Computational Methods for Transient Coupled-Physics Problems (KAIST Fall 2005/KCP)

18 October 2005: Basic Transient Analysis Techniques

20 October 2005: Staggered Solution Procedures

25 October 2005: Structure-Acoustic Interaction Analysis

27 October 2005: Stabilization Strategies and their Applications

01 November 2005: Fine Points in Transient Analysis Techniques

03 November 2005: Simulation of Multi-Physics Problems

Coupled Problems

Consider the following governing equations of several field problems defined as

$$\begin{aligned} \text{Field Problem S:} \quad & \dot{\mathbf{x}}_s = \mathbf{A}_s \mathbf{x}_s + \mathbf{B}_s \, \mathbf{u}_s \\ \text{Field Problem F:} \quad & \dot{\mathbf{x}}_f = \mathbf{A}_f \mathbf{x}_f + \mathbf{B}_f \, \mathbf{u}_f \\ & \cdots \\ \text{Field Problem E:} \quad & \dot{\mathbf{x}}_e = \mathbf{A}_e \mathbf{x}_e + \mathbf{B}_e \, \mathbf{u}_e \end{aligned} \tag{1}$$

If the excitations in the above governing equations, viz., $\{\mathbf{u}_s, \mathbf{u}_f, \ldots, \mathbf{u}_e\}$ are exclusively function of time only, then the above field equations are denoted as *singled field equations.*

On the other hand, if the excitations are functions of other field variables, viz.,

$$\begin{aligned} \mathbf{u}_s &= \mathbf{u}(t, \mathbf{x}_f, \ldots, \mathbf{x}_e) \\ \mathbf{u}_f &= \mathbf{u}(t, \mathbf{x}_s, \ldots, \mathbf{x}_e) \\ \mathbf{u}_e &= \mathbf{u}(t, \mathbf{x}_s, \ldots, \mathbf{x}_f) \end{aligned} \tag{2}$$

then the field equations modeled by (1) are called *coupled field equations.* If the variables $\{\mathbf{x}_s, \mathbf{x}_f, \ldots, \mathbf{x}_e\}$ represents distinctly different physical phenomena, then the coupled field equations (1) are called *coupled physics problems.*

Special cases of coupled problems: All partitioned equations become coupled problems. Consider a two-bar system when assembled and when partitioned as shown in Fig. 1.

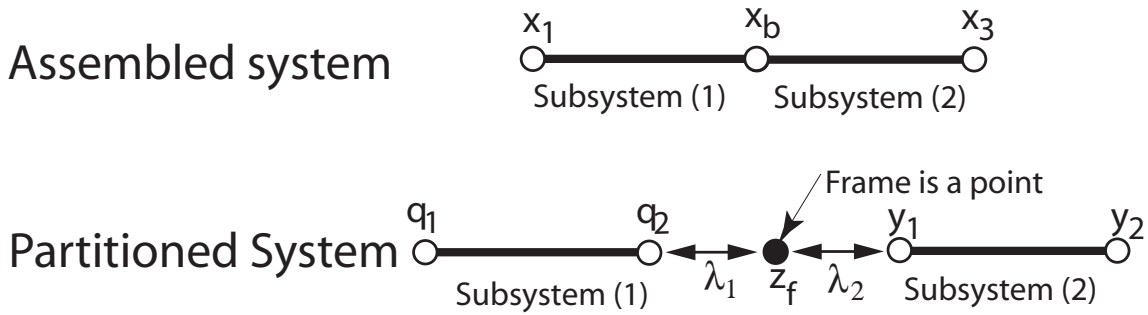

Fig. 1 Two-Bar System Assembled and Partitioned

The equations of motion for the assembled two-bar system may be expressed as

$$\frac{1}{2}\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_1+m_2 & 0 \\ 0 & 0 & m_2 \end{bmatrix}\begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_b \\ \ddot{x}_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1+k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_b \\ x_3 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_b \\ f_3 \end{Bmatrix} \tag{3}$$

