A partitioning method for parallelization of large systems in realtime.

Claude Lacoursière*, Fredrik Nordfeldth†, Mattias Linde+,

HPC2N/UMIT
University of Umeå
SE-901 87, Umeå, Sweden
* claude@hpc2n.umu.se
+ linde@cs.umu.se

Algoryx Simulation AB
SE-907 19, Umeå, Sweden
† fredrik.nordheldth@algoryx.se

We consider real-time simulation of multibody systems in the context of a virtual environment training simulator for sailors manipulating anchoring and tugging cables for stabilizing oil rigs. In such a scenario, there can be several ships mounted with cranes and winches, one oil rig, and several cables modeled with discrete elements [5]. Dry friction is also omnipresent is these systems. The scene used to produce the data in Fig. 1 2 contained a nearly one thousand rigid bodies and elements, as well as three thousand interactions which are either kinematic constraints or very stiff forces. Integrating the Differential Algebraic Equations (DAE)s of motions of such a system requires the solution of sparse linear systems of equations up to ten thousand equations in as many unknowns. In the real-time context of a 3D application running at the standard 60Hz rate, this means a computational budget of roughly 5ms, leaving time for other parts of the application. The only reasonable solution to this is parallelization. This is well understood and active field of research. However, the overhead of current techniques and libraries is such that they are advantageous only for very large systems, and certainly not in the real-time context. This goes both for graph analysis – METIS[2] can take more than 2ms on the problems we consider – and plain factorization [4]. In addition, good load balancing graph analysis with METIS for instance generally produces many fill-ins and offsets benefits of parallelism towards much larger problems.

We discuss two aspect of the solution to this problem. The first is a much simplified graph analysis which takes some of the physics into account in the heuristics. The second is a matrix splitting which decouples two subsystems in such a way that strong interactions are taken into account in a stable way, but at the cost of accuracy.

The physics based heuristics assigns favorable cut weights only where bodies in a small clique are connected with interactions that generate relatively low net forces, and which have low relative velocities in the direction of the applied forces or kinematic constraints. In the present context, these conditions are usually met for the elements of the cables which are linked with two and three body interactions for stretch and bend deformations.

The other heuristic is the estimation of the cost of an "island" which is a group of connected bodies. This cost is crucial for load balancing. However, one needs something like AMD[1] which is expensive. Therefore, we compute the degree n_b of each body–the number of interactions it participates in–and the length l_i of all the interaction loops. We sum $c = \sum_b n_b^2 + \sum_i l$ for the approximate cost. This is the same as the overall METIS algorithm but with extreme simplifications at each step.

At this point, we have the problem of splitting the cutting nodes. Consider a multibody system with block diagonal mass matrix M, generalized positions x, and generalized velocities $\dot{x} = v$. This is subject to constraints $g(x) = 0$ with Jacobians $G = \partial g/\partial x$. We also include strong potentials $U = (1/2\varepsilon)\|g_s(s)\|^2$ with $G_s = \partial g_s/\partial x$. Our time-stepping algorithm which is a variant of RATTLE and the implicit midpoint rule can be written in one of the two following forms

$$\begin{bmatrix} M & -G^T & -G_s^T \\ G & 0 & 0 \\ G_s & 0 & T \end{bmatrix} \begin{bmatrix} v_{k+1} \\ \lambda \\ \lambda_s \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \text{ or } \begin{bmatrix} [M + G_s^T T^{-1} G_s] & -G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} v_{k+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} \tilde{a} \\ b \end{bmatrix}. \tag{1}$$

Here, $T = O(\varepsilon)$ is a block diagonal matrix of compliance, so that T^{-1} is a stiffness matrix, $a = Mv_k + hf_e$ contains the external, non-stiff forces, and the details of vectors a and b depend on the discretization and constraint stabilization scheme. Our choices are explained elsewhere [3].

Our two strategies are as follows. The "kinematic" version is to consider bodies that belong to island II in Fig. 1 in the coupling block at the center, as kinematic as far as system I is concerned, and vice versa. This leads to small jerks since any force that should propagate to the finite, but presumably large inertia of the other system, is now absorbed by an infinite mass.

