Algorithms in FAST v8

Bonnie Jonkman

May 20, 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()

  $y\_MAP \leftarrow$ CalcOutput($p\_MAP, u\_MAP, x\_MAP, xd\_MAP, z\_MAP$)
  $y\_FEAM \leftarrow$ CalcOutput($p\_FEAM, u\_FEAM, x\_FEAM, xd\_FEAM, z\_FEAM$)
  $y\_IceF \leftarrow$ 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

  $u \leftarrow$ u_vec(u_HD, u_SD, u_ED)
  $k \leftarrow 0$
  loop ▷ Solve for loads and accelerations (direct feed-through terms)
    $y\_ED \leftarrow$ ED_CalcOutput($p\_ED, u\_ED, x\_ED, xd\_ED, z\_ED$)
    $y\_SD \leftarrow$ SD_CalcOutput($p\_SD, u\_SD, x\_SD, xd\_SD, z\_SD$)
    $y\_HD \leftarrow$ HD_CalcOutput($p\_HD, u\_HD, x\_HD, xd\_HD, z\_HD$)
    if $k \geq k\_max$ **then**
      exit loop
    end if
    $u\_MAP\_tmp \leftarrow$ TransferMeshMotions(y_ED)
    $u\_FEAM\_tmp \leftarrow$ TransferMeshMotions(y_ED)
    $u\_IceF\_tmp \leftarrow$ TransferMeshMotions(y_SD)
    $u\_HD\_tmp \leftarrow$ TransferMeshMotions(y_ED, y_SD)
    $u\_SD\_tmp \leftarrow$ TransferMeshMotions(y_ED)
      $\cup$ TransferMeshLoads(y_SD,
        y_HD, u_HD_tmp,
        y_IceF, u_IceF_tmp)
    $u\_ED\_tmp \leftarrow$ TransferMeshLoads(y_ED,
        y_HD, u_HD_tmp,
        y_SD, u_SD_tmp,
        y_MAP, u_MAP_tmp,
        y_FEAM, u_FEAM_tmp)

    $U\_Residual \leftarrow u -$ u_vec($u\_HD\_tmp, u\_SD\_tmp, u\_ED\_tmp$)

    if last Jacobian was calculated at least DT_UJac seconds ago **then**
      Calculate $\frac{\partial U}{\partial u}$

```
        end if
        Solve ∂U/∂u Δu = −U_Residual for Δu

        if ‖Δu‖_2 < tolerance then                    ▷ To be implemented later
            exit loop
        end if

        u ← u + Δu
        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) ← TransferMeshMotions(y_ED, y_SD)
    u_SD(not accelerations) ← TransferMeshMotions(y_ED)

    u_MAP ← TransferMeshMotions(y_ED)
    u_FEAM ← TransferMeshMotions(y_ED)
    u_IceF ← TransferMeshMotions(y_SD)
end procedure
```

3.3 Implementation of line2-to-line2 loads mapping

The inverse-lumping of loads is computed by a block matrix solve for the distributed forces and moments, using the following equation:

$$\begin{bmatrix} F^{DL} \\ M^{DL} \end{bmatrix} = \begin{bmatrix} A & 0 \\ B & A \end{bmatrix} \begin{bmatrix} F^{D} \\ M^{D} \end{bmatrix} \tag{1}$$

Because the forces do not depend on the moments, we first solve for the distributed forces, F^D:

$$\begin{bmatrix} F^{DL} \end{bmatrix} = \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} F^{D} \end{bmatrix} \tag{2}$$

We then use the known values to solve for the distributed moments, M^D:

$$\begin{bmatrix} M^{DL} \end{bmatrix} = \begin{bmatrix} B & A \end{bmatrix} \begin{bmatrix} F^{D} \\ M^{D} \end{bmatrix} = \begin{bmatrix} B \end{bmatrix} \begin{bmatrix} F^{D} \end{bmatrix} + \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} M^{D} \end{bmatrix} \tag{3}$$

or

$$\begin{bmatrix} M^{DL} \end{bmatrix} - \begin{bmatrix} B \end{bmatrix} \begin{bmatrix} F^{D} \end{bmatrix} = \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} M^{D} \end{bmatrix} \tag{4}$$

Rather than store the matrix B, we directly perform the cross products that the matrix B represents. This makes the left-hand side of Equation 4 known, leaving us with one matrix solve. This solve uses the same matrix A used to obtain the distributed forces in Equation 2; A depends only on element connectivity ***(bjj: check that this is true)***. We use the LU factorization of matrix A so that the second solve does not introduce much additional overhead.