Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 576–585

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Exploiting Hybrid Parallelism in the Kinematic Analysis of Multibody Systems Based on Group Equations

Gregorio Bernabé[1], José-Carlos Cano[2], Javier Cuenca[1], Antonio Flores[1], Domingo Giménez[2], Mariano Saura-Sánchez[3], and Pablo Segado-Cabezos[3]

[1] Department of Engineering and Technology of Computers, University of Murcia, Spain
{gbernabe,jcuenca,aflores}@um.es
[2] Department of Computing and Systems, University of Murcia, Spain
{josecarlos.canol,domingo}@um.es
[3] Department of Mechanical Engineering, Technical University of Cartagena, Spain
{msaura.sanchez,pablo.segado}@upct.es

Abstract

Computational kinematics is a fundamental tool for the design, simulation, control, optimization and dynamic analysis of multibody systems. The analysis of complex multibody systems and the need for real time solutions requires the development of kinematic and dynamic formulations that reduces computational cost, the selection and efficient use of the most appropriated solvers and the exploiting of all the computer resources using parallel computing techniques. The topological approach based on group equations and natural coordinates reduces the computation time in comparison with well-known global formulations and enables the use of parallelism techniques which can be applied at different levels: simultaneous solution of equations, use of multithreading routines, or a combination of both. This paper studies and compares these topological formulation and parallel techniques to ascertain which combination performs better in two applications. The first application uses dedicated systems for the real time control of small multibody systems, defined by a few number of equations and small linear systems, so shared-memory parallelism in combination with linear algebra routines is analyzed in a small multicore and in Raspberry Pi. The control of a Stewart platform is used as a case study. The second application studies large multibody systems in which the kinematic analysis must be performed several times during the design of multibody systems. A simulator which allows us to control the formulation, the solver, the parallel techniques and size of the problem has been developed and tested in more powerful computational systems with larger multicores and GPU.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: Multibody systems, group equations, Stewart platform, hybrid parallelism

1 Introduction

Multibody systems (MBS) are mechanical systems formed by rigid and flexible bodies which are connected by means of mechanical joints in such a way that there is relative movement

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.041

between their bodies. The study of these relationships is known as kinematic modeling and analysis of the multibody system. Large and complex multibody systems require the use of computational methods to solve their kinematics, which is the basis for the design, simulation, control, optimization and dynamic analysis of the multibody system. All these tasks must either be performed for real time applications, like the simulation, control and dynamic analysis, or repeated a large number of times, for design and optimization processes. In both cases, efficient kinematic formulations are needed together with the selection of the most appropriate solvers and the best exploitation of the computer resources.

With regard to the kinematic modeling and analysis of a MBS, the analyst must select a vector $\boldsymbol{q}$ of coordinates that defines the position and orientation of each body of the MBS in the space. Then, these coordinates are related by means of a nonlinear system of constraint equations $\boldsymbol{\Phi}(\boldsymbol{q}) = 0$. Different formulations use different sets of coordinates, types of constraint equations and methods to solve the kinematic problem. Global formulations, on the one hand, select as many coordinates as needed to define the position of each body independently of the other bodies, and then all these coordinates are related by using the constraint equations associated to each type of mechanical joint in the MBS. These formulations use a large number of coordinates and constraint equations, and their size increases with the complexity of the MBS. On the other hand, topological formulations exploit the topology of the MBS to reduce the dimension of the problem by relating the position of each body with respect to its preceding one, which reduces the size of $\boldsymbol{q}$, or by dividing the MBS into an ordered set of simpler subsystems (known as the kinematic structure of the MBS), whose kinematics can be solved in the specific sequence in which they have been ordered [11].

The topological approach based on Group Equations has proved to be more efficient than global formulations, with reductions in the execution times of up to 50% in the kinematic analysis of 2D and 3D MBS [12]. One drawback of that work is that the results have been obtained for small subsystems, and one advantage is that, depending on the kinematic structure of the MBS, the kinematic analysis of some of the subsystems can be performed independently, which allows us a better exploitation of the computer resources by applying parallelism to reduce the execution time in real-time applications or for highly demanding computations.

The two aforementioned considerations (limitation in the number of subsystems and possibility of exploitation of parallelism) motivate the present work, which aims to develop a simulator for the computational kinematic analysis of multi-body systems to allow us to analyze the efficiency of the group equations approach and to identify the most efficient parallelization strategy, depending on the topology of the system to be analyzed.

