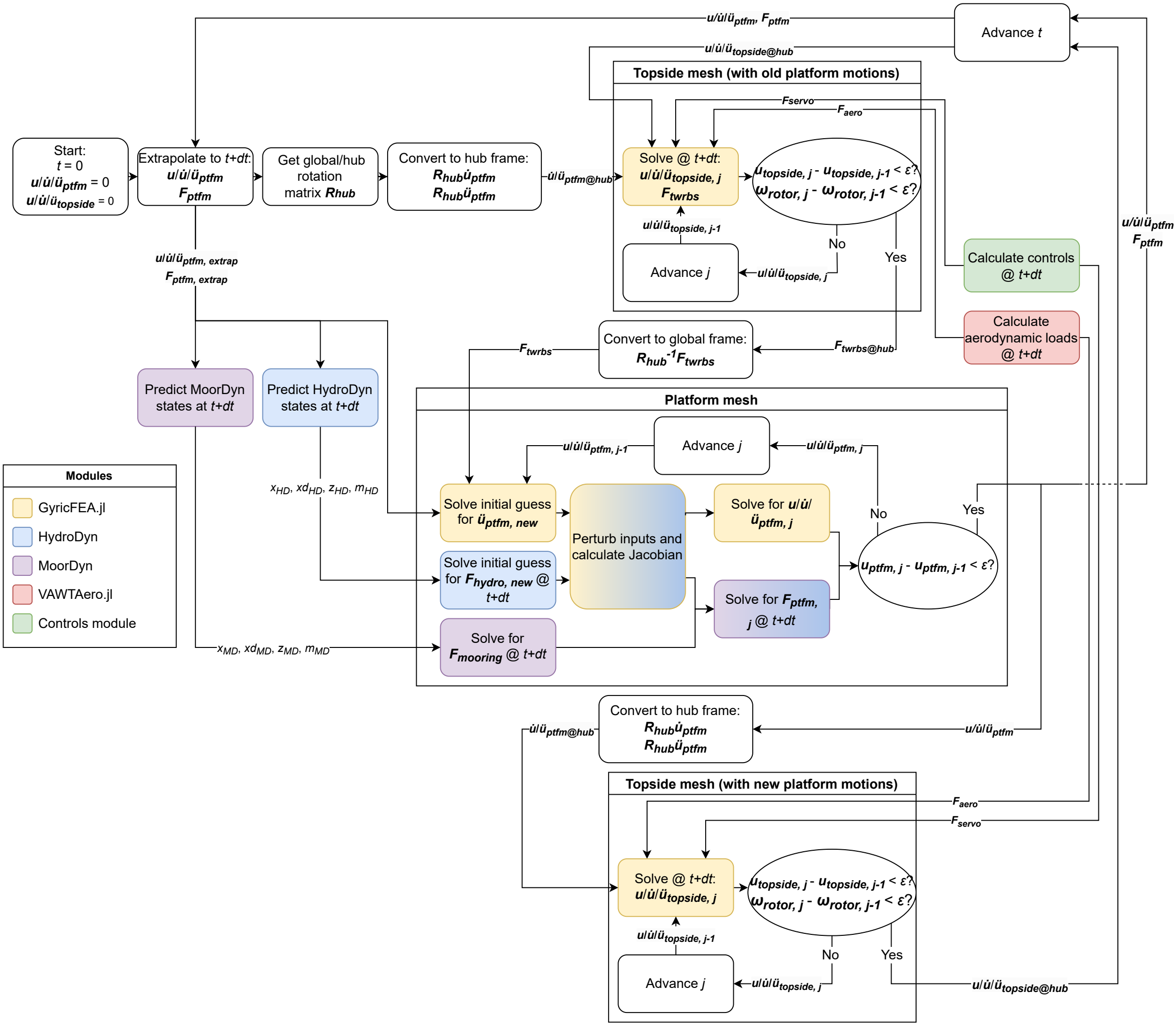
Start:
t = 0
u/u̇/ü_ptfm = 0
u/u̇/ü_topside = 0
Extrapolate to t+dt:
u/u̇/ü_ptfm
F_ptfm
Get global/hub rotation matrix R_hub
Convert to hub frame:
R_hub u̇_ptfm
R_hub ü_ptfm
u̇/ü_ptfm@hub
u/u̇/ü_ptfm, F_ptfm
Advance t
u/u̇/ü_topside@hub
Topside mesh (with old platform motions)
F_servo
F_aero
Solve @ t+dt:
u/u̇/ü_topside, j
F_twrbs
u_topside, j - u_topside, j-1 < ε?
ω_rotor, j - ω_rotor, j-1 < ε?
u/u̇/ü_topside, j-1
Advance j
u/u̇/ü_topside, j
No
Yes
u/u̇/ü_ptfm
F_ptfm
Calculate controls @ t+dt
Calculate aerodynamic loads @ t+dt
u/u̇/ü_ptfm, extrap
F_ptfm, extrap
Convert to global frame:
R_hub^-1 F_twrbs
F_twrbs
F_twrbs@hub
Predict MoorDyn states at t+dt
Predict HydroDyn states at t+dt
Platform mesh
Advance j
u/u̇/ü_ptfm, j-1
u/u̇/ü_ptfm, j
x_HD, xd_HD, z_HD, m_HD
Solve initial guess for ü_ptfm, new
Perturb inputs and calculate Jacobian
Solve for u/u̇/ü_ptfm, j
Solve initial guess for F_hydro, new @ t+dt
Solve for F_ptfm, j @ t+dt
u_ptfm, j - u_ptfm, j-1 < ε?
No
Yes
x_MD, xd_MD, z_MD, m_MD
Solve for F_mooring @ t+dt
Modules
GyricFEA.jl
HydroDyn
MoorDyn
VAWTAero.jl
Controls module
Convert to hub frame:
R_hub u̇_ptfm
R_hub ü_ptfm
u̇/ü_ptfm@hub
u/u̇/ü_ptfm
Topside mesh (with new platform motions)
F_aero
F_servo
Solve @ t+dt:
u/u̇/ü_topside, j
u_topside, j - u_topside, j-1 < ε?
ω_rotor, j - ω_rotor, j-1 < ε?
u/u̇/ü_topside, j-1
Advance j
u/u̇/ü_topside, j
No
Yes
u/u̇/ü_topside@hub