

Issues Exporting a Multibody Dynamics System Model into a Finite Element Analysis Model

2010-01-0947

Published
04/12/2010

Jose L Ortiz
MSC Software Corporation

Daniel Heiserer
BMW Group

Copyright © 2010 SAE International

ABSTRACT

Nowadays there is an increasing need to streamline CAE processes. One such process consists of translating a Multibody Dynamics System (MBS) model into an equivalent Finite Element Analysis (FEA) model. Typically, users start with the creation of a MBS model which is set at a desired operating point by means of running simulations in the MBS domain (e.g. dynamics, statics.) The MBS model is then further translated into an equivalent FEA model which is used to perform simulations in the FEA domain (e.g. passive safety/crash, noise vibration harshness/NVH.)

Currently, the translation of the MBS model into a FEA model is done either manually or by means of using a user-written script. This paper shows that a user-written script that translates a MBS model into a FEA model can not provide a high fidelity translation. In general, it is found that eigenvalues computed by the FEA code would not match eigenvalues computed by the MBS code.

This work presents mathematical details showing that an accurate translation ought to be performed by an export algorithm from within the MBS code. Examples of such algorithm implementation in MSC Adams (Adams) are presented.

INTRODUCTION

A typical analysis scenario in the auto industry may consist of translating a MBS model into an equivalent FEA model. The MBS model may be an optimized chassis assembly which is simulated using the MBS code up to a desired operating point. The analyst then translates the MBS model into an

equivalent FEA model, refines the model, and finally performs an NVH analysis.

Currently a FEA model for vibration analysis is generated from scratch without making reuse of the MBS model. Such a task is very time extensive and error prone. As an alternative, the task of translating a MBS model into an equivalent FEA model is performed by using a user-written script. However, this paper shows that, in general, the fidelity of such a translation is not accurate enough, considering that eigenvalues computed by the FEA code will not match eigenvalues computed by the MBS code sufficiently. This new approach enables unified modeling of the MBS and FEA world concerning handling and NVH analyses, resulting in reduced modeling effort and reduced development cycles.

This work presents mathematical derivations showing under what circumstances eigenvalues computed by the FEA code will match eigenvalues computed by the MBS code. A layout of an export process is presented along with implementation details and comments. The derivations apply to all known MBS codes in the market (such as Adams) but it is limited to the class of FEA codes which either implement linear constraint equations, or make use of instantaneous linear constraint equations, such as MSC Nastran (Nastran).

Next section presents the basic derivations of the exporting algorithm followed by the description of the main implementation details. Two industry examples are showcased.

MATHEMATICAL DETAILS

Let's assume that both the MBS and FEA codes implement the same set of generalized coordinates q to describe the dynamics of the model. Notice that all MBS codes in the market implement full non linear equations. First, we find the linear form of the equations of motion from within the MBS code.

MBS LINEARIZATION

At the operating point, the Index-3 set of equations of motion and constraint equations [1][2][3] assembled by the MBS code have the following form:

$$\mathbf{M}\ddot{q} + \Phi_q^T \lambda = f \quad (1)$$

$$\Phi(q) = 0 \quad (2)$$

Here $\mathbf{M}$ is the mass matrix, Φ are the non linear constraints, f is a force vector, and λ are the Lagrange multipliers. The objective is to put Equations (1) and (2) into a linear form (state-space representation). With such a goal in mind, the generalized coordinates q can be partitioned into a set of independent and dependent coordinates:

$$q = \begin{Bmatrix} u \\ v \end{Bmatrix} \quad (3)$$

Using this partition, Equations (1) and (2) take the form:

$$\begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \ddot{v} \end{Bmatrix} + \begin{bmatrix} \Phi_u^T \\ \Phi_v^T \end{bmatrix} \lambda = \begin{Bmatrix} f_u \\ f_v \end{Bmatrix} \quad (4)$$

$$\Phi(u, v) = 0 \quad (5)$$

Define:

$$\mathbf{P} = (-\Phi_u^{-1} \Phi_v)^T \quad (6)$$

Matrix $\mathbf{P}$ can be used to eliminate the Lagrange multipliers from Equation (4) to obtain:

$$(M_3 + \mathbf{P}M_1)\ddot{u} + (M_4 + \mathbf{P}M_2)\ddot{v} = f_v + \mathbf{P}f_u \quad (7)$$

To eliminate $\ddot{u}$ from Equation (7) we differentiate Equation (2) twice:

$$\Phi_q \dot{q} + \Phi_t = 0 \quad (8)$$

$$\Phi_q \ddot{q} + (\Phi_q \dot{q})_q \dot{q} + 2\Phi_{qt} \dot{q} + \Phi_{tt} = 0 \quad (9)$$

Using Equation (3), the first term of Equation (9) can be split in two as follows:

$$\Phi_u \ddot{u} + \Phi_v \ddot{v} + (\Phi_q \dot{q})_q \dot{q} + 2\Phi_{qt} \dot{q} + \Phi_{tt} = 0 \quad (10)$$

Solving for $\ddot{u}$ we obtain:

$$\ddot{u} = -\Phi_u^{-1} \Phi_v \ddot{v} - \Phi_u^{-1} ((\Phi_q \dot{q})_q \dot{q} + 2\Phi_{qt} \dot{q} + \Phi_{tt}) \quad (11)$$

$$\ddot{u} = \mathbf{P}^T \ddot{v} + \Psi \quad (12)$$

Term Ψ above can be identified by inspection. Using Equation (12), Equation (7) transforms to:

$$(M_3 \mathbf{P}^T + \mathbf{P}M_1 \mathbf{P}^T + M_4 + \mathbf{P}M_2)\ddot{v} + (M_3 + \mathbf{P}M_1)\Psi = f_v + \mathbf{P}f_u \quad (13)$$