The partitioned equations can be written as

For the partitioned bar (1):

$$\frac{m_1}{2}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{Bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{Bmatrix}+k_1\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{Bmatrix}q_1\\ q_2\end{Bmatrix}=\begin{Bmatrix}f_1\\ f_2\end{Bmatrix}-\begin{Bmatrix}0\\ \lambda_1\end{Bmatrix} \quad (4)$$

For the partitioned bar (2):

$$\frac{m_2}{2}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{Bmatrix}\ddot{y}_1\\ \ddot{y}_2\end{Bmatrix}+k_2\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{Bmatrix}y_1\\ y_2\end{Bmatrix}=\begin{Bmatrix}\hat{f}_1\\ \hat{f}_2\end{Bmatrix}-\begin{Bmatrix}\lambda_2\\ 0\end{Bmatrix} \quad (5)$$

with the following interface force balance and kinematical constraint:

$$\begin{aligned}\text{Interface force balance:} \quad & \lambda_1+\lambda_2=0\\ \text{Interface kinematic constraints:} \quad & \mathbf{q}_2-\mathbf{z}_f=0\\ & \mathbf{y}_1-\mathbf{z}_f=0\end{aligned} \quad (6)$$

Clearly, as λ_1 and λ_2 are functions of the field variables, the two partitioned equations (4) and (5) are coupled field equations, although they are not coupled multiphysics problems, where the nature of coupling is stated by (6).

Generalization to large-scale coupled problems:

A generalization of the simple two-bar problems can be expressed in terms of canonical representations as

$$
\begin{aligned}
\dot{\mathbf{x}}_1 &= \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_1\mathbf{u}_1(t) - \mathbf{C}_1\boldsymbol{\lambda}_1 \\
\dot{\mathbf{x}}_2 &= \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_2\mathbf{u}_2(t) - \mathbf{C}_2\boldsymbol{\lambda}_2 \\
\mathbf{C}_1^T\mathbf{x}_1 - \mathbf{L}_1\mathbf{z}_f &= 0 \\
\mathbf{C}_2^T\mathbf{x}_2 - \mathbf{L}_2\mathbf{z}_f &= 0 \\
-(\mathbf{L}^T\boldsymbol{\lambda}_1 + \mathbf{L}_2^T\boldsymbol{\lambda}_2) &= 0
\end{aligned}
\tag{7}
$$

Three Solution Procedures for the Solution of Coupled Systems (7)*:*

- Simultaneous solution (or elimination)
- field variable elimination
- Partitioned solution

Simultaneous Solution:

The matrix form of (7) may be written as

$$\begin{bmatrix} (\frac{d}{dt}\mathbf{I} - \mathbf{A}_1) & 0 & \mathbf{C}_1 & 0 & 0 \\ 0 & (\frac{d}{dt}\mathbf{I} - \mathbf{A}_2) & 0 & \mathbf{C}_2 & 0 \\ \mathbf{C}_1^T & 0 & 0 & 0 & -\mathbf{L}_1 \\ 0 & \mathbf{C}_2^T & 0 & 0 & -\mathbf{L}_2 \\ 0 & 0 & -\mathbf{L}_1^T & -\mathbf{L}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \mathbf{z}_f \end{Bmatrix} = \begin{Bmatrix} \mathbf{B}_1\mathbf{u}_1(t) \\ \mathbf{B}_2\mathbf{u}_2(t) \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{8}$$

Computational issue: Observe that the above equation leads to indefinite system, i.e., some of the diagonal entries are zero, which calls for special treatments in the solution process.

Software modularity issue: Suppose $\mathbf{x}_1$ refers to structures and $\mathbf{x}_2$ refers to fluid or acoustic field. Very seldom, production-level software modules can process two different types of differential equations, hyperbolic and parabolic, in a single solver. In addition, available solvers have been developed for specific field equations, structures, fluid, thermal ,etc.. This means one must integrate several solvers into one solver module, a time-consuming and error-prone endeavor.

Field Variable Elimination:

An alternative approach to solve the coupled equation (8) is to eliminate of the variable, e.g., $\mathbf{x}_2$, as well as the interface Lagrange multipliers ($\boldsymbol{\lambda}$) and frame displacement ($\mathbf{z}_f$). This destroys matrix sparseness and leads to dense solution matrices, which not only present challenge for matrix manipulations but also software merging requirements, thus software modularity is lost.