The remainder of the paper is organized as follows. Section 2, briefly shows the main ideas of the kinematic analysis based on Group Equations. The case study (Stewart platform) used is introduced in Section 3. The parallel implementations of the simulator based on Group Equations are discussed in Section 4. The experimental results obtained with the application of these parallel implementations are summarized in Section 5. Experiments are conducted in different computational systems (multicore, Raspberry Pi and GPU) to analyze the preferred system depending on our goal: reduction of time or energy consumption, or use for control in real-time or for simulation. The use of various linear algebra libraries is also analyzed. Finally, Section 6 concludes the paper and outlines some possible research directions.

2 Computational Kinematics Based on Group Equations

The topological approach based on Group equations has been shown to be more efficient than a global formulation, using the same type and size of the vector of coordinates $\boldsymbol{q}$. This result

is achieved because, instead of solving the whole system of constraint equations $\boldsymbol{\Phi(q)} = 0$ as the global formulations do, the topological formulation divides the MBS into a number of subsystems, called Structural Groups (SG), which can be modeled and solved by using a reduced set of group coordinates ($\boldsymbol{q_G}$) and the corresponding vector of group constraint equations which, considering a holonomous and scleronomous MBS, can be expressed as $\boldsymbol{\Phi(q_G)} = 0$. In order to solve the kinematics of a certain SG, the vector $\boldsymbol{q_G}$ is split into a vector of independent group coordinates $\boldsymbol{h}$, whose values are known from the kinematic analysis of previous SG, and a vector of dependent group coordinates, $\boldsymbol{\varphi}$, whose values have to be calculated by solving the group constraint equations by means of the iterative Newton-Raphson method, equation 1, where $\boldsymbol{\Phi_\varphi}$ represents the Jacobian matrix of the constraint equations with respect to the vector $\boldsymbol{\varphi}$, and k identifies the number of the iteration.

$$\boldsymbol{\Phi(q_G)} = 0 \rightarrow (\boldsymbol{\Phi_\varphi})_{k-1} \cdot (\boldsymbol{\varphi_k} - \boldsymbol{\varphi_{k-1}}) = -\boldsymbol{\Phi(q_G)}_{k-1} \tag{1}$$

The obtained dependent coordinates are used to solve the linear system of equations for the dependent velocities, $\dot{\boldsymbol{\varphi}}$, which are generated by deriving the constraint equations with respect to time, equation 2, where $\boldsymbol{\Phi_h}$ represents the Jacobian matrix with respect to the independent coordinates.

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{q_G}) = 0 \rightarrow \boldsymbol{\Phi_\varphi} \cdot \dot{\boldsymbol{\varphi}} = -\boldsymbol{\Phi_h} \cdot \dot{\boldsymbol{h}} \tag{2}$$

Finally, the linear system of equations for the dependent accelerations is obtained by deriving the vector of velocity constraints in relation to time, equation 3, where $\dot{\boldsymbol{q}}_{\boldsymbol{G}}$ represents the time derivative of the Jacobian matrix of the constraint equations with respect to the group coordinates, $\boldsymbol{q_G}$.

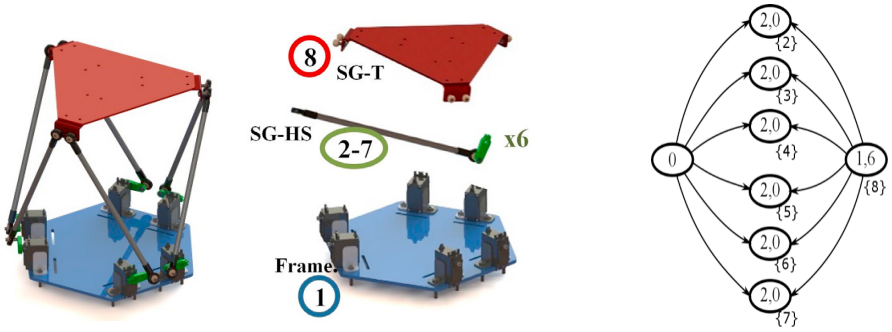
$$\ddot{\boldsymbol{\Phi}}\left(\boldsymbol{q_G}\right) = 0 \rightarrow \boldsymbol{\Phi_\varphi} \cdot \ddot{\boldsymbol{\varphi}} = -\left(\boldsymbol{\Phi_h} \cdot \ddot{\boldsymbol{h}} + \dot{\boldsymbol{\Phi}}_{\boldsymbol{q_G}} \cdot \dot{\boldsymbol{q}}_{\boldsymbol{G}}\right) \tag{3}$$

The kinematic analysis of the whole mechanical system can be carried out by solving each SG in the order given by its kinematic structure, which determines the dependencies between the SGs and the order in which they have to be solved. In addition, the kinematic structure guides the parallelization strategy.

