8th. World Congress on Computational Mechanics (WCCM8)
5th. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008)
June 30 – July 5, 2008

Exploiting Parallelism in Ground Vehicle Dynamics

***Michael K. McCullough [1], William C. Prescott [2] and Ashok Khubchandani [3]**

[1] Michael K. McCullough
1205 Coleman Ave
Santa Clara CA 95124
USA
Michael.mccullough@baesystems.com

[2] William C. Prescott
LMS North America
2651 Crosspark Road
Coralville, IA 52246
USA
Bill.prescott@lmsintl.com

[3] Ashok Khubchandani
1205 Coleman Ave
Santa Clara CA 95124
USA
Ashok.khubchandani@baesystems.com

Key Words: *Parallel processing, vehicle dynamics, Tracked suspension, terrain interaction, contact elements, dynamic coupling, shared memory*

ABSTRACT

A tracked vehicle suspension system model is used as an example of a large scale multibody dynamics model to study the potential for exploiting parallel processing and the types of architectural changes that must be made in solver algorithms as well as the dependence of solver algorithms on particular computer hardware architectures. General formulations for multi-body dynamics applications and their model solver algorithms possess many attributes which have inhibited exploitation of parallel processing. Extensive research and theoretical success has been achieved in quantifying the means and benefit for optimally exploiting parallelism in the kinematic topology of multibody systems [1], by decomposition into independent kinematic chains. However, as the tracked vehicle suspension system will serve to illustrate, kinematic dependence is not always the most computationally burdensome form of coupling in multibody systems. Practical applications in vehicle dynamics frequently require a broader interpretation of the system topology to support identification of independent computational threads and also to minimize the communication bandwidth burdens [2] between and among the threads. Using examples from ground vehicle suspension dynamics the weaker forms of coupling in multi-body systems are defined which lead to specific architectural changes in a commercial multi-body solver written for single processor computers. Expanding upon the topological constructs and simplified numerical experiments introduced earlier [2], this paper reports on the conclusions regarding the potential for parallelization of multi-body algorithms drawn from numerical experiments using the representative example problem, a flexibly linked, tracked suspension vehicle dynamics model (Figure 1).

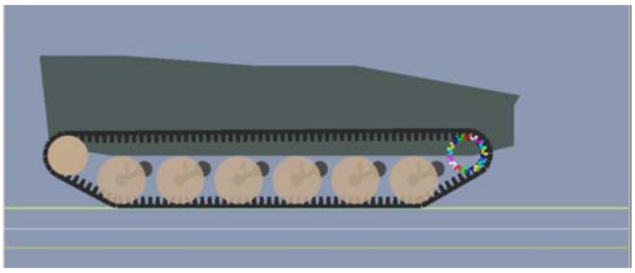
Figure 1: High Mobility Tracked Vehicle Vehicle Dynamics Model

The dependence of results upon the particular algorithmic formulation and solver code architecture used is made thru a direct comparison of the context of these two investigations. The original work, performed using an open source object-oriented multi-body dynamics code [3], was limited by the constructs of that code. Similarly, many of the experiments performed in the current investigation were attempts to overcome limitations of the particular formulation of the commercial multi-body dynamics computer code [4] used for this study. In spite of the difference in the multi-body solvers used, similar results were eventually obtained with respect to the size of shared memory, particularly cache memory for multicore processors, the need to minimize inter-process data communications, and the point at which there is diminishing efficiency of added processors.

To highlight the dependence upon algorithmic formulation results will also be contrasted with recent advances in implicit integration of multi-body dynamics equations. An attractive alternative to explicit integration, an implicit integrator can take a much larger step size than an explicit integrator, often resulting in a shorter simulation. However, an implicit formulation requires the Jacobian of the equations of motion, including generalized forces, with respect to the generalized coordinates. This creates a numerical coupling among force and mass quantities that requires additional considerations in attempting to decompose the system into separate computational threads. These include various ways to implement thread-safe sparse matrix solutions, and ways to achieve subsystem decomposition while maintaining the accuracy and stability of the solution.

Table 1 shows preliminary run times for various threads used by an explicit integrator. These results will be contrasted with those from the implicit integration scheme and parallelization of that solver. The reduction in benefit with the fifth processor is a direct consequence of the computer hardware architecture, as this required communication across the data bus connecting the two processor boards. Extrapolation to expected results on larger problems using larger multiprocessor machines with a faster interprocess communication backplane are also developed.

Table 1: Quad Processor Speedup Results for Explicit Solution

No. of threads	**CPU Times (sec)**	**Wall time (Sec)**	**Speedup (baseline Linux 1 thread)**
1	3043	3085	1.00
2	3352	2191	1.41
3	3070	1623	1.90
4	3212	1401	2.20
5	3143	1501	2.06

REFERENCES

[1] Critchley, J. H. “A Parallel Logarithmic Time Complexity Algorithm For the Simulation of General Multibody System Dynamics”. PhD Thesis, Rensselaer Polytechnic Institute, Troy NY, March 2003.

[2] J. Critchley and M. McCullough, “APPLICATION OF PARALLEL COMPUTING TO JOINT-DISCONNECTED MULTIBODY SYSTEMS”, DETC2005-84049 Proceedings of ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference Long Beach, California, USA, September 24-28, 2005.

[3] YAMS Development Team, 2005. YetAnotherMultibodySimulator (YAMS). On the World Wide Web at http://www.multibodydynamics.com

[4] LMS-North America, VL.Motion Rev 7A beta test version. August 2007.