Field elimination is a leftover legacy from pre-computer era. Because modern matrix software can exploit the sparseness of matrix profiles to optimize the algebraic calculations, field elimination practice is less and less utilized. Consequently, it is discouraged for most applications.

Partitioned Solution:

- Observe that the only unknown in the right-hand side of the first row in the coupled equation(8) is the interface force $\boldsymbol{\lambda}_1$. Hence, if $\boldsymbol{\lambda}_1$ is known, the field equation for $\mathbf{x}_1$ can be solved independently from the coupling variable $\mathbf{x}_2$. This means that one can utilize a modular solver that has been tailired to treat the single field model.
- The preceding observation applies to $\mathbf{x}_2$ provided $\boldsymbol{\lambda}_2$ is known.
- Hence, a partitioned solution of the coupled system model equation (8) is reduced to the task of obtaining the interface force $\boldsymbol{\lambda}$. This will be discussed later in the lecture in more detail.

- *Algorithms for Partitioned Analysis:*

  - Explicit-Explicit Procedure
  - Explicit-Implicit Procedure
  - Implicit-Implicit Procedure
  - Staggered Procedure

Explicit-Explicit Solution Procedure:

There are two ways of implementing an explicit-explicit procedure for coupled systems: modularity-preserving implementation and conventional explicit method. To understand the implementation details, we specialize the coupled system given by equation (8) to two undamped structural subsystems:

$$\begin{bmatrix} (\mathbf{M}_1 \frac{d^2}{dt^2} + \mathbf{K}_1) & 0 & \mathbf{C}_1 & 0 & 0 \\ 0 & (\mathbf{M}_2 \frac{d^2}{dt^2} + \mathbf{K}_2) & 0 & \mathbf{C}_2 & 0 \\ \mathbf{C}_1^T & 0 & 0 & 0 & -\mathbf{L}_1 \\ 0 & \mathbf{C}_2^T & 0 & 0 & -\mathbf{L}_2 \\ 0 & 0 & -\mathbf{L}_1^T & -\mathbf{L}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \mathbf{z}_f \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1(t) \\ \mathbf{f}_2(t) \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{9}$$

The assembled equations of motion from the above partitioned equation set can be expressed as (see, e.g., Fig.1 and equation(3)):

$$\begin{bmatrix} \mathbf{M}_{11} & 0 & 0 \\ 0 & \mathbf{M}_{bb}^{(1)} + \mathbf{M}_{bb}^{(2)} & 0 \\ 0 & 0 & \mathbf{M}_{22} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{y}}_1 \\ \ddot{\mathbf{y}}_b \\ \ddot{\mathbf{y}}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{1b} & 0 \\ \mathbf{K}_{b1} & \mathbf{K}_{bb}^{(1)} + \mathbf{K}_{bb}^{(2)} & \mathbf{K}_{b2} \\ 0 & \mathbf{K}_{2b} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{y}_1 \\ \mathbf{y}_b \\ \mathbf{y}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \mathbf{f}_b \\ \mathbf{f}_2 \end{Bmatrix} \tag{10}$$

where it is understood that

$$\mathbf{x}_1 = \begin{Bmatrix} \mathbf{y}_1 \\ \mathbf{y}_b \end{Bmatrix}, \qquad \mathbf{x}_2 = \begin{Bmatrix} \mathbf{y}_b \\ \mathbf{y}_2 \end{Bmatrix} \tag{11}$$

and the mass matrix is diagonal.

Conventional implementation of explicit-explicit procedure:

The assembled system (10) is node-by-node partitioned as

$$\begin{bmatrix} \mathbf{M}_{11} & 0 \\ 0 & \mathbf{M}_{bb}^{(1)} + \mathbf{M}_{bb}^{(2)} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{y}}_1 \\ \ddot{\mathbf{y}}_b \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{1b} & 0 \\ \mathbf{K}_{b1} & \mathbf{K}_{bb}^{(1)} + \mathbf{K}_{bb}^{(2)} & \mathbf{K}_{b2} \end{bmatrix} \begin{Bmatrix} \mathbf{y}_1 \\ \mathbf{y}_b \\ \mathbf{y}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \mathbf{f}_b \end{Bmatrix} \tag{12}$$

