A comparison of Algorithms for Sparse Matrix Factoring and Variable Reordering aimed at Real-time Multibody Dynamic Simulation

Jose-Luis Torres-Moreno, Jose-Luis Blanco, Javier López-Martínez, Antonio Giménez-Fernández [1]

[1] Department of Engineering, University of Almería {jltmoreno,jlblanco,jlm167,agimfer}@ual.es

Abstract

The increasing competitiveness of sectors such as automotive, robotics or spatial aircraft manufacturing calls for a reduction of their lifecycle-design stages. In this context, being able to perform dynamic analyses of complex Multibody Systems (MBS) before manufacturing is becoming more and more valuable. This discipline is useful for studying the dynamic behavior of such products, but also for testing electronic controllers even before a prototype is built. Not surprisingly, it is a common practice to combine both real components and simulation models through Hardware-in-the-loop testing, i.e. testing a real electronic control unit (ECU) with the dynamics of a simulated vehicle. A natural prerequisite for those applications is being able to solve the equations of motion that govern these models faster than real-time. However, this kind of problems involves computationally expensive linear algebra operations, hence the interest in identifying the computational bottlenecks in order to discern on which operations shall we focus our optimization efforts.

The aim of this paper is comparing the overall computation time spent in solving the equations required for performing a dynamic simulation of a MBS, using different linear algebra packages freely available as Open Source C/C++ libraries. We focus on different matrix factoring and variable ordering algorithms, since their selection is shown to have a great impact on performance. The benchmark is carried out by means of real-time capable code that implements an augmented Lagrangian formulation based on the principle of virtual powers, modeled in natural coordinates and using the fourth-order Runge-Kutta numerical integration method. The test problem for our benchmark is an $N_x \times N_y$ four-bar planar mechanism, similar to the one employed in [1] but with multiple (N_y) loops stacked vertically. Such a MBS has N_y degrees-of-freedom. The elements have a uniformly-distributed mass of 1 kg each and the unique acting forces are the body weights.

Let the MBS be characterized by N natural coordinates $\mathbf{q}$ which, being redundant, require the introduction of M kinematic constraint equations: $\mathbf{\Phi}(\mathbf{q}, t) = 0$. The equations of motion for such a MBS can be formulated from the application of the method of virtual powers [2]. Taking into account the forces at the pairs, one arrives at $\dot{\mathbf{q}}^{*\top}(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}) + \mathbf{\Phi}_q^\top \boldsymbol{\lambda} = 0$, where $\dot{\mathbf{q}}^*$ are the virtual velocities, which must satisfy the time derivative of $\mathbf{\Phi}(\mathbf{q}, t)$ at a stationary time, $\mathbf{M}$ is the mass matrix, $\mathbf{Q}$ represents the generalized forces, $\mathbf{\Phi_q}$ is the Jacobian of the constraint equations and $\boldsymbol{\lambda}$ is a vector containing the Lagrange multipliers which determine the magnitude of the constrained forces. Therefore, for a general MBS we have $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_q^\top \boldsymbol{\lambda} = \mathbf{Q}$. This equation and the kinematic restrictions form a set of $N + M$ mixed differential algebraic equations (DAE) of index-3, whose unknowns are $\mathbf{q}$ and $\boldsymbol{\lambda}$. However, by differentiating twice the constraint equations this system is reduced to an index-1 system which is more easily solved. Rearranging in matrix form leads to:

$$\underbrace{\left[\begin{array}{c|c} \mathbf{M} & \mathbf{\Phi_q}^\top \\ \hline \mathbf{\Phi_q} & \mathbf{0} \end{array}\right]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix}}_{\mathbf{b}} \text{, with } \mathbf{c} = -\dot{\mathbf{\Phi}}_q \dot{\mathbf{q}} - 2\xi\omega\dot{\mathbf{\Phi}} - \omega^2 \mathbf{\Phi} \qquad (1)$$

Therefore, performing real-time dynamic simulations implies being able to solve for the unknown accelerations $\ddot{\mathbf{q}}$ at each time step as many times as required by the employed numerical integration method (e.g. four times with a 4^{th} order Runge-Kutta). Any efficient approach to solve for the $\ddot{\mathbf{q}}$ in (1) should account for the sparsity pattern of the augmented matrix $\mathbf{A}$. In particular, $\mathbf{M}$ is block sparse and positive definite, $\mathbf{\Phi_q}$ is unsymmetric but highly sparse, while $\mathbf{A}$ as a whole is symmetric but neither positive nor negative definite.

