = {}^{KAD}y$$

$^{KAD}t(2) = {}^{KAD}t(1)$

$^{KAD}t(1) = t + {}^{KAD}\Delta t$

ELSEIF ! $\left(InterpOrder == 2 \right)$

$^{KAD}y(3) = {}^{KAD}y(2)$

$^{KAD}y(2) = {}^{KAD}y(1)$

$^{KAD}y(1) = {}^{KAD}y$

$^{KAD}t(3) = {}^{KAD}t(2)$

$^{KAD}t(2) = {}^{KAD}t(1)$

$^{KAD}t(1) = t + {}^{KAD}\Delta t$

END IF

ELSE ! $\left(t == 0 \right)$

IF $\left(InterpOrder == 1 \right)$ THEN

$^{KAD}y(2) = {}^{KAD}y$

$^{KAD}y(1) = {}^{KAD}y$

ELSEIF ! $\left(InterpOrder == 2 \right)$

$^{KAD}y(3) = {}^{KAD}y$

$^{KAD}y(2) = {}^{KAD}y$

$^{KAD}y(1) = {}^{KAD}y$

END IF

END IF

Ensure that we only call KiteAeroDyn once per KiteAeroDyn time step:

$^{KAD}NewTime = FALSE$

END

Call KiteAeroDyn_Output_ExtrapInterp($^{KAD}y(:)$, $^{KAD}t(:)$, ^{KAD}y , t)

Model the rotor/drivetrain dynamics, including the effects from the Controller and KiteAeroDyn, and calculate the reaction loads on the pylons for transfer to MBDyn at t :

Call Rotor($^{MBD}\Lambda^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{\omega}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{a}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{\alpha}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{Ctrl}\Omega^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{Ctrl}T^{GenSPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{KAD}\vec{F}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{KAD}\vec{M}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{g}$, $^{MBD}m^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}I_{Rot}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}I_{Tran}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}x_{CM}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{F}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{M}^{SPyRtr}\left[n_{Pylons}, n_2 \right]$)

Call Rotor($^{MBD}\Lambda^{PPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{\omega}^{PPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{a}^{PPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{MBD}\vec{\alpha}^{PPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{Ctrl}\Omega^{PPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{Ctrl}T^{GenPPyRtr}\left[n_{Pylons}, n_2 \right]$, $^{KAD}\vec{F}^{PPyRtr}\left[n_{Pylons}, n_2 \right]$,

$^{KAD}\vec{M}^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}\vec{g}$, $^{MBD}m^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}I_{Rot}^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}I_{Tran}^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}x_{CM}^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}\vec{F}^{PPyRtr}\left[n_{Pylons},n_2\right]$, $^{MBD}\vec{M}^{PPyRtr}\left[n_{Pylons},n_2\right]$)

where:

$$^{Ctrl}T^{GenSPyRtr}\left[n_{Pylons},n_2\right]=\begin{cases} ^{Ctrl}Motor\,7 & for\left(\left(n_{Pylons}=1\right).AND.\left(n_2=1\right)\right) \\ ^{Ctrl}Motor\,2 & for\left(\left(n_{Pylons}=1\right).AND.\left(n_2=2\right)\right) \\ ^{Ctrl}Motor\,8 & for\left(\left(n_{Pylons}=2\right).AND.\left(n_2=1\right)\right) \\ ^{Ctrl}Motor\,1 & for\left(\left(n_{Pylons}=2\right).AND.\left(n_2=2\right)\right) \end{cases}$$

$$^{Ctrl}T^{GenPPyRtr}\left[n_{Pylons},n_2\right]=\begin{cases} ^{Ctrl}Motor\,6 & for\left(\left(n_{Pylons}=1\right).AND.\left(n_2=1\right)\right) \\ ^{Ctrl}Motor\,3 & for\left(\left(n_{Pylons}=1\right).AND.\left(n_2=2\right)\right) \\ ^{Ctrl}Motor\,5 & for\left(\left(n_{Pylons}=2\right).AND.\left(n_2=1\right)\right) \\ ^{Ctrl}Motor\,4 & for\left(\left(n_{Pylons}=2\right).AND.\left(n_2=2\right)\right) \end{cases}$$

Transfer outputs from KiteAeroDyn to MBDyn at t:

$$^{MBD}\vec{F}_j^{Fus}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{Fus}\right)$$

$$^{MBD}\vec{M}_j^{Fus}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{Fus},{}^{KAD\,Out}\vec{u}_j^{Fus},{}^{KAD}\vec{F}_j^{Fus},{}^{KAD}\vec{M}_j^{Fus}\right)$$

$$^{MBD}\vec{F}_j^{SWn}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{SWn}\right)$$

$$^{MBD}\vec{M}_j^{SWn}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{SWn},{}^{KAD\,Out}\vec{u}_j^{SWn},{}^{KAD}\vec{F}_j^{SWn},{}^{KAD}\vec{M}_j^{SWn}\right)$$

$$^{MBD}\vec{F}_j^{PWn}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{PWn}\right)$$

$$^{MBD}\vec{M}_j^{PWn}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{PWn},{}^{KAD\,Out}\vec{u}_j^{PWn},{}^{KAD}\vec{F}_j^{PWn},{}^{KAD}\vec{M}_j^{PWn}\right)$$

$$^{MBD}\vec{F}_j^{VS}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{VS}\right)$$

$$^{MBD}\vec{M}_j^{VS}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{VS},{}^{KAD\,Out}\vec{u}_j^{VS},{}^{KAD}\vec{F}_j^{VS},{}^{KAD}\vec{M}_j^{VS}\right)$$

$$^{MBD}\vec{F}_j^{SHS}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{SHS}\right)$$

$$^{MBD}\vec{M}_j^{SHS}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{SHS},{}^{KAD\,Out}\vec{u}_j^{SHS},{}^{KAD}\vec{F}_j^{SHS},{}^{KAD}\vec{M}_j^{SHS}\right)$$

$$^{MBD}\vec{F}_j^{PHS}=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{PHS}\right)$$

$$^{MBD}\vec{M}_j^{PHS}=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{PHS},{}^{KAD\,Out}\vec{u}_j^{PHS},{}^{KAD}\vec{F}_j^{PHS},{}^{KAD}\vec{M}_j^{PHS}\right)$$

$$^{MBD}\vec{F}_j^{SPy}\left[n_{Pylons}\right]=M_F^{P2P}\left({}^{KAD}\vec{F}_j^{SPy}\left[n_{Pylons}\right]\right)$$

$$^{MBD}\vec{M}_j^{SPy}\left[n_{Pylons}\right]=M_M^{P2P}\left({}^{MBD}\vec{u}_j^{SPy}\left[n_{Pylons}\right],{}^{KAD\,Out}\vec{u}_j^{SPy}\left[n_{Pylons}\right],{}^{KAD}\vec{F}_j^{SPy}\left[n_{Pylons}\right],{}^{KAD}\vec{M}_j^{SPy}\left[n_{Pylons}\right]\right)$$

Commented [JJ54]: This math assumes the top node of the pylon is node 1 and that the pylons are numbered from inboard to outboard.