$$\begin{bmatrix} \mathbf{M}_{bb}^{(1)} + \mathbf{M}_{bb}^{(2)} & 0 \\ 0 & \mathbf{M}_{22} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{y}}_b \\ \ddot{\mathbf{y}}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{b1} & \mathbf{K}_{bb}^{(1)} + \mathbf{K}_{bb}^{(2)} & \mathbf{K}_{b2} \\ 0 & \mathbf{K}_{2b} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{y}_1 \\ \mathbf{y}_b \\ \mathbf{y}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_b \\ \mathbf{f}_2 \end{Bmatrix} \tag{13}$$

Rearranging the preceding two equations, we have

$$[\mathbf{M}_1 + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{M}_{bb}^{(2)} \end{bmatrix}]\ddot{\mathbf{x}}_1 + \mathbf{K}_1\mathbf{x}_1 + \begin{bmatrix} 0 & 0 \\ \mathbf{K}_{bb}^{(2)} & \mathbf{K}_{b2} \end{bmatrix}\mathbf{x}_2 = \mathbf{f}_1 \tag{14}$$

$$[\mathbf{M}_2 + \begin{bmatrix} \mathbf{M}_{bb}^{(1)} & 0 \\ 0 & 0 \end{bmatrix}]\ddot{\mathbf{x}}_2 + \mathbf{K}_2\mathbf{x}_2 + \begin{bmatrix} \mathbf{K}_{b1} & \mathbf{K}_{bb}^{(1)} \\ 0 & 0 \end{bmatrix}\mathbf{x}_1 = \mathbf{f}_2 \tag{15}$$

where $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{K}_1$ and $\mathbf{K}_2$ are defined as

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{M}_{11} & 0 \\ 0 & \mathbf{M}_{bb}^{(1)} \end{bmatrix}, \mathbf{K}_1 = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{1b} \\ \mathbf{K}_{b1} & \mathbf{K}_{bb}^{(1)} \end{bmatrix}, \mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_{bb}^{(2)} & 0 \\ 0 & \mathbf{M}_{22} \end{bmatrix}, \mathbf{K}_2 = \begin{bmatrix} \mathbf{K}_{bb}^{(2)} & \mathbf{K}_{b2} \\ \mathbf{K}_{2b} & \mathbf{K}_{22} \end{bmatrix} \quad (16)$$

We recall the central difference method as

$$\begin{aligned} \dot{\mathbf{x}}^{n+1/2} &= \dot{\mathbf{x}}^{n-1/2} + h\ddot{\mathbf{x}}^{n} \\ \mathbf{x}^{n+1} &= \mathbf{x}^{n} + h\dot{\mathbf{x}}^{n+1/2} \end{aligned} \quad (17)$$

Hence, as can be seen from the node-wise partitioned equation(14), in order to compute $\ddot{\mathbf{x}}^n$ the analysis software module of subsystem (1) must borrow the boundary mass matrix $\mathbf{M}_{bb}^{(2)}$ from the software module of subsystem(2), and the internal force vector:

$$\mathbf{f}_b^{int} = [\, \mathbf{K}_{bb}^{(2)} \quad \mathbf{K}_{b2} \,] \, \mathbf{x}_2 \quad (18)$$

