Algorithms in FAST v8

Bonnie Jonkman

March 19, 2014

1 Definitions and Nomenclature

Module Name	**Abbreviation in Module**	**Abbreviation in this Document**
ElastoDyn	ED	ED
AeroDyn	AD	AD
ServoDyn	SrvD	SrvD
SubDyn	SD	SD
HydroDyn	HydroDyn	HD
MAP	MAP	MAP
FEAMooring	FEAM	FEAM
InflowWind	IfW	IfW
IceFloe	IceFloe	IceF

Table 1: Abbreviations for modules in FAST v8

2 Initializations

3 Input-Output Relationships

3.1 Input-Output Solves (Option 2 Before 1)

This algorithm documents the procedure for the Input-Output solves in FAST, assuming all modules are in use. If an individual module is not in use during a particular simulation, the calls to that module's subroutines are omitted and the module's inputs and outputs are neither set nor used.

```
procedure CalcOutputs_And_SolveForInputs()
    y_ED ← ED_CalcOutput(p_ED, u_ED, x_ED, xd_ED, z_ED)

    u_AD ← TransferOutputsToInputs(y_ED)
    y_AD ← AD_CalcOutput(p_AD, u_AD, x_AD, xd_AD, z_AD)

    u_SrvD ← TransferOutputsToInputs(y_ED, y_AD)
    y_SrvD ← SrvD_CalcOutput(p_SrvD, u_SrvD,
                                    x_SrvD, xd_SrvD, z_SrvD)

    u_ED(not platform reference point) ← TransferOutputsToInputs(y_SrvD, y_AD)
    u_HD ← TransferMeshMotions(y_ED)
    u_SD ← TransferMeshMotions(y_ED)
    u_MAP ← TransferMeshMotions(y_ED)
    u_FEAM ← TransferMeshMotions(y_ED)

    ED_HD_SD_Mooring_Ice_InputOutputSolve()

    If AeroDyn or ServoDyn had states to update, we should do this:
        u_AD ← TransferOutputsToInputs(y_ED)
        u_SrvD ← TransferOutputsToInputs(y_ED, y_AD)
    However, they don't so we'll omit these steps for efficiency.
end procedure
```

Note that inputs to *ElastoDyn* before calling CalcOutput() in the first step are not set in CalcOutputs_And_SolveForInputs(). Instead, the *ElastoDyn* inputs are set depending on where CalcOutputs_And_SolveForInputs() is called:

- At time 0, the inputs are the initial guess from *ElastoDyn*;
- On the prediction step, the inputs are extrapolated values from the time history of ElastoDyn inputs;
- On the first correction step, the inputs are the values calculated in the prediction step;
- On subsequent correction steps, the inputs are the values calculated in the previous correction step.

3.2 Input-Output Solve for *HydroDyn*, *SubDyn*, *MAP*, *FEAMooring*, *IceFloe*, and the Platform Reference Point Mesh in *ElastoDyn*

This procedure implements Solve Option 1 for the accelerations and loads in *HydroDyn*, *SubDyn*, *MAP*, *FEAMooring*, and *ElastoDyn* (at its platform reference point mesh). The other input-output relationships for these modules are solved using Solve Option 2.

```
procedure ED_HD_SD_Mooring_Ice_InputOutputSolve()

    y_MAP ← CalcOutput(p_MAP, u_MAP, x_MAP, xd_MAP, z_MAP)
    y_FEAM ← CalcOutput(p_FEAM, u_FEAM, x_FEAM, xd_FEAM, z_FEAM)
    y_IceF ← CalcOutput(p_IceF, u_IceF, x_IceF, xd_IceF, z_IceF)

      ▷ Form u vector using loads and accelerations from u_HD, u_SD, and
    platform reference input from u_ED

    u ← u_vec(u_HD, u_SD, u_ED)
    k ← 0
    loop      ▷ Solve for loads and accelerations (direct feed-through terms)
        y_ED ← ED_CalcOutput(p_ED, u_ED, x_ED, xd_ED, z_ED)
        y_SD ← SD_CalcOutput(p_SD, u_SD, x_SD, xd_SD, z_SD)
        y_HD ← HD_CalcOutput(p_HD, u_HD, x_HD, xd_HD, z_HD)
        if k ≥ k_max then
            exit loop
        end if
        u_MAP_tmp ← TransferMeshMotions(y_ED)
        u_FEAM_tmp ← TransferMeshMotions(y_ED)
        u_IceF_tmp ← TransferMeshMotions(y_SD)
        u_HD_tmp ← TransferMeshMotions(y_ED, y_SD)
        u_SD_tmp ← TransferMeshMotions(y_ED)
                        ∪ TransferMeshLoads(y_HD, u_HD_tmp,
                                            y_IceF, u_IceF_tmp)
        u_ED_tmp ← TransferMeshLoads(y_ED,
                                          y_HD, u_HD_tmp,
                                          y_SD, u_SD_tmp,
                                          y_MAP, u_MAP_tmp,
                                          y_FEAM, u_FEAM_tmp)

        U_Residual ← u − u_vec(u_HD_tmp, u_SD_tmp, u_ED_tmp)

        if last Jacobian was calculated at least DT_UJac seconds ago then
            Calculate ∂U/∂u
        end if
```

Solve $\frac{\partial U}{\partial u}\delta u = -U\_Residual$ for δu

if $\|\delta u\|_2 <$ tolerance **then** ▷ To be implemented later
exit loop
end if

$u \leftarrow u + \delta u$
Transfer u to u_HD, u_SD, and u_ED ▷ loads and accelerations only
$k = k + 1$
end loop
▷ Transfer non-acceleration fields to motion input meshes

u_HD(not accelerations) $\leftarrow$ TransferMeshMotions(y_ED, y_SD)
u_SD(not accelerations) $\leftarrow$ TransferMeshMotions(y_ED)

$u\_MAP \leftarrow$ TransferMeshMotions(y_ED)
$u\_FEAM \leftarrow$ TransferMeshMotions(y_ED)
$u\_IceF \leftarrow$ TransferMeshMotions(y_SD)
end procedure