3 Case Study: Stewart Platform

The Stewart platform MBS is used as case study to analyze the application of parallelism techniques for speeding up the kinematic analysis based on Group Equations. Figure 1 shows the structure of a Stewart platform and its decomposition in SGs. The general structure is shown in Figure 1a, and 1b shows the decomposition in SGs. The platform is a parallel robot with six degrees of freedom, composed of six handles, moved by their corresponding rotatory actuators, and which are connected by spherical joints to six sticks (rockers) each one of which is connected to the terminal by cardan joints. The mechanism works through inverse kinematics (input movement to the terminal), and the structural analysis of the mechanism determines seven structural groups: the terminal (SG-T), with 12 dependent coordinates; and six SGs composed of handle-stick pairs (SG-HS), with 15 dependent coordinates each. The terminal element must be solved before the six SG-HS can be solved (Figure 1c). The groups SG-HS are independent, and they can be solved in parallel.

Simulations for the Stewart platform depend on the parameters in Table 1. Simulations can be carried out in four modes: global with dense matrices, global with sparse matrices, based on structural groups with dense matrices, based on structural groups with sparse matrices. The number of non-zero elements is approximately three times the dimension of the matrix, so the



(a) Stewart platform (b) Decomposition into SGs (c) Dependencies between SGs

Figure 1: Stewart platform and its decomposition and dependencies between SGs

matrices are highly sparse. When the matrices are dense, the library used for the solution of linear algebra problems is LAPACK [2], in some cases in the MKL implementation [9] or with MAGMA [1], and when the matrices are sparse, the MA27 solver from HSL is used [8].

parameter	explanation
`tEnd`	a maximum execution time is established
`dt`	time step
`tEnd2`	number of iterations for the position problem
`nSG`	number of structural groups
`nSG-T`	dimension of the SG-T matrix
`nSG-HS`	dimension of the SG-HS matrix

Table 1: Parameters for simulations of the Stewart platform

4 Parallel Implementations of the Method Based on Group Equations

The results obtained when using the Group Equations method are compared with those with the global formulation. In the latter case the matrices to work with are larger, and the matricial routines work with large, sparse matrices. We call the global approach solved with the MKL library GMKL, which works with dense matrices, and the same approach solved with MA27 is called GMA27, which exploits the sparse structure of the matrices. The parallel routines for the Group Equations method are called GEXXX, where XXX correspond to different ways of explotation of the parallelism:

- GEMKL: the parallelism is exploited just by using the multithreading version of MKL. So, the SGs are solved consecutively, using parallelism only to solve each group.
- GEOMP+MKL: OpenMP is used to start threads which work simultaneously in the solution of different groups, one thread per group. Nested parallelism can be used for fuller exploitation of the parallelism [5]. The matrix problems for each group are solved by calling MKL, which can be sequential or multithreading. Two levels of parallelism are exploited, OpenMP and MKL, and different combinations of OpenMP and MKL threads should be considered to find the best combination.

- GEOMP+MA27: OpenMP parallelism is exploited, with calls to the routine MA27 for the solution of the matrix problems for each equation.
- GEMAGMA: GPU parallelism is exploited by solving the matrix problems with MAGMA [1].

A scheme of the Group Equations method is shown in Algorithm 1. The number of external iterations is determined by the time step and the maximum execution time (`dt` and `tEnd` in Table 1). It is normally high, while few iterations are required for the position problems (line 4 in Algorithm 1 and `tEnd2` in Table 1). Parallelism can be exploited with the use of parallel linear algebra routines in the solution of the kinematic problems. These routines can exploit parallelism at different levels, inside a multicore system (MKL) or with calls to Graphic Processing Units (MAGMA). But parallelism can be exploited in the Group Equations approach by simultaneously solving the problems for the SG in the system (`for all` loop in line 3 of Algorithm 1).