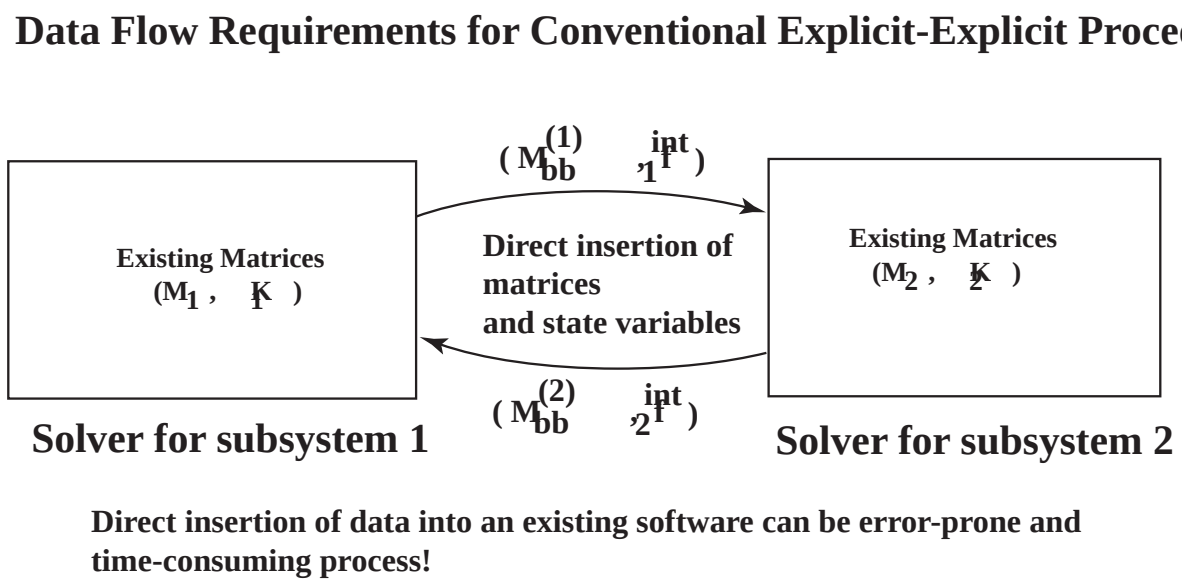

Fig. 1 Data Flow in Conventional Explicit-Explicit Procedure

Observe that it is much easier for us to output matrix and vectors onto an independent output device than to put them into an existing software module. In other words, the node-by-node partitioned equation set given by (12) and (13) does not lead to software modularity for solving the coupled equations.

Of course, one may envision to model the two subsystems in a single software. However, we should keep in mind that we have used the preceding structure-structure coupling as an illustration. Clearly, for

coupled physics problems where $\mathbf{x}_1$ and $\mathbf{x}_2$ represent two different field variables, implementing two distinct physics into one single software module is time consuming and limit their application ranges. We now present an explicit-explicit partitioned procedure that preserves software modularity.

Modular implementation of explicit-explicit procedure:

Let us recall the first two equations from the partitioned equation(9):

$$\begin{aligned} \mathbf{M}_1\ddot{\mathbf{x}}_1 &= \mathbf{f}_1 - \mathbf{K}_1\mathbf{x}_1 - \mathbf{C}_1\boldsymbol{\lambda}_1 \\ \mathbf{M}_2\ddot{\mathbf{x}}_2 &= \mathbf{f}_2 - \mathbf{K}_2\mathbf{x}_2 - \mathbf{C}_2\boldsymbol{\lambda}_2 \end{aligned} \tag{19}$$

To advance from $t + nh$ to $t + (n+1)h$ using the central difference method (17), we need to compute the accelerations:

$$\begin{aligned} \ddot{\mathbf{x}}_1^n &= \mathbf{M}_1^{-1}(\mathbf{f}_1^n - \mathbf{K}_1\mathbf{x}_1^n - \mathbf{C}_1\boldsymbol{\lambda}_1^n) \\ \ddot{\mathbf{x}}_2^n &= \mathbf{M}_2^{-1}(\mathbf{f}_2^n - \mathbf{K}_2\mathbf{x}_2^n - \mathbf{C}_2\boldsymbol{\lambda}_2^n) \end{aligned} \tag{20}$$

We know the present displacements, $(\mathbf{x}_1^n, \mathbf{x}_2^n)$, but not the interface forces, $(\boldsymbol{\lambda}_1^n, \boldsymbol{\lambda}_2^n)$. To compute the latter, we recall the third and fourth equations in (9) and twice differentiate them to obtain:

$$\begin{aligned} \mathbf{C}_1\ddot{\mathbf{x}}_1^n - \mathbf{L}_1\ddot{\mathbf{z}}_f^n &= 0 \\ \mathbf{C}_2\ddot{\mathbf{x}}_2^n - \mathbf{L}_2\ddot{\mathbf{z}}_f^n &= 0 \end{aligned} \tag{21}$$