Commented [JJ55]: This math is now done in the C controller.

$$
{}^{MBD}\vec{F}_j^{PPy}\left[n_{Pylons}\right] = M_F^{P2P}\left({}^{KAD}\vec{F}_j^{PPy}\left[n_{Pylons}\right]\right)
$$

$$
{}^{MBD}\vec{M}_j^{PPy}\left[n_{Pylons}\right] = M_M^{P2P}\left({}^{MBD}\vec{u}_j^{PPy}\left[n_{Pylons}\right], {}^{KAD\,Out}\vec{u}_j^{PPy}\left[n_{Pylons}\right], {}^{KAD}\vec{F}_j^{PPy}\left[n_{Pylons}\right], {}^{KAD}\vec{M}_j^{PPy}\left[n_{Pylons}\right]\right)
$$

Transfer outputs from HydroDyn to MBDyn at t:

$$
{}^{MBD}\vec{F}^{Ptfm} = M_F^{P2P}\left({}^{HD}AllHdroOrigin\%Force(:,1)\right)
$$

$$
{}^{MBD}\vec{M}^{Ptfm} = M_M^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{HD}Mesh\%TranslationDisp(:,:), {}^{HD}AllHdroOrigin\%Force(:,1), {}^{HD}AllHdroOrigin\%Moment(:,1)\right)
$$

Transfer outputs from MoorDyn to MBDyn at t for the tether:

$$
{}^{MBD}\vec{F}_j^{SWn} = {}^{MBD}\vec{F}_j^{SWn} + M_F^{P2P}\left({}^{MD[1]}PtFairleadLoad(1)\right)
$$

$$
{}^{MBD}\vec{M}_j^{SWn} = {}^{MBD}\vec{M}_j^{SWn} + M_M^{P2P}\left({}^{MBD}\vec{u}_j^{SWn}, {}^{MD[1]}PtFairleadDisplacement(1), {}^{MD[1]}PtFairleadLoad(1), \vec{0}\right)
$$

$$
{}^{MBD}\vec{F}_j^{PWn} = {}^{MBD}\vec{F}_j^{PWn} + M_F^{P2P}\left({}^{MD[1]}PtFairleadLoad(1)\right)
$$

$$
{}^{MBD}\vec{M}_j^{PWn} = {}^{MBD}\vec{M}_j^{PWn} + M_M^{P2P}\left({}^{MBD}\vec{u}_j^{PWn}, {}^{MD[1]}PtFairleadDisplacement(1), {}^{MD[1]}PtFairleadLoad(1), \vec{0}\right)
$$

Commented [JJ56]: See earlier comment about mesh mapping with Wn above.

$$
{}^{MBD}\vec{F}^{Ptfm} = {}^{MBD}\vec{F}^{Ptfm} + M_F^{P2P}\left({}^{MD[1]}PtFairleadLoad(2)\right)
$$

$$
{}^{MBD}\vec{M}^{Ptfm} = {}^{MBD}\vec{F}^{Ptfm} + M_M^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MD[1]}PtFairleadDisplacement(2), {}^{MD[1]}PtFairleadLoad(2), \vec{0}\right)
$$

Transfer outputs from MoorDyn to MBDyn at t for the mooring system:

$$
{}^{MBD}\vec{F}^{Ptfm} = {}^{MBD}\vec{F}^{Ptfm} + M_F^{P2P}\left({}^{MD[2]}PtFairleadLoad\right)
$$

$$
{}^{MBD}\vec{M}^{Ptfm} = {}^{MBD}\vec{M}^{Ptfm} + M_M^{P2P}\left({}^{MBD}\vec{u}^{Ptfm}, {}^{MD[2]}PtFairleadDisplacement, {}^{MD[2]}PtFairleadLoad, \vec{0}\right)
$$

Private SUBROUTINES

Rotor (SUBROUTINE Rotor)

Implements the structural dynamics of a rotor/drivetrain analytically to calculate the reaction loads (forces and moments) applied on the nacelle, including the applied aerodynamic loads, rotor inertial loads, rotor gyroscopic loads, etc. The analytical formulation assumes that the rotor/drivetrain is a rigid body rotating about the local x-axis of the nacelle coordinate system and that the structure is axisymmetric about this axis (with no imbalances) such that the calculations do not depend on the azimuth angle of the rotor. That is, for a body-fixed (x,y,z) coordinate system in the rotor/drivetrain, it is assumed that:

$$
{}^{CM}y = {}^{CM}z = 0
$$

$$
I_{xy} = I_{yz} = I_{xz} = 0
$$

$$
I_{xx} = I^{Rot}
$$

$$
I_{yy} = I_{zz} = I^{Tran}
$$

Inputs	Outputs	States	Parameters
• Λ^{Nac} – Displaced rotation (absolute orientation) of the nacelle (-)	• $\vec{F}^{React}$ – reaction forces applied on the nacelle at the rotor reference point		

• $\vec{\omega}^{Nac}$ – Rotational velocity (absolute) of the nacelle (rad/s) • $\vec{a}^{Nac}$ – Translational acceleration (absolute) of the nacelle at the rotor reference point (m/s^2) • $\vec{\alpha}^{Nac}$ – Rotational acceleration (absolute) of the nacelle (rad/s^2) • Ω^{Rtr} – Rotor speed about the shaft axis (relative to the nacelle) (rad/s) • T^{Gen} – electrical generator torque applied to the rotor/drivetrain about the shaft axis (N·m) • $\vec{F}^{Aero}$ – aerodynamic forces applied on the rotor at the rotor reference point expressed in the global inertial-frame coordinate system (N) • $\vec{M}^{Aero}$ – aerodynamic moments applied on the rotor about the rotor reference point expressed in the global inertial-frame coordinate system (N·m) • $\vec{g}$ – gravity vector expressed in the global inertial-frame coordinate system (m/s^2) • m – rotor/drivetrain mass (kg) • I^{Rot} – rotor/drivetrain rotational inertia about the shaft axis (kg·m^2) • I^{Tran} – rotor/drivetrain transverse inertia about the rotor reference point (kg·m^2) • ${}^{CM}x$ – distance along the shaft from the rotor reference point to the	expressed in the global inertial-frame coordinate system (N) • $\vec{M}^{React}$ – reaction moments applied on the nacelle about the rotor reference point expressed in the global inertial-frame coordinate system (N·m)		

