Model Based Partitioned Simulation of Coupled Systems

Carlos A. Felippa† and K. C. Park‡

† Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado at Boulder, Boulder, Colorado 80309-0429, USA
email: carlos.felippa@colorado.edu

Abstract. This tutorial paper is extracted from a set of graduate lectures on the time-domain simulation of structural dynamics and coupled systems. This material has also served as a basis for a CISM lecture series on FSI. For the treatment of coupled systems, emphasis is placed on partitioned analysis procedures. Although the subject emerged in the present form over 20 years ago, the time-consuming study of competing formulations and implementations can be streamlined through the use of various tools such as reduction to model equations, and the help of computer algebra systems.

Keywords: computational structural dynamics, coupled systems, fluid-structure interaction, multiphysics, computer algebra, partitioned analysis, time integration.

1 Introduction

What’s hot in computational mechanics? The three “multis”: multiscale, multiphysics and multiprocessing. Collectively these trends pertain to the formulation and model-based simulation of coupled systems: systems whose behavior is driven by the interaction of functionally distinct components. The nature of these components broadly defines the “multi” discipline. Material models that span a range of physical scales (for example, molecular through crystal) are the framework of multiscale simulations. Multiphysics addresses the interaction of different physical behavior, as in structures and fluids, at similar physical scales. Multiprocessing refers to computational methods that use system decomposition to achieve concurrency. Summarizing, system breakdown is governed by: (S) physical scales in multiscale, (P) physical behavior in multiphysics, and (C) computer implementation considerations in multiprocessing. Plainly a three-level hierarchy: (S)-(P)-(C), can be discerned, but this level of full generality has not been reached in practice.

The “hot areas” share a common feature: explosive complexity. Choices among models, algorithms and implementations grow combinatorially in the number of components. Consider for example a fluid-structure interaction problem.
Whereas a FEM model for the structure may be viewed as natural, the choice of fluid model can vary across a wide spectrum, depending on what physical effects (flow, turbulence, acoustic shocks, mixing, slosh, cavitation, moving boundaries, bubbles, etc.) are to be captured. Discretization methods vary accordingly. If control is added to the picture, for example to simulate maneuvers of a flexible airplane, further choices emerge. So far this applies to components in isolation. Treating interaction requires additional decisions at interfaces. For example: do meshes match? can meshes slip past each other? how can reduced or spectral models be linked to physical models? To make things more difficult, often models that work correctly with isolated components break down when coupled. But that is not all. Proceeding to the computer implementation and testing levels may bring up further options, in particular if parallel processing issues are important.

How to cope with this combinatorial explosion? Analytical treatment can go so far in weeding out choices. The traditional way to go beyond that frontier is numerical experimentation. This has limitations: the most one can hope to do is take “potshots” at the computational application domain. It can only show that a particular numerical model works or doesn’t. A “bridging” tool between human analytical thought and numerical testing has gained popularity over the past decade: computer algebra systems (CAS) able to carry out symbolic computations. This is due to technical improvements in general-purpose CAS such as Mathematica and Maple, as well as availability on inexpensive personal computers and laptops. (This migration keeps licensing costs reasonable.) Furthermore, Maple is accessible as a toolbox of the widely used Matlab system. A related factor is wider exposure in higher education: many universities now have site licenses, which facilitate access and use of CAS for course assignments and projects.

In computational mechanics, CAS tools can be used for a spectrum of tasks: formulation, prototyping, implementation, performance evaluation, and automatic code generation. Although occasionally advertised as “doing mathematics by computer” the phrase is misleading: as of now only humans can do mathematics. But a CAS can provide timely help. Here is a first-hand example: the first author needed four months to formulate, implement and test the 6-node finite element triangle in 1965 as part of thesis work [7]. Using a CAS, a similar process can be completed in less than a week, and demonstrated to students in 20 minutes.

In the present chapter, Mathematica [41] is employed as a CAS “filter tool” in the design and analysis of time integration methods for structural dynamics and coupled systems. The main purpose of the filter is to weed out unsuitable methods by working on model test systems. This initial pass streamlines subsequent stages of numerical experimentation and computer implementation.