Substituting (20) into the above equation leads to

$$\begin{bmatrix} \mathbf{C}_1^T\mathbf{M}_1^{-1}\mathbf{C}_1 & 0 & \mathbf{L}_1 \\ 0 & \mathbf{C}_2\mathbf{M}_2^{-1}\mathbf{C}_2 & \mathbf{L}_2 \\ \mathbf{L}_1^T & \mathbf{L}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \ddot{\mathbf{z}}_f \end{Bmatrix}^{n+1} = \begin{Bmatrix} \mathbf{C}_1^T\mathbf{M}_1^{-1}(\mathbf{f}_1 - \mathbf{K}_1\mathbf{x}_1) \\ \mathbf{C}_2^T\mathbf{M}_2^{-1}(\mathbf{f}_2 - \mathbf{K}_2\mathbf{x}_2) \\ 0 \end{Bmatrix}^n \tag{22}$$

Note that when the mass matrices $\mathbf{M}_{(1,2)}$ are diagonal, the computation of the interface from the preceding expression is trivially simple.

Once $\{\boldsymbol{\lambda}_1^n, \boldsymbol{\lambda}_2^n\}$ are computed, the accelerations $\{\ddot{\mathbf{x}}_1^n, \ddot{\mathbf{x}}_2^n\}$ are computed from (20). Subsequently, the displacements and velocities are updated by (17)

The solver for (22) can be constructed in a separate module from the software modules of subsystems 1 and 2 such that no direct data exchange between two subsystem software modules is required.