Commented [JJ57]: This is input in place of:

$\dot{\Omega}^{Rtr}$ – Rotor acceleration about the shaft axis (relative to the nacelle) (rad/s^2)

center of mass of the rotor/drivetrain (positive along positive x) (m)			

Compute the inputs relative to the rotor/drivetrain CM and expressed in the local nacelle coordinate system:

$$^{CM}\vec{r} = {}^{CM}x\hat{x}^{Nac}$$

$$^{CM}I^{Tran} = I^{Tran} - m\,{}^{CM}x^2$$

$$\begin{Bmatrix} ^{CM}F_x^{Aero} \\ ^{CM}F_y^{Aero} \\ ^{CM}F_z^{Aero} \end{Bmatrix} = \Lambda^{Nac}\vec{F}^{Aero}$$

$$\begin{Bmatrix} ^{CM}M_x^{Aero} \\ ^{CM}M_y^{Aero} \\ ^{CM}M_z^{Aero} \end{Bmatrix} = \Lambda^{Nac}\left\{\vec{M}^{Aero} - {}^{CM}\vec{r}\times\vec{F}^{Aero}\right\}$$

$$\begin{Bmatrix} g_x \\ g_y \\ g_z \end{Bmatrix} = \Lambda^{Nac}\vec{g}$$

$$\vec{\omega}^{Rtr} = \vec{\omega}^{Nac} + \Omega^{Rtr}\hat{x}^{Nac}$$

$$\vec{\alpha}^{Rtr} = \vec{\alpha}^{Nac}$$

Commented [JJ58]: The equation implemented neglects the rotor acceleration about the shaft axis. The correct equation should be:

$$\vec{\alpha}^{Rtr} = \vec{\alpha}^{Nac} + \dot{\Omega}^{Rtr}\hat{x}^{Nac}$$

, but the rotor acceleration about the shaft axis is not needed because the generator torque is input instead.

$$\begin{Bmatrix} \omega_x^{Rtr} \\ \omega_y^{Rtr} \\ \omega_z^{Rtr} \end{Bmatrix} = \Lambda^{Nac}\vec{\omega}^{Rtr}$$

$$\begin{Bmatrix} ^{CM}a_x^{Rtr} \\ ^{CM}a_y^{Rtr} \\ ^{CM}a_z^{Rtr} \end{Bmatrix} = \Lambda^{Nac}\left\{\vec{a}^{Nac} + \vec{\alpha}^{Rtr}\times{}^{CM}\vec{r} + \vec{\omega}^{Rtr}\times\left\{\vec{\omega}^{Rtr}\times{}^{CM}\vec{r}\right\}\right\}$$

$$\begin{Bmatrix} \alpha_x^{Rtr} \\ \alpha_y^{Rtr} \\ \alpha_z^{Rtr} \end{Bmatrix} = \Lambda^{Nac}\vec{\alpha}^{Rtr}$$

Compute the reaction loads applied to the rotor/drivetrain at the rotor/drivetrain CM and expressed in the local nacelle coordinate system:

$$\begin{Bmatrix} ^{CM}F_x^{React} \\ ^{CM}F_y^{React} \\ ^{CM}F_z^{React} \end{Bmatrix} = \begin{Bmatrix} -{}^{CM}F_x^{Aero} - mg_x + m\,{}^{CM}a_x^{Rtr} \\ -{}^{CM}F_y^{Aero} - mg_y + m\,{}^{CM}a_y^{Rtr} \\ -{}^{CM}F_z^{Aero} - mg_z + m\,{}^{CM}a_z^{Rtr} \end{Bmatrix}$$

$$\begin{Bmatrix} ^{CM}M_x^{React} \\ ^{CM}M_y^{React} \\ ^{CM}M_z^{React} \end{Bmatrix} = \begin{Bmatrix} T^{Gen} \\ -{}^{CM}M_y^{Aero} + I^{Rot}\alpha_y^{Rtr} + \left(I^{Rot} - {}^{CM}I^{Tran}\right)\omega_z^{Rtr}\omega_x^{Rtr} \\ -{}^{CM}M_z^{Aero} + I^{Rot}\alpha_z^{Rtr} - \left(I^{Rot} - {}^{CM}I^{Tran}\right)\omega_y^{Rtr}\omega_x^{Rtr} \end{Bmatrix}$$

Commented [JJ59]: The first equation should be:

$$^{CM}M_x^{React} = -{}^{CM}M_x^{Aero} + I^{Rot}\alpha_x^{Rtr}$$

But this equals the equation implemented because the generator torque is input instead of the rotor acceleration about the shaft axis.

Compute the reaction loads applied to the nacelle (this is equal, but opposite to the reaction loads applied to the rotor/drivetrain) at the rotor/drivetrain reference point and expressed in the global inertial frame coordinate system:

$$\vec{F}^{React} = -\left[\Lambda^{Nac}\right]^T \begin{Bmatrix} {}^{CM}F_x^{React} \\ {}^{CM}F_y^{React} \\ {}^{CM}F_z^{React} \end{Bmatrix}$$

$$\vec{M}^{React} = -\left[\Lambda^{Nac}\right]^T \begin{Bmatrix} {}^{CM}M_x^{React} \\ {}^{CM}M_y^{React} \\ {}^{CM}M_z^{React} \end{Bmatrix} + {}^{CM}\vec{r} \times \vec{F}^{React}$$

AfterPredict

This routine updates the actual states based on the temporary states at the successful completion of time step t (including $t = 0$). That said, time has already been updated to $t = t + \Delta t$ before this routine is called, so technically, this routine is first called at $t = \Delta t$.

IF $\left(MOD\left(NINT\left(\frac{t}{\Delta t}\right), N_{KAD/MBD} \right) == 0 \right)$ THEN

$\quad {}^{KAD}NewTime = TRUE$

END

IF $\left(MOD\left(NINT\left(\frac{t}{\Delta t}\right), N_{Ctrl/MBD} \right) == 0 \right)$ THEN

$\quad {}^{Ctrl}NewTime = TRUE$

$\quad {}^{MBD}OtherStates = {}^{MBD}u$

END

${}^{MD[1]}OtherStates = {}^{MD[1]}u$

${}^{MD[1]}x = {}^{MD[1]}x^{Copy}$

${}^{MD[2]}OtherStates = {}^{MD[2]}u$

${}^{MD[2]}x = {}^{MD[2]}x^{Copy}$

${}^{HD}OtherStates^{u} = {}^{HD}u$

${}^{HD}x = {}^{HD}x^{Copy}$

${}^{HD}x^{d} = {}^{HD}x^{dCopy}$

${}^{HD}OtherStates = {}^{HD}OtherStates^{Copy}$

Output

This routine is called at the successful completion of time step t (including $t = 0$) to write output data to a file.