These features allow us to employ the following different methods to solve the $\mathbf{Ax} = \mathbf{b}$ system in (1):

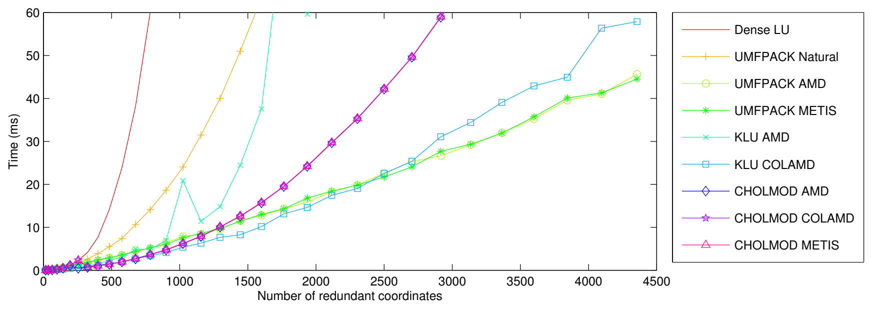
1. **LU factorization:** By decomposing the augmented matrix as $\mathbf{A} = \mathbf{LU}$, the linear system $\mathbf{LUx} = \mathbf{b}$ can be efficiently solved by forward ($\mathbf{L}\tilde{\mathbf{y}} = \mathbf{b}$) and back-substitution ($\mathbf{Ux} = \tilde{\mathbf{y}}$).

This work has been partially funded by the Spanish "Ministerio de Ciencia e Innovación" under the contract "DAVARBOT" (DPI 2011-22513) and the grant program JDC-MICINN 2011, as well as by the Andalusian Regional Government grant programs FPDU 2008 and FPDU 2009, cofinanced by the European Union through the European Regional Development Fund (ERDF).

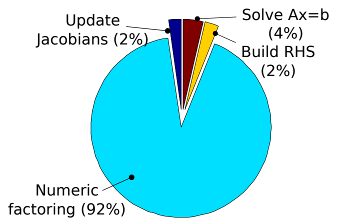
2. **Schür complement-based factorization:** By means of a smart block-factorization of $\mathbf{A}$ we can turn the system (1) into two coupled linear systems whose coefficient matrices are both definite positive, enabling the application of the efficient Cholesky ($\mathbf{LL}^\top$) decomposition twice, then solving by forward and back-substitution. Please, refer to [3] for a detailed description of this technique.

For our benchmark we implemented the following solvers: (i) a dense LU solver (i.e. which does not exploit the problem sparse pattern), (ii) two sparse LU solvers, using the algorithms UMFPACK [4] and KLU, and (iii) a Schür complement-based sparse solver based on the popular CHOLMOD algorithm for the two required Cholesky factorizations. The C++ library Eigen has been used for implementing the dense LU solver. Eigen has demonstrated in benchmarks to be faster than free implementations of BLAS (e.g. ATLAS, uBlas) and of a comparable performance to non-free versions (e.g. Intel MKL, GOTO).

We have evaluated the performance of each solver during a dynamic simulation of our parameterized $N_x \times N_y$ multibody benchmark for $N = N_x = N_y$ values ranging from 1 to 32. Notice that this leads to a constrained MBS with $N = N_y$ degrees of freedom. The times taken by each solver are summarized in Fig. 1(a) for increasing values of N. Notice that, excepting the LU dense implementation, all other sparse solvers have been designed to carry out a *symbolic factorization* of the matrices only once at the beginning of the simulation, since the structure of the problem matrices do not change. Therefore, during a real-time simulation it becomes enough to: (i) update the numeric values of the sparse Jacobians and the RHS vector ($\mathbf{b}$ in (1)), (ii) compute the numeric factorization and (iii) solve the linear system ($\mathbf{Ax} = \mathbf{b}$). As it is well-known from algebra and graph-theory, reordering the problem variables becomes crucial in determining the density of the factored matrices which, in turn, determines the cost of the numeric factorization –the computationally most expensive step in our implementation, as shown in Fig. 1(b). That is why we tested each algorithm with different variable orderings: "natural" (leaving variables as they are defined), AMD (Approximate Minimum Degree), COLAMD (Column AMD) and METIS (a graph partitioning algorithm).



(a)



(b)

Figure 1. (a) Computation times for each algorithm and ordering strategy for one complete time step (including numeric factorization and linear system solving). (b) A typical distribution of times within one time step for the UMFPACK algorithm with AMD ordering.

Our conclusion is that CHOLMOD (with any ordering) and KLU+COLAMD are the most efficient solvers for mid-size MBS problems, whereas UMFPACK (with either AMD or METIS) emerges as the optimal choice as the number of bodies increases above the few hundreds.

References

[1] M. González, F. González, D. Dopico, and A. Luaces. On the effect of linear algebra implementations in real-time multibody system dynamics. *Computational Mechanics*, 41(4):607–615, 2008.

[2] J. Garcia de Jalon and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems*. Springer-Verlag, 1994.

[3] R. Von Schwerin. *Multibody system simulation: numerical methods, algorithms, and software*, volume 7. Springer, 1999.

[4] T.A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):165–195, 2004.