Algorithm 1 Scheme of the Group Equations method

```
1: for number of external iterations (tEnd*dt) do
2:     Solve kinematic of terminal (size nSG-T) //MKL parallelism
3:     for all structural components (nSG) do
4:         for number of internal iterations (tEnd2) do
5:             Solve kinematic of structural component (size nSG-HS) //MKL parallelism
6:         end for
7:     end for
8: end for
```

5 Experimental Results

Experiments are conducted for matrices of various sizes and in various computational systems (multicore, Raspberry Pi and GPU). Initial experiments are carried out for the dimensions corresponding to the Stewart platform. These experiments analyze the application of parallelism to the control problem. Larger configurations are considered in order to analyze the use of parallelism for bigger multibody systems and how the parallelization techniques scale.

5.1 Global Formulation versus Group Equations Method

The use of MKL and MA27 (dense and sparse solvers) is compared initially for the global formulation. They correspond to versions GMKL and GMA27. The multithreading parallelism of MKL is also analyzed. A second group of experiments corresponds to the Group Equations approach. The use of OpenMP to exploit shared-memory parallelism is analyzed, as is also the combination of OpenMP parallelism with MKL (sequential and parallel, versions GEMKL and GEOMP+MKL) and MA27 (version GEOMP+MA27). Experiments are carried out in a CPU Intel Core i5-2400 3.10 GHz with 4 cores, without Hyper-Threading, 16 GB RAM, and Windows7 SP1 64 bits.

Table 2 compares the execution times obtained with the global formulation with exploitation of parallelism through MKL (GMKL) and taking advantage of sparsity (GMA27), and with the Group Equations method with shared-memory parallelism using OpenMP in combination with multithreading MKL (GEOMP+MKL) and MA27 (GEOMP+MA27). The version GEMKL corresponds to GEOMP+MKL with 1 OpenMP thread. Experiments are carried out with the number of groups of the Stewart platform (`nSG`=6), and with sizes `nSG-T`=12 and

`nSG-HS`=15 for the smallest problem, and `nSG-T` is fixed to 24 and `nSG-HS` takes the values $\{30, 60, 120, 240, 360, 480, 720\}$ for the other problems. "Total size" represents the size of the matrices for the global formulation. We are comparing the performance of the different approaches, without considering the precision of the results, so the parameters corresponding to the outer and inner iterations are fixed to `tEnd`=200 and time step `dt`=0.01, with a total of 20000 iterations, and `tEnd2`=1 (immediate convergence of the Newton-Raphson method). Each value corresponds to the mean of three executions. For each version exploiting parallelism, experiments were carried out for 1 to 4 threads (the number of cores in the system) and with all the possible combinations of OpenMP and MKL threads. The number of threads which gives the lowest execution time is also shown. The speed-up achieved with respect to GMA27 is shown in Figure 2. Some conclusions can be drawn:

		GMKL		GMA27	GEOMP+MKL		GEOMP+MA27	
Total size	`nSG-HS`	time	th.	time	time	th.×th.	time	th.
102	15	2.52	1	4.19	**0.53**	3×1	0.74	3
204	30	9.11	3	9.02	**1.13**	3×1	1.40	3
384	60	32.43	3	13.45	**2.39**	3×1	2.84	3
744	120	151.94	4	25.29	8.40	3×1	**5.71**	3
1464	240	1127.25	4	48.26	42.33	3×1	**12.23**	3
2184	360	3579.63	4	71.27	147.41	3×1	**18.23**	3
2904	480	7149.04	4	94.19	323.77	3×1	**24.35**	3
4344	720	22840.51	4	140.13	828.97	1×4	**36.59**	3

Table 2: Comparison of the execution times (in seconds) and combination of threads with which they are obtained for the four versions considered, in multicore

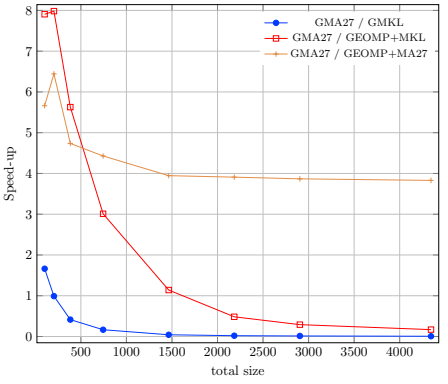


Figure 2: Speed-up of parallel versions in relation with the global formulation approach with MA27. The combinations of threads which give the lowest execution times (Table 2) are considered in all cases

- Multithreading MKL is preferable to MA27 for small sizes (those corresponding to the case study), for which it is preferable not to exploit sparsity.
- When the matrix size increases, the exploitation of sparsity through MA27 is advisable, with a clear advantage over MKL for large matrices. This is due to the $O(n)$ cost of MA27 compared to $O(n^3)$ for MKL without exploitation of the sparsity.
- The Group Equations method clearly outperforms the global formulation, with a speed-up of close to four for the largest problems considered.