Calculate the KiteFASTMBD write outputs and write them to the output file, together with the module-level write output data currently stored in MiscVars.

This is a list of all possible output parameters available within the KiteFASTMBD (not including the module-level outputs available from KiteAeroDyn, InflowWind, MoorDyn, HydroDyn, and the Controller). The names are grouped by meaning, but can be ordered in the OUTPUTS section of the KiteMBDyn Preprocessor input file as you see fit.

Fusβ refers to output β on the fuselage, where β is a one-digit number in the range [1,9] corresponding to the finite-element node for motions or Gauss point for loads identified by entry β in the ***FusOutNd*** list. Setting β > ***NFusOuts*** yields invalid output.

SWnβ and PWnβ refer to output β on the starboard and port wings, respectively, where β is a one-digit number in the range [1,9] corresponding to the finite-element node for motions or Gauss point for loads identified by entry β in the ***SWnOutNd*** and ***PWnOutNd*** lists, respectively. Setting β > ***NSWnOuts*** and ***NPWnOuts***, respectively, yields invalid output.

VSβ refers to output β on the vertical stabilizer, where β is a one-digit number in the range [1,9] corresponding to the finite-element node for motions or Gauss point for loads identified by entry β in the ***VSOutNd*** list. Setting β > ***NVSOuts*** yields invalid output.

SHSβ and PHSβ refer to output β on the starboard and port horizontal stabilizers, respectively, where β is a one-digit number in the range [1,9] corresponding to the finite-element node for motions or Gauss point for loads identified by entry β in the ***SHSOutNd*** and ***PHSOutNd*** lists, respectively. Setting β > ***NSHSOuts*** and ***NPHSOuts***, respectively, yields invalid output.

SPα and PPα refer to pylon α on the starboard and port wings, respectively, where α is a one-digit number in the range [1,9]. SPαβ and PPαβ refer to output β on pylon α on the starboard and port wings, respectively, where α is a one-digit number in the range [1,9] and β is a one-digit number in the range [1,9] corresponding to the finite-element node for motions or Gauss point for loads identified by entry β in the ***PylOutNd*** list. Setting α > ***NumPylons*** or setting β > ***NPylOuts*** yields invalid output. If ***NumPylons*** > 9, only the first 9 pylons can be output.

For the fuselage, wings, vertical stabilizer, horizontal stabilizers, and pylons, the local structural coordinate system is used for output, where n is normal to the chord pointed toward the suction surface, c is along the chord pointed toward the trailing edge, and the spanwise (s) axis is directed into the airfoil following the right-hand rule i.e. $s = n \times c$.

For the floating platform (buoy), the buoy coordinate system is used for output, where the local x, y, and z are aligned with the global inertial frame (X,Y,Z) coordinate system when the buoy is undisplaced, with X pointed in the nominal 0˚ wind direction, Z pointed up (opposite gravity), and Y pointed to the left when looking downwind along 0˚ wind (following the right-hand rule).

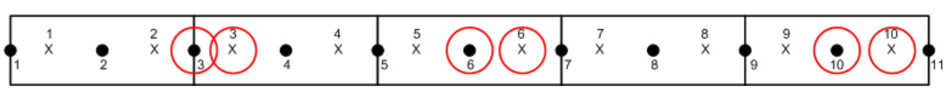


Figure: Example member with 5 finite elements, 11 nodes (•), and 10 Gauss points (x) (each finite element in MBDyn has 2 end nodes, 1 middle node, and 2 Gauss points). The red circles identify the finite-element nodes where motions are output and Gauss points where loads are output when ***NOuts*** = 3 and ***OutNd*** = 3, 6, 10.

Channel Name(s)	Unit(s)	Description
Fuselage		
FusβTDx, FusβTDy, FusβTDz, FusβRDx, FusβRDy, FusβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at Fusβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
FusβRVn, FusβRVc, FusβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at Fusβ expressed in the local structural coordinate

Commented [JJ60]: The new OUTPUT section of the KiteMBDyn Preprocessor input file should look something like this:

--- OUTPUT ---
True SumPrint Print summary data to <RootName>.sum? (flag)
"ES10.3E2" OutFmt Format used for text tabular output, excluding the time channel; resulting field should be 10 characters (string)
4 NFusOuts Number of fuselage outputs (-) [0 to 9]
2, 4, 6, 8 FusOutNd List of fuselage nodes/points whose values will be output (-) [1 to NFusOuts] [unused for NFusOuts=0]
4 NSWnOuts Number of starboard wing outputs (-) [0 to 9]
2, 4, 6, 8 SWnOutNd List of starboard wing nodes/points whose values will be output (-) [1 to NSWnOuts] [unused for NSWnOuts=0]
4 NPWnOuts Number of port wing outputs (-) [0 to 9]
2, 4, 6, 8 PWnOutNd List of port wing nodes/points whose values will be output (-) [1 to NPWnOuts] [unused for NPWnOuts=0]
2 NVSOuts Number of vertical stabilizer outputs (-) [0 to 9]
2, 4 VSOutNd List of vertical stabilizer nodes/points whose values will be output (-) [1 to NVSOuts] [unused for NVSOuts =0]
1 NSHSOuts Number of starboard horizontal stabilizer outputs (-) [0 to 9]
2 SHSOutNd List of starboard horizontal stabilizer nodes/points whose values will be output (-) [1 to NSHSOuts] [unused for NSHSOuts=0]
1 NPHSOuts Number of port horizontal stabilizer outputs (-) [0 to 9]
2 PHSOutNd List of port horizontal stabilizer nodes/points whose values will be output (-) [1 to NPHSOuts] [unused for NPHSOuts=0]
2 NPylOuts Number of pylon outputs (-) [0 to 9]
2, 4 PylOutNd List of pylon nodes/points whose values will be output (-) [1 to NPylOuts] [unused for NPylOuts=0]
OutList The next line(s) contains a list of output parameters. See OutListParameters.xlsx for a listing of available output channels (quoted string)
END of input file (the word "END" must appear in the first 3 columns of this last OutList line)