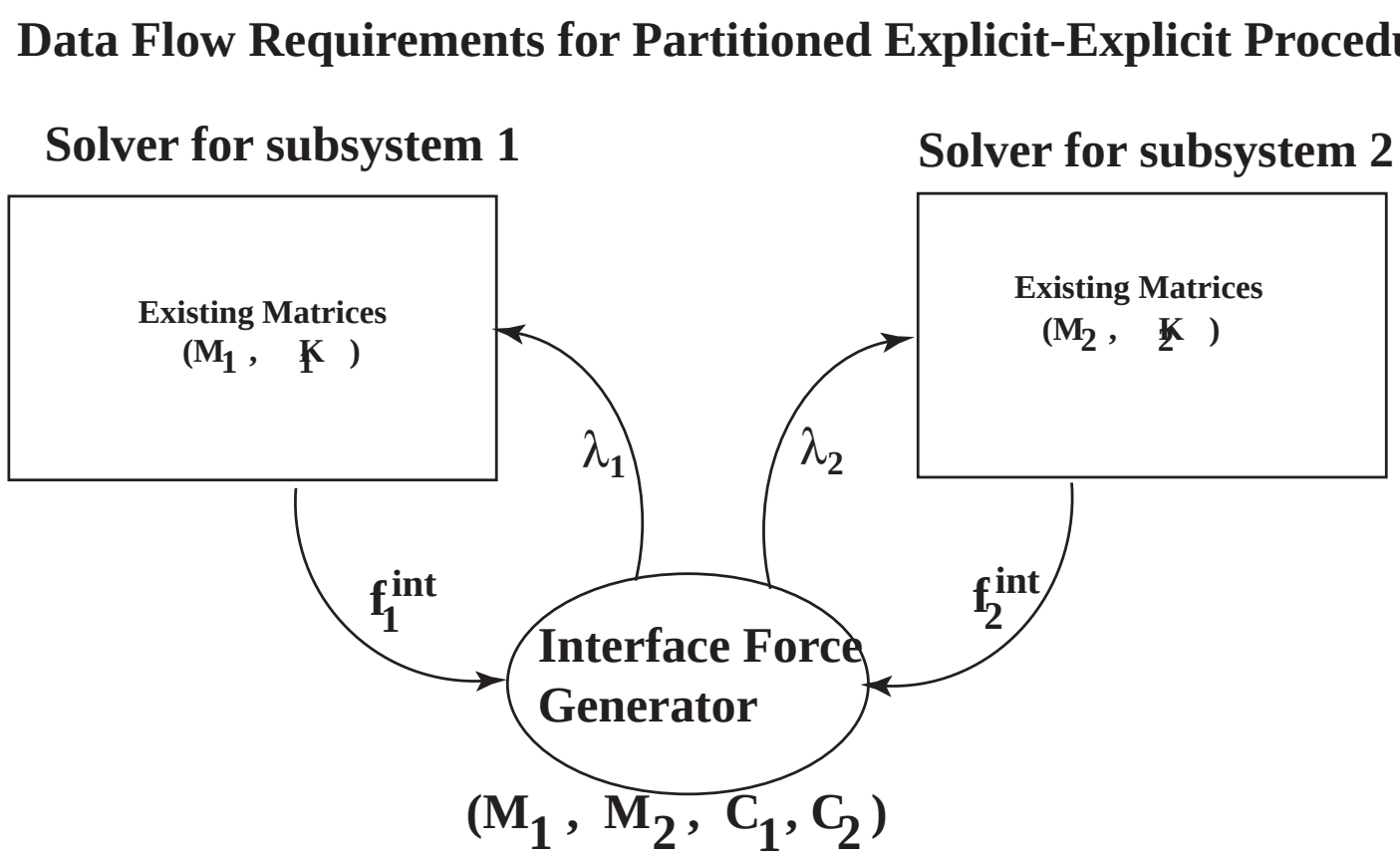

Fig. 2 Data Flow in Partitioned Explicit-Explicit Procedure

Explicit-implicit procedure:

If we solve the first of equation(19) by the central difference method(17), and the second by the trapezoidal rule:

$$\begin{aligned} \dot{\mathbf{x}}_2^{(n+1/2)} &= \dot{\mathbf{x}}_2^n + \delta\ddot{\mathbf{x}}^{(n+1/2)}, \quad \mathbf{x}_2^{(n+1/2)} = \mathbf{x}_2^n + \delta\dot{\mathbf{x}}^{(n+1/2)}, \quad \delta = \tfrac{1}{2}h \\ \dot{\mathbf{x}}_2^{(n+1)} &= 2\dot{\mathbf{x}}_2^{(n+1/2)} - \dot{\mathbf{x}}^{(n)}, \quad \mathbf{x}_2^{(n+1)} = 2\mathbf{x}_2^{(n+1/2)} - \mathbf{x}^{(n)} \end{aligned} \tag{23}$$

The following difference equations result

$$\begin{aligned} \ddot{\mathbf{x}}_1^n &= \mathbf{M}_1^{-1}(\mathbf{f}_1^n - \mathbf{K}_1\mathbf{x}_1^n - \mathbf{C}_1\boldsymbol{\lambda}_1^n) \\ \dot{\mathbf{x}}_1^{n+1/2} &= \dot{\mathbf{x}}_1^{n-1/2} + h\ddot{\mathbf{x}}_1^n \\ \mathbf{x}_1^{n+1} &= \mathbf{x}_1^n + h\dot{\mathbf{x}}_1^{n+1/2} \end{aligned} \tag{24}$$

$$\begin{bmatrix} \mathbf{C}_1^T\mathbf{M}_1^{-1}\mathbf{C}_1 & 0 & \mathbf{L}_1 \\ 0 & \mathbf{C}_2\mathbf{E}_2^{-1}\mathbf{C}_2 & \mathbf{L}_2 \\ \mathbf{L}_1^T & \mathbf{L}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \mathbf{z}_f \end{Bmatrix}^{n+1} = \begin{Bmatrix} \mathbf{C}_1^T\mathbf{M}_1^{-1}(\mathbf{f}_1 - \mathbf{K}_1\mathbf{x}_1) \\ \mathbf{C}_2^T\mathbf{E}_2^{-1}\mathbf{g}_2^n \\ 0 \end{Bmatrix}^n \tag{25}$$

where $\mathbf{E}_2$ and $\mathbf{g}_2^n$ are given by

$$\begin{aligned} \mathbf{E}_2 &= (\mathbf{M}_2 + \delta^2\mathbf{K}_2) \\ \mathbf{g}_2^n &= \mathbf{M}_2(\mathbf{x}_2^n + 2\delta\dot{\mathbf{x}}_2^n) + \delta^2(2\mathbf{f}_2^{(n+1/2)} - \mathbf{K}_2\mathbf{x}_2^n - \mathbf{C}_2\boldsymbol{\lambda}_2^n) \end{aligned} \tag{26}$$