- OpenMP parallelism achieves higher speed-ups than MKL from the smallest size, and only for the largest size experimented with (size 4344 in Table 2) is the MKL parallelism preferable.
- The best results are obtained with three OpenMP threads, which is not surprising given that six group equations are solved simultaneously.

The results shown so far are with six equations, in line with the Stewart platform. The influence of the topological structure of the system (the number of equations) is analyzed with experiments with three numbers of structural groups, `nSG`=6, 16, 22, and with four numbers of coordinates, `nSG-HS`=15, 30, 60, 120. `nSG-T` takes value 12 for `nSG-HS`=15 and 24 in the other cases. The variations in the number of groups and the size of the matrices allow us to analyze the application of parallelism for multibody systems larger than our case study. The Group Equations method allows us to exploit parallelism so reducing the execution time with respect to sequential executions. The improvement increases with the number of groups and the number of coordinates. The exploitation of the sparsity is advantageous from between 60 and 120 coordinates (see again Table 2).

`nSG`	`nSG-HS`	MKL	MA27	GEOMP+MKL	GEOMP+MA27
6	15	0.85	1.9	**0.65**	0.76
	30	2.13	3.33	**1.42**	1.79
	60	5.54	7.24	**2.72**	3.13
	120	20.18	15.96	9.61	**6.49**
16	15	2.07	4.86	**0.98**	1.66
	30	4.91	8.61	**2.35**	3.07
	60	14.02	18.69	**4.65**	5.68
	120	52.71	41.43	16.16	**13.23**
22	15	3.00	7.18	**1.35**	2.19
	30	7.15	12.96	**2.62**	4.27
	60	20.77	27.95	**6.53**	8.02
	120	78.89	61.01	23.47	**19.38**

Table 3: Comparison of the execution times for several combinations of number of structural groups (`nSG`) and number of coordinates of each group (`nSG-HS`), for sequential MKL and MA27, and combining them with OpenMP parallelism with four threads

Two-level parallelism can be used for a better exploitation of the parallelism in larger multicore systems [5]. Figure 3 shows the speed-up achieved for various combinations of OpenMP and MKL threads. The experiments were carried out in a system with two hexa-cores Intel Xeon E5-2620 at 2 GHz and 32 GB of RAM. For small problems OpenMP parallelism clearly outperforms MKL, but for larger sizes it is preferable to use hybrid OpenMP+MKL parallelism.

5.2 Raspberry Pi versus Multicore Systems

Raspberry Pi are small, cheap systems with low energy consumption [10], so they are appropriate for control environments, for example, the Stewart platform. Parallel algorithms and libraries are being adapted for these systems [4]. Experiments were carried out in a Raspberry Pi 2 Model B (RP2) and a Raspberry Pi 3 Model B (RP3), with four cores each, and processor architecture ARMv8 quad core 64Bit (RP3) and ARMv7 quad core 32Bit (RP2). MKL is not available, and LAPACK without multithreading is used for linear algebra routines.

Table 4 compares the execution times in RP2 and RP3 with those obtained in the multicore systems in the previous subsection, the Core i5-2400 (i5) and the Xeon E5-2620 (E5). The times correspond to executions with four (RP2, RP3 and i5) or six (E5) OpenMP threads. MKL is

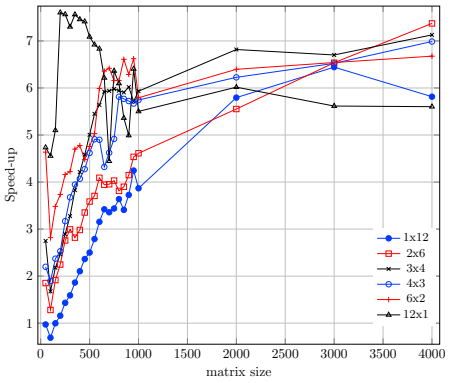


Figure 3: Speed-up with different OpenMP×MKL threads, with 12 cores

used in multicore, and reference LAPACK without parallelism in Raspberry Pi. The numbers of external and internal iterations and the number of groups are 300, 20 and 6, respectively. Better results are obtained in RP3 than in RP2, but the difference decreases with larger problem sizes. Lower times are obtained in multicore, and the difference increases with the problem size. The table also shows the energy consumption with the four systems, considering a Thermal Design Power (TDP) of 4 W for Raspberry Pi, 15 W for the i5 and 95 W for the E5. The lowest values of TDP are obtained with i5, which is a laptop with much lower power consumption than the desktop, and also slower. Raspberry Pi has the lowest TDP for the smallest size (that of the Stewart platform) and so is competitive with general purpose multicore systems for control problems, due to low power consumption, price and size.