		system
FusβTAn, FusβTAc, FusβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at Fusβ expressed in the local structural coordinate system (does not include gravity)
FusβFRn, FusβFRc, FusβFRs, FusβMRn, FusβMRc, FusβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at Fusβ expressed in the local structural coordinate system
Starboard (Right) Wing		
SWnβTDx, SWnβTDy, SWnβTDz, SWnβRDx, SWnβRDy, SWnβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at SWnβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
SWnβRVn, SWnβRVc, SWnβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at SWnβ expressed in the local structural coordinate system
SWnβTAn, SWnβTAc, SWnβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at SWnβ expressed in the local structural coordinate system (does not include gravity)
SWnβFRn, SWnβFRc, SWnβFRs, SWnβMRn, SWnβMRc, SWnβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at SWnβ expressed in the local structural coordinate system
Port (Left) Wing		
PWnβTDx, PWnβTDy, PWnβTDz, PWnβRDx, PWnβRDy, PWnβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at PWnβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
PWnβRVn, PWnβRVc, PWnβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at PWnβ expressed in the local structural coordinate system
PWnβTAn, PWnβTAc, PWnβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at PWnβ expressed in the local structural coordinate system (does not include gravity)
PWnβFRn, PWnβFRc, PWnβFRs, PWnβMRn, PWnβMRc, PWnβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at PWnβ expressed in the local structural coordinate system
Vertical Stabilizer		
VSβTDx, VSβTDy, VSβTDz, VSβRDx, VSβRDy, VSβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at VSβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
VSβRVn, VSβRVc, VSβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at VSβ expressed in the local structural coordinate system
VSβTAn, VSβTAc, VSβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at VSβ expressed in the local structural coordinate system (does not include gravity)
VSβFRn, VSβFRc, VSβFRs, VSβMRn, VSβMRc, VSβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at VSβ expressed in the local structural coordinate system
Starboard (Right) Horizontal Stabilizer		
SHSβTDx, SHSβTDy, SHSβTDz, SHSβRDx, SHSβRDy, SHSβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at SHSβ relative to the undeflected rigid-body

		position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
SHSβRVn, SHSβRVc, SHSβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at SHSβ expressed in the local structural coordinate system
SHSβTAn, SHSβTAc, SHSβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at SHSβ expressed in the local structural coordinate system (does not include gravity)
SHSβFRn, SHSβFRc, SHSβFRs, SHSβMRn, SHSβMRc, SHSβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at SHSβ expressed in the local structural coordinate system
Port (Left) Horizontal Stabilizer		
PHSβTDx, PHSβTDy, PHSβTDz, PHSβRDx, PHSβRDy, PHSβRDz	(m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at PHSβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
PHSβRVn, PHSβRVc, PHSβRVs	(deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at PHSβ expressed in the local structural coordinate system
PHSβTAn, PHSβTAc, PHSβTAs	(m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at PHSβ expressed in the local structural coordinate system (does not include gravity)
PHSβFRn, PHSβFRc, PHSβFRs, PHSβMRn, PHSβMRc, PHSβMRs	(N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at PHSβ expressed in the local structural coordinate system
Pylons		
SPαβTDx, SPαβTDy, SPαβTDz, SPαβRDx, SPαβRDy, SPαβRDz, PPαβTDx, PPαβTDy, PPαβTDz, PPαβRDx, PPαβRDy, PPαβRDz	(m), (m), (m), (deg), (deg), (deg), (m), (m), (m), (deg), (deg), (deg)	Translational and rotational (angular) deflections at SPαβ and PPαβ relative to the undeflected rigid-body position/orientation in the kite coordinate system; the rotations are output as Euler angles in a x-y'-z'' (roll-pitch-yaw) rotation sequence
SPαβRVn, SPαβRVc, SPαβRVs, PPαβRVn, PPαβRVc, PPαβRVs	(deg/s), (deg/s), (deg/s), (deg/s), (deg/s), (deg/s)	Absolute rotational (angular) velocity at SPαβ and PPαβ expressed in the local structural coordinate system
SPαβTAn, SPαβTAc, SPαβTAs, PPαβTAn, PPαβTAc, PPαβTAs	(m/s^2), (m/s^2), (m/s^2), (m/s^2), (m/s^2), (m/s^2)	Absolute translational acceleration at SPαβ and PPαβ expressed in the local structural coordinate system (does not include gravity)
SPαβFRn, SPαβFRc, SPαβFRs, SPαβMRn, SPαβMRc, SPαβMRs, PPαβFRn, PPαβFRc, PPαβFRs, PPαβMRn, PPαβMRc, PPαβMRs	(N), (N), (N), (N·m), (N·m), (N·m), (N), (N), (N), (N·m), (N·m), (N·m)	Shear force and bending moment reaction loads at SPαβ and PPαβ expressed in the local structural coordinate system
Rotors		
SPαTRtSpd, SPαBRtSpd, PPαTRtSpd, PPαBRtSpd	(rad/s), (rad/s), (rad/s), (rad/s)	Rotor speed of the top (T) and bottom (B) rotor on SPα and PPα (relative to the nacelle)
SPαTRtAcc, SPαBRtAcc, PPαTRtAcc, PPαBRtAcc	(rad/s^2), (rad/s^2), (rad/s^2), (rad/s^2)	Rotor acceleration of the top (T) and bottom (B) rotor on SPα and PPα (relative to the nacelle)
Energy Kite		
KitePxi, KitePyi, KitePzi, KiteRoll, KitePitch, KiteYaw	(m), (m), (m), (deg), (deg), (deg)	Translational position and rotational (angular) orientation of the energy kite fuselage reference point in the global inertial-frame coordinate system; the rotations are output as Euler angles in a X-Y'-Z'' (roll-pitch-yaw) rotation sequence