Once the interface forces are computed from (25), the solution for $\mathbf{x}_2$ is obtained by solving the following matrix equation:

$$\begin{aligned}
\mathbf{E}_2\mathbf{x}_2^{(n+1} &= \mathbf{g}^n - \mathbf{C}_2\lambda_2^{(n+1)} \\
\ddot{\mathbf{x}}_2^{(n+1)} &= (\mathbf{f}_2 - \mathbf{K}_2\mathbf{x}^{(n+1} - \mathbf{C}_2\boldsymbol{\lambda}_2)^{(n+1)} \\
\dot{\mathbf{x}}_2^{(n+1)} &= \dot{\mathbf{x}}_2^n + \delta(\ddot{\mathbf{x}}_2^{(n+1)} + \ddot{\mathbf{x}}_2^n)
\end{aligned} \tag{27}$$

For subsequent steps, one repeats the equation sequence from(24) through (27).

The essential difference of the explicit-implicit procedure from the explicit-explicit procedure is that one must factorize the implicit solution matrix $\mathbf{E}_2$. Hence, the stability of the total computational procedure is dictated by the stability limit of the explicit method as the implicit partition assures its unconditional computational stability.

Implicit-implicit procedure:

If we solve equation(19) by the by the trapezoidal rule, we have the following equation set:

$$\begin{bmatrix} \mathbf{C}_1^T\mathbf{E}_1^{-1}\mathbf{C}_1 & 0 & \mathbf{L}_1 \\ 0 & \mathbf{C}_2\mathbf{E}_2^{-1}\mathbf{C}_2 & \mathbf{L}_2 \\ \mathbf{L}_1^T & \mathbf{L}_2^T & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \mathbf{z}_f \end{Bmatrix}^{n+1} = \begin{Bmatrix} \mathbf{C}_1^T\mathbf{E}_1^{-1}\mathbf{g}_1^n \\ \mathbf{C}_2^T\mathbf{E}_2^{-1}\mathbf{g}_2^n \\ 0 \end{Bmatrix}^n \tag{28}$$

where $\{\mathbf{E}_i, \mathbf{g}_i^n, \ i = 1, 2\}$ are given by

$$\begin{aligned} \mathbf{E}_i &= (\mathbf{M}_i + \delta^2\mathbf{K}_i) \\ \mathbf{g}_i^n &= \mathbf{M}_i(\mathbf{x}_i^n + 2\delta\dot{\mathbf{x}}_i^n) + \delta^2(2\mathbf{f}_i^{(n+1/2)} - \mathbf{K}_i\mathbf{x}_i^n - \mathbf{C}_i\boldsymbol{\lambda}_i^n) \end{aligned} \tag{29}$$

Once the interface forces are computed from (28), the solution for $\{\mathbf{x}_i, \ i = 1, 2\}$ is obtained by solving the following matrix equation:

$$\begin{aligned} \mathbf{E}_i\mathbf{x}_i^{(n+1} &= \mathbf{g}_i^n - \mathbf{C}_i\lambda_i^{(n+1)} \\ \ddot{\mathbf{x}}_i^{(n+1)} &= (\mathbf{f}_i - \mathbf{K}_i\mathbf{x}^{(n+1} - \mathbf{C}_i\boldsymbol{\lambda}_i)^{(n+1)} \\ \dot{\mathbf{x}}_i^{(n+1)} &= \dot{\mathbf{x}}_i^n + \delta(\ddot{\mathbf{x}}_i^{(n+1)} + \ddot{\mathbf{x}}_i^n) \end{aligned} \tag{30}$$

For subsequent steps, one repeats the equation sequence from(28) through (30).

Clearly, the implicit-implicit procedure is the most expensive as it requires two matrix factorization and the corresponding back-substitutions, in addition to the implicit solution ofthe interface forces given by (28).

The advantage for the price paid for in matrix factorization is its unconditional stability. For software modularity and computational efficiency, several iterative solution procedures have been proposed to solve the interface forces (28).

Staggered Solution Procedures – to be continued