		Execution time				Energy consumption			
nSG-T	nSG-HS	RP2	RP3	i5	E5	RP2	RP3	i5	E5
12	18	1.15	0.68	0.40	0.11	4.6	3.4	6.0	10.5
24	36	5.90	4.08	1.46	0.26	23.6	20.4	21.9	24.7
36	54	15.67	11.67	1.94	0.48	62.7	58.4	29.1	45.6
48	72	31.11	26.72	2.95	0.69	124.4	133.6	44.3	65.6
60	90	58.84	53.36	5.14	1.09	235.4	266.8	77.1	103.4
72	108	93.38	89.17	7.17	1.48	373.5	445.9	107.6	140.6
84	126	148.42	137.69	9.64	2.10	593.7	688.5	144.6	199.5
96	144	214.06	204.77	12.85	2.67	856.2	1023.9	192.8	253.7

Table 4: Comparison of the execution times (in seconds) and the energy consumption (multiplying TDP by execution time) in Raspberry Pi 2, Raspberry Pi 3 and two multicore systems, for six structural groups and several sizes of the equations of the terminal (`nSG-T`) and of the equations of the handle-stick pairs (`nSG-HS`)

5.3 Experiments on GPU

In contrast to Raspberry Pi, GPUs are especially useful for large problems [6, 3, 7], and there are efficient linear algebra libraries for GPU [1]. Table 5 compares the execution times obtained with MKL in the two hexa-cores Intel Xeon E5-2620 of subsection 5.1 and MAGMA in a GPU GTX950. The sizes corresponding to the terminal and the handle-stick are divided in three groups. Six structural groups are used and the outer and inner loops are run 10 times each.

		OpenMP×MKL				
nSG-T	nSG-HS	6×2	2×6	MAGMA	6×2/2×6	6×2/MAGMA
12	18	0.006	0.010	0.331	0.60	0.02
24	36	0.008	0.013	0.336	0.62	0.02
36	54	0.010	0.021	0.373	0.48	0.03
48	72	0.025	0.057	0.417	0.44	0.06
60	90	0.034	0.067	0.492	0.51	0.07
72	108	0.047	0.077	0.563	0.61	0.08
400	400	0.721	0.697	3.638	1.03	0.20
600	600	1.819	2.023	5.995	0.90	0.30
800	800	3.211	4.399	8.808	0.73	0.36
1000	1000	6.928	7.376	13.470	0.94	0.51
2000	2000	43.728	40.588	53.038	1.08	0.82
3000	3000	143.072	115.185	138.463	1.24	1.03
4000	4000	328.249	253.709	293.713	1.29	1.12

Table 5: Comparison of the execution times (in seconds) with OpenMP and MKL and MAGMA for several sizes of the equations of the terminal (`nSG-T`) and of the equations of the handle-stick pairs (`nSG-HS`)

OpenMP parallelism is the best option for the small sizes of control problems, but for medium sizes the hybrid OpenMP+MKL parallelism is preferred. The CPU-GPU comparison depends on the characteristics of the CPU and the GPU on hand, but, due to the high latency of the CPU-GPU transfers, exploiting the manycore capacity of the GPU would be advantageous only for large problems. The Thermal Design Power of each hexa-core is 95 W, while for the GTX960 it is 365 W. Thus, the use of GPU is not advisable for this problem for low execution times or power consumption.

6 Conclusions and Future Work

The computational kinematic formulation based on group equations is a topological approach that exploits the kinematic structure of a multi-body system to divide it in several subsystems or structural groups of smaller sizes. The kinematics are solved in a more efficient way than for the whole system, and parallel programming techniques can be applied to solve the subsystems independently.

An analysis of the exploitation of the parallelism for the Group Equations formulation has been carried out on various computational platforms and basic linear algebra libraries. The Stewart platform was initially used to analyze the Group Equations formulation and the exploitation of sparsity and parallelism. The scalability to larger