KiteTVx, KiteTVy, KiteTVz, KiteRVx, KiteRVy, KiteRVz	(m/s), (m/s), (m/s), (deg/s), (deg/s), (deg/s)	Absolute translational and rotational (angular) velocity of the energy kite fuselage reference point expressed in the kite coordinate system
KiteTAx, KiteTAy, KiteTAz, KiteRAx, KiteRAy, KiteRAz	(m/s^2), (m/s^2), (m/s^2), (deg/s^2), (deg/s^2), (deg/s^2)	Absolute translational and rotational (angular) acceleration of the energy kite fuselage reference point expressed in the kite coordinate system
Floating Platform (Buoy)		
BuoySurge, BuoySway, BuoyHeave BuoyRoll, BuoyPitch, BuoyYaw	(m), (m), (m), (deg), (deg), (deg)	Translational position and rotational (angular) orientation of the buoy reference point in the global inertial-frame coordinate system; the rotations are output as Euler angles in a X-Y'-Z'' (roll-pitch-yaw) rotation sequence
BuoyTVx, BuoyTVy, BuoyTVz, BuoyRVx, BuoyRVy, BuoyRVz	(m/s), (m/s), (m/s), (deg/s), (deg/s), (deg/s)	Absolute translational and rotational (angular) velocity of the buoy reference point expressed in the buoy coordinate system
BuoyTAx, BuoyTAy, BuoyTAz	(m/s^2), (m/s^2), (m/s^2),	Absolute translational acceleration of the buoy reference point expressed in the buoy coordinate system
BIMUPxi, BIMUPyi, BIMUPzi BIMURoll, BIMUPitch, BIMUYaw	(m), (m), (m), (deg), (deg), (deg)	Translational position and rotational (angular) orientation of the buoy inertial measurement unit in the global inertial-frame coordinate system; the rotations are output as Euler angles in a X-Y'-Z'' (roll-pitch-yaw) rotation sequence
BIMUTVx, BIMUTVy, BIMUTVz, BIMURVx, BIMURVy, BIMURVz	(m/s), (m/s), (m/s), (deg/s), (deg/s), (deg/s)	Absolute translational and rotational (angular) velocity of the buoy inertial measurement unit expressed in the buoy coordinate system
BIMUTAx, BIMUTAy, BIMUTAz	(m/s^2), (m/s^2), (m/s^2),	Absolute translational acceleration of the buoy inertial measurement unit expressed in the buoy coordinate system
BBSRPxi, BBSRPyi, BBSRPzi BBSRRoll, BBSRPitch, BBSRYaw	(m), (m), (m), (deg), (deg), (deg)	Translational position and rotational (angular) orientation of the buoy BS reference point in the global inertial-frame coordinate system; the rotations are output as Euler angles in a X-Y'-Z'' (roll-pitch-yaw) rotation sequence
BBSRTVx, BBSRTVy, BBSRTVz, BBSRRVx, BBSRRVy, BBSRRVz	(m/s), (m/s), (m/s), (deg/s), (deg/s), (deg/s)	Absolute translational and rotational (angular) velocity of the buoy BS reference point expressed in the buoy coordinate system
BBSRTAx, BBSRTAy, BBSRTAz	(m/s^2), (m/s^2), (m/s^2),	Absolute translational acceleration of the buoy BS reference point expressed in the buoy coordinate system

These are calculated within KiteFASTMBD as follows:

Fuselage:

$$\begin{Bmatrix} Fus\beta TDx \\ Fus\beta TDy \\ Fus\beta TDz \\ Fus\beta RDx \\ Fus\beta RDy \\ Fus\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO} \left\{ {}^{MBD}\vec{p}^{Fus}_{FusOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{FusR}_{FusOutNd[\beta]} \\ \dfrac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{FusR}_{FusOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{Fus}_{FusOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} Fus\beta RVn \\ Fus\beta RVc \\ Fus\beta RVs \end{Bmatrix} = \frac{180}{\pi} \, {}^{MBD}\Lambda^{Fus}_{FusOutNd[\beta]} \, {}^{MBD}\vec{\omega}^{Fus}_{FusOutNd[\beta]}$$

$$\begin{Bmatrix} Fus\beta TAn \\ Fus\beta TAc \\ Fus\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{Fus}_{FusOutNd[\beta]} \, {}^{MBD}\vec{a}^{Fus}_{FusOutNd[\beta]}$$

$$\begin{Bmatrix} Fus\beta FRn \\ Fus\beta FRc \\ Fus\beta FRs \\ Fus\beta MRn \\ Fus\beta MRc \\ Fus\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{Fus}_{FusOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{Fus}_{FusOutNd[\beta]} \end{Bmatrix}$$

Starboard (Right) Wing:

$$\begin{Bmatrix} SWn\beta TDx \\ SWn\beta TDy \\ SWn\beta TDz \\ SWn\beta RDx \\ SWn\beta RDy \\ SWn\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO} \left\{ {}^{MBD}\vec{p}^{SWn}_{SWnOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{SWnR}_{SWnOutNd[\beta]} \\ \frac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{SWnR}_{SWnOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{SWn}_{SWnOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} SWn\beta RVn \\ SWn\beta RVc \\ SWn\beta RVs \end{Bmatrix} = \frac{180}{\pi} \, {}^{MBD}\Lambda^{SWn}_{SWnOutNd[\beta]} \, {}^{MBD}\vec{\omega}^{SWn}_{SWnOutNd[\beta]}$$

$$\begin{Bmatrix} SWn\beta TAn \\ SWn\beta TAc \\ SWn\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{SWn}_{SWnOutNd[\beta]} \, {}^{MBD}\vec{a}^{SWn}_{SWnOutNd[\beta]}$$

$$\begin{Bmatrix} SWn\beta FRn \\ SWn\beta FRc \\ SWn\beta FRs \\ SWn\beta MRn \\ SWn\beta MRc \\ SWn\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{SWn}_{SWnOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{SWn}_{SWnOutNd[\beta]} \end{Bmatrix}$$

Port (Left) Wing:

$$\begin{Bmatrix} PWn\beta TDx \\ PWn\beta TDy \\ PWn\beta TDz \\ PWn\beta RDx \\ PWn\beta RDy \\ PWn\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO}\left\{ {}^{MBD}\vec{p}^{PWn}_{PWnOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{PWnR}_{PWnOutNd[\beta]} \\ \frac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{PWnR}_{PWnOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{PWn}_{PWnOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} PWn\beta RVn \\ PWn\beta RVc \\ PWn\beta RVs \end{Bmatrix} = \frac{180}{\pi} {}^{MBD}\Lambda^{PWn}_{PWnOutNd[\beta]} \; {}^{MBD}\vec{\omega}^{PWn}_{PWnOutNd[\beta]}$$

$$\begin{Bmatrix} PWn\beta TAn \\ PWn\beta TAc \\ PWn\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{PWn}_{PWnOutNd[\beta]} \; {}^{MBD}\vec{a}^{PWn}_{PWnOutNd[\beta]}$$

$$\begin{Bmatrix} PWn\beta FRn \\ PWn\beta FRc \\ PWn\beta FRs \\ PWn\beta MRn \\ PWn\beta MRc \\ PWn\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{PWn}_{PWnOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{PWn}_{PWnOutNd[\beta]} \end{Bmatrix}$$

Vertical Stabilizer:

$$\begin{Bmatrix} VS\beta TDx \\ VS\beta TDy \\ VS\beta TDz \\ VS\beta RDx \\ VS\beta RDy \\ VS\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO}\left\{ {}^{MBD}\vec{p}^{VS}_{VSOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{VSR}_{VSOutNd[\beta]} \\ \frac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{VSR}_{VSOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{VS}_{VSOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} VS\beta RVn \\ VS\beta RVc \\ VS\beta RVs \end{Bmatrix} = \frac{180}{\pi} {}^{MBD}\Lambda^{VS}_{VSOutNd[\beta]} \; {}^{MBD}\vec{\omega}^{VS}_{VSOutNd[\beta]}$$

$$\begin{Bmatrix} VS\beta TAn \\ VS\beta TAc \\ VS\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{VS}_{VSOutNd[\beta]} \; {}^{MBD}\vec{a}^{VS}_{VSOutNd[\beta]}$$

$$\begin{Bmatrix} VS\beta FRn \\ VS\beta FRc \\ VS\beta FRs \\ VS\beta MRn \\ VS\beta MRc \\ VS\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{VS}_{VSOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{VS}_{VSOutNd[\beta]} \end{Bmatrix}$$