The second strategy is to replace some constraint equations in the vector g and move them instead in g_s, obtaining the second form in Eqn. (1), which gives the "added mass" from the $G^T T^{-1} G$ term. This can

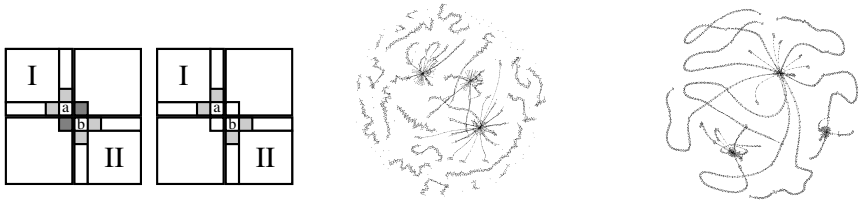

Figure 1: From left to right: the system matrix in full, the split matrix, the interaction graph, the cut graph. In the matrix diagram, the shaded entries are nonzero.

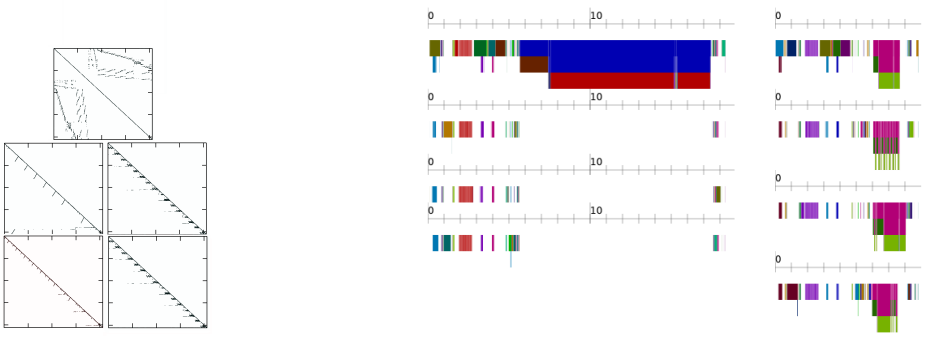

Figure 2: On left panel on top is the spy diagram of the full system matrix. Below are the spy diagrams of from AMD and METIS reordering the full system, left to right. Below, in the same order, and the reordering after island separation and splitting. On the left are timing diagram with one horizontal line per thread. The first picture is for the serial version, where the time for linear algebra is in blue and purple. On the far right is the split version where the linear algebra time is in purple and green. This shows that the load balance is good and that real-time is achieved with the splitting heuristics.

be done safely when g_s is not too large, so that is used as a weight in our graph partitioning algorithm. We then form the clique system, which is the central block in Fig. 1. When this is split, we make an approximation of $\bar{v}_{k+1}^{(i)}$ for each subsystem based on current velocities and forces. Using these techniques, as shown in Fig. 2, we have managed good speedup on a single four core CPU with four threads per core, and this made the application reach the real-time regime, leaving sufficient resources available for the other tasks needed for the application. Strong, stable, and numerically inexpensive coupling is still an issue and ideally, we would like to avoid this and rely solely on direct factorization, and this is something we are working on.

Acknowledgments

This work was supported by Konsberg Maritime, Algoryx Simulations AB, and by the High Performance Computing Center North (HPC2N).

References

[1] Patrick R. Amestoy, Enseeiht-Irit, Timothy A. Davis, and Iain S. Duff. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):381–388, 2004. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1024074.1024081.

[2] Geroge Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graph. *SIAM J. Sci. Comp.*, 20(1):359–392, 1998.

[3] Claude Lacoursière. Regularized, stabilized, variational methods for multibodies. In P. Bunus and et. al., editors, *SIMS 2007*, pages 40–48. Linköping Univ. E. Press, Dec 2007.

[4] Claude Lacoursière, Mattias Linde, and Olof Sabelström. Direct sparse factorization of blocked saddle point matrices. In *PARA 2010, Part II*, volume 7134 of *LNCS*, pages 324–335. Springer, 2012.

[5] Martin Servin, Claude Lacoursiere, Fredrik Nordfelth, and Kenneth Bodin. Hybrid, multi-resolution wires with massless frictional contacts. *IEEE TVGC*, 17(7):970–982, 2011. ISSN 1077–2626.