Finally, taking a variation of Equation (13) we obtain the linear form of the original set of equations of motion (1) and (2):

$$\delta\{(M_3 \mathbf{P}^T + \mathbf{P}M_1 \mathbf{P}^T + M_4 + \mathbf{P}M_2)\ddot{v} + (M_3 + \mathbf{P}M_1)\Psi\} = \delta\{f_v + \mathbf{P}f_u\} \quad (14)$$

Observe Equation (13) is written in terms of a set of independent coordinates and it is fully non linear. Equation (14) is the corresponding linear form of the MBS formulation. Although Equations (13) and (14) depend on the derivatives of the dependent states, those derivatives can always be computed from the constraint equations. Equation (14) will be compared with the corresponding equations assembled by the FEA code.

FEA EQUATION ASSEMBLY

FEA codes will assemble the same set of [equations \(1\)](#) and [\(2\)](#) but there are two major differences. First, the constraint equations are either linear or instantaneous linear forms. Second, the states are either linear or incremental. Hence, without loss of generality, the FEA equations of motion [\[4\]](#) [\[5\]](#) can be written as:

$$\begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \begin{Bmatrix} \delta \ddot{u} \\ \delta \ddot{v} \end{Bmatrix} + \begin{bmatrix} \Phi_u^T \\ \Phi_v^T \end{bmatrix} \delta \lambda = \begin{Bmatrix} \delta f_u \\ \delta f_v \end{Bmatrix} \quad (15)$$

Similarly, the constraint equations are expressed as:

$$\Phi_q \delta q = 0 \quad (16)$$

$$\Phi_u \delta u + \Phi_v \delta v = 0 \quad (17)$$

Notice [Equation \(16\)](#) is linear. For example, when using Nastran, all entries of matrix Φ_q are constants either taken from the MPC cards, or evaluated from the instantaneous linearizations of the expressions corresponding to RJOINT cards.

Define:

$$\mathbf{P} = (-\Phi_u^{-1} \Phi_v)^T \quad (18)$$

Matrix $\mathbf{P}$ defined in [Equation \(18\)](#) is numerically identical to matrix $\mathbf{P}$ defined in [Equation \(6\)](#). Nevertheless, [Equation \(18\)](#) defines a constant matrix while [Equation \(6\)](#) defines a non linear expression. Using [Equation \(18\)](#), [Equation \(15\)](#) reduces to:

$$(M_3 + \mathbf{P}M_1)\delta \ddot{u} + (M_4 + \mathbf{P}M_2)\delta \ddot{v} = \delta f_v + \mathbf{P} \delta f_u \quad (19)$$

An expression for $\delta \ddot{u}$ can be obtained by differentiating [Equation \(17\)](#) twice and then using [Equation \(18\)](#) to obtain:

$$\delta \ddot{u} = \mathbf{P}^T \delta \ddot{v} \quad (20)$$

Using [Equation \(20\)](#), [Equation \(19\)](#) reduces to:

$$(M_3 \mathbf{P}^T + \mathbf{P}M_1 \mathbf{P}^T + M_4 + \mathbf{P}M_2)\delta \ddot{v} = \delta f_v + \mathbf{P} \delta f_u \quad (21)$$

[Equation \(21\)](#) is the final set of equations assembled by the FEA code.

The fundamental question is: Would eigenvalues computed from [Equation \(21\)](#) match eigenvalues computed from [Equation \(14\)](#)?

MATCHING EIGENVALUES

In order to find under what circumstances eigenvalues computed using [Equation \(21\)](#) match eigenvalues computed using [Equation \(14\)](#), we take the variation of the later to obtain:

$$(M_3 \mathbf{P}^T + \mathbf{P}M_1 \mathbf{P}^T + M_4 + \mathbf{P}M_2)\delta \ddot{v} + \{\delta(M_3 \mathbf{P}^T + \mathbf{P}M_1 \mathbf{P}^T + M_4 + \mathbf{P}M_2)\ddot{v} + \delta((M_3 + \mathbf{P}M_1)\Psi)\} = \delta f_v + \mathbf{P} \delta f_u + \delta \mathbf{P} f_u \quad (22)$$

Comparing [Equations \(22\)](#) with [Equation \(21\)](#) we find that eigenvalues computed by the FEA code will match eigenvalues computed by the MBS code only if the following conditions are met:

- The model is in static equilibrium.
- The term Φ_u shown in [Equation \(11\)](#) is zero (e.g. the model has scleronomic constraints). This condition is usually met by the great majority of MBS models in the auto industry.
- The export algorithm provides the term $\delta \mathbf{P} f_u$ shown on the right-hand side of [Equation \(22\)](#). In general, this term is difficult to evaluate either symbolically or numerically. This term could be zero depending on the nature of the constraints and the applied forces.

If the first two conditions are met, then the terms between curly brackets on the left-hand side of [Equation \(22\)](#) vanish. The first term between curly brackets vanishes because $\ddot{v} = 0$ at static equilibrium. Also, given that velocities are zero, term Ψ is stationary and equal to zero. The third condition implies that an extra residual force $\delta \mathbf{P} f_u$ needs to be exported.

For the sake of verification, let's assume matrix $\mathbf{P}$ in [Equation \(22\)](#) is a constant. If that is the case, [Equation \(22\)](#) would match [Equation \(21\)](#) exactly. Therefore, the residual force can be interpreted as a correction to compensate for the limitation of the FEA codes regarding the implementation of the constraint equations.