Starboard (Right) Horizontal Stabilizer:

$$\begin{Bmatrix} SHS\beta TDx \\ SHS\beta TDy \\ SHS\beta TDz \\ SHS\beta RDx \\ SHS\beta RDy \\ SHS\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO}\left\{ {}^{MBD}\vec{p}^{SHS}_{SHSOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{SHSR}_{SHSOutNd[\beta]} \\ \dfrac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{SHSR}_{SHSOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{SHS}_{SHSOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} SHS\beta RVn \\ SHS\beta RVc \\ SHS\beta RVs \end{Bmatrix} = \frac{180}{\pi} {}^{MBD}\Lambda^{SHS}_{SHSOutNd[\beta]} {}^{MBD}\vec{\omega}^{SHS}_{SHSOutNd[\beta]}$$

$$\begin{Bmatrix} SHS\beta TAn \\ SHS\beta TAc \\ SHS\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{SHS}_{SHSOutNd[\beta]} {}^{MBD}\vec{a}^{SHS}_{SHSOutNd[\beta]}$$

$$\begin{Bmatrix} SHS\beta FRn \\ SHS\beta FRc \\ SHS\beta FRs \\ SHS\beta MRn \\ SHS\beta MRc \\ SHS\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{SHS}_{SHSOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{SHS}_{SHSOutNd[\beta]} \end{Bmatrix}$$

Port (Left) Horizontal Stabilizer:

$$\begin{Bmatrix} PHS\beta TDx \\ PHS\beta TDy \\ PHS\beta TDz \\ PHS\beta RDx \\ PHS\beta RDy \\ PHS\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO}\left\{ {}^{MBD}\vec{p}^{PHS}_{PHSOutNd[\beta]} - {}^{MBD}\vec{p}^{FusO} \right\} - {}^{MBD}\vec{p}^{PHSR}_{PHSOutNd[\beta]} \\ \dfrac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO} \right]^T \left[{}^{MBD}\Lambda^{PHSR}_{PHSOutNd[\beta]} \right]^T {}^{MBD}\Lambda^{PHS}_{PHSOutNd[\beta]} \right) \end{Bmatrix}$$

$$\begin{Bmatrix} PHS\beta RVn \\ PHS\beta RVc \\ PHS\beta RVs \end{Bmatrix} = \frac{180}{\pi} {}^{MBD}\Lambda^{PHS}_{PHSOutNd[\beta]} {}^{MBD}\vec{\omega}^{PHS}_{PHSOutNd[\beta]}$$

$$\begin{Bmatrix} PHS\beta TAn \\ PHS\beta TAc \\ PHS\beta TAs \end{Bmatrix} = {}^{MBD}\Lambda^{PHS}_{PHSOutNd[\beta]}\ {}^{MBD}\vec{a}^{PHS}_{PHSOutNd[\beta]}$$

$$\begin{Bmatrix} PHS\beta FRn \\ PHS\beta FRc \\ PHS\beta FRs \\ PHS\beta MRn \\ PHS\beta MRc \\ PHS\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{PHS}_{PHSOutNd[\beta]} \\ {}^{MBD}\vec{M}R^{PHS}_{PHSOutNd[\beta]} \end{Bmatrix}$$

Pylons:

$$\begin{Bmatrix} SP\alpha\beta TDx \\ SP\alpha\beta TDy \\ SP\alpha\beta TDz \\ SP\alpha\beta RDx \\ SP\alpha\beta RDy \\ SP\alpha\beta RDz \\ PP\alpha\beta TDx \\ PP\alpha\beta TDy \\ PP\alpha\beta TDz \\ PP\alpha\beta RDx \\ PP\alpha\beta RDy \\ PP\alpha\beta RDz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{SPy}_{PylOutNd[\beta]}[\alpha] - {}^{MBD}\vec{p}^{FusO}\right\} - {}^{MBD}\vec{p}^{SPyR}_{PylOutNd[\beta]}[\alpha] \\ \frac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO}\right]^T \left[{}^{MBD}\Lambda^{SPyR}_{PylOutNd[\beta]}[\alpha]\right]^T {}^{MBD}\Lambda^{SPy}_{PylOutNd[\beta]}[\alpha]\right) \\ {}^{MBD}\Lambda^{FusO}\left\{{}^{MBD}\vec{p}^{PPy}_{PylOutNd[\beta]}[\alpha] - {}^{MBD}\vec{p}^{FusO}\right\} - {}^{MBD}\vec{p}^{PPyR}_{PylOutNd[\beta]}[\alpha] \\ \frac{180}{\pi} F^{EulerExtract}\left(\left[{}^{MBD}\Lambda^{FusO}\right]^T \left[{}^{MBD}\Lambda^{PPyR}_{PylOutNd[\beta]}[\alpha]\right]^T {}^{MBD}\Lambda^{PPy}_{PylOutNd[\beta]}[\alpha]\right) \end{Bmatrix}$$

$$\begin{Bmatrix} SP\alpha\beta RVn \\ SP\alpha\beta RVc \\ SP\alpha\beta RVs \\ PP\alpha\beta RVn \\ PP\alpha\beta RVc \\ PP\alpha\beta RVs \end{Bmatrix} = \begin{Bmatrix} \frac{180}{\pi}\ {}^{MBD}\Lambda^{SPy}_{PylOutNd[\beta]}[\alpha]\ {}^{MBD}\vec{\omega}^{SPy}_{PylOutNd[\beta]}[\alpha] \\ \frac{180}{\pi}\ {}^{MBD}\Lambda^{PPy}_{PylOutNd[\beta]}[\alpha]\ {}^{MBD}\vec{\omega}^{PPy}_{PylOutNd[\beta]}[\alpha] \end{Bmatrix}$$

$$\begin{Bmatrix} SP\alpha\beta TAn \\ SP\alpha\beta TAc \\ SP\alpha\beta TAs \\ PP\alpha\beta TAn \\ PP\alpha\beta TAc \\ PP\alpha\beta TAs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{SPy}_{PylOutNd[\beta]}[\alpha]\ {}^{MBD}\vec{a}^{SPy}_{PylOutNd[\beta]}[\alpha] \\ {}^{MBD}\Lambda^{PPy}_{PylOutNd[\beta]}[\alpha]\ {}^{MBD}\vec{a}^{PPy}_{PylOutNd[\beta]}[\alpha] \end{Bmatrix}$$

$$\begin{Bmatrix} SP\alpha\beta FRn \\ SP\alpha\beta FRc \\ SP\alpha\beta FRs \\ SP\alpha\beta MRn \\ SP\alpha\beta MRc \\ SP\alpha\beta MRs \\ PP\alpha\beta FRn \\ PP\alpha\beta FRc \\ PP\alpha\beta FRs \\ PP\alpha\beta MRn \\ PP\alpha\beta MRc \\ PP\alpha\beta MRs \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{F}R^{SPy}_{PylOutNd[\beta]}[\alpha] \\ {}^{MBD}\vec{M}R^{SPy}_{PylOutNd[\beta]}[\alpha] \\ {}^{MBD}\vec{F}R^{PPy}_{PylOutNd[\beta]}[\alpha] \\ {}^{MBD}\vec{M}R^{PPy}_{PylOutNd[\beta]}[\alpha] \end{Bmatrix}$$

Rotors

$$\begin{Bmatrix} SP\alpha TRtSpd \\ SP\alpha BRtSpd \\ PP\alpha TRtSpd \\ PP\alpha BRtSpd \end{Bmatrix} = \begin{Bmatrix} {}^{Ctrl}\Omega^{SPyRtr}[\alpha,1] \\ {}^{Ctrl}\Omega^{SPyRtr}[\alpha,2] \\ {}^{Ctrl}\Omega^{PPyRtr}[\alpha,1] \\ {}^{Ctrl}\Omega^{PPyRtr}[\alpha,2] \end{Bmatrix}$$

$$\begin{Bmatrix} SP\alpha TRtAcc \\ SP\alpha BRtAcc \\ PP\alpha TRtAcc \\ PP\alpha BRtAcc \end{Bmatrix} = \begin{Bmatrix} {}^{Ctrl}\alpha^{SPyRtr}[\alpha,1] \\ {}^{Ctrl}\alpha^{SPyRtr}[\alpha,2] \\ {}^{Ctrl}\alpha^{PPyRtr}[\alpha,1] \\ {}^{Ctrl}\alpha^{PPyRtr}[\alpha,2] \end{Bmatrix}$$

Energy Kite

$$\begin{Bmatrix} KitePxi \\ KitePyi \\ KitePzi \\ KiteRoll \\ KitePitch \\ KiteYaw \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{p}^{FusO} \\ \dfrac{180}{\pi} F^{EulerExtract}\left({}^{MBD}\Lambda^{FusO}\right) \end{Bmatrix}$$

$$\begin{Bmatrix} KiteTVx \\ KiteTVy \\ KiteTVz \\ KiteRVx \\ KiteRVy \\ KiteRVz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO\,MBD}\vec{v}^{FusO} \\ \dfrac{180}{\pi}\,{}^{MBD}\Lambda^{FusO\,MBD}\vec{\omega}^{FusO} \end{Bmatrix}$$

$$\begin{Bmatrix} KiteTAx \\ KiteTAy \\ KiteTAz \\ KiteRAx \\ KiteRAy \\ KiteRAz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{FusO\,MBD}\vec{a}^{FusO} \\ \dfrac{180}{\pi}\,{}^{MBD}\Lambda^{FusO\,MBD}\vec{\alpha}^{FusO} \end{Bmatrix}$$

Floating Platform (Buoy)

$$\begin{Bmatrix} BuoySurge \\ BuoySway \\ BuoyHeave \\ BuoyRoll \\ BuoyPitch \\ BuoyYaw \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{p}^{Ptfm} \\ \dfrac{180}{\pi} F^{EulerExtract}\left({}^{MBD}\Lambda^{Ptfm}\right) \end{Bmatrix}$$

$$\begin{Bmatrix} BuoyTVx \\ BuoyTVy \\ BuoyTVz \\ BuoyRVx \\ BuoyRVy \\ BuoyRVz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{Ptfm\,MBD}\vec{v}^{Ptfm} \\ \dfrac{180}{\pi}\,{}^{MBD}\Lambda^{Ptfm\,MBD}\vec{\omega}^{Ptfm} \end{Bmatrix}$$

$$\begin{Bmatrix} BuoyTAx \\ BuoyTAy \\ BuoyTAz \end{Bmatrix} = {}^{MBD}\Lambda^{Ptfm\,MBD}\vec{a}^{Ptfm}$$

$$\begin{Bmatrix} BIMUPxi \\ BIMUPyi \\ BIMUPzi \\ BIMURoll \\ BIMUPitch \\ BIMUYaw \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{p}^{PtfmIMU} \\ \dfrac{180}{\pi} F^{EulerExtract}\left({}^{MBD}\Lambda^{PtfmIMU}\right) \end{Bmatrix}$$

$$\begin{Bmatrix} BIMUTVx \\ BIMUTVy \\ BIMUTVz \\ BIMURVx \\ BIMURVy \\ BIMURVz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{PtfmIMU\,MBD}\vec{v}^{PtfmIMU} \\ \dfrac{180}{\pi}\,{}^{MBD}\Lambda^{PtfmIMU\,MBD}\vec{\omega}^{PtfmIMU} \end{Bmatrix}$$

$$\begin{Bmatrix} BIMUTAx \\ BIMUTAy \\ BIMUTAz \end{Bmatrix} = {}^{MBD}\Lambda^{PtfmIMU\,MBD}\vec{a}^{PtfmIMU}$$

$$\begin{Bmatrix} BBSRPxi \\ BBSRPyi \\ BBSRPzi \\ BBSRRoll \\ BBSRPitch \\ BBSRYaw \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\vec{p}^{BS\,Re\,f} \\ \dfrac{180}{\pi} F^{EulerExtract}\left({}^{MBD}\Lambda^{BS\,Re\,f}\right) \end{Bmatrix}$$

$$\begin{Bmatrix} BBSRTVx \\ BBSRTVy \\ BBSRTVz \\ BBSRRVx \\ BBSRRVy \\ BBSRRVz \end{Bmatrix} = \begin{Bmatrix} {}^{MBD}\Lambda^{BS\,Re\,f\,MBD}\vec{v}^{BS\,Re\,f} \\ \dfrac{180}{\pi}\,{}^{MBD}\Lambda^{BS\,Re\,f\,MBD}\vec{\omega}^{BS\,Re\,f} \end{Bmatrix}$$

$$\begin{Bmatrix} BBSRTAx \\ BBSRTAy \\ BBSRTAz \end{Bmatrix} = {}^{MBD}\Lambda^{BS\,Re\,f\,MBD}\vec{a}^{BS\,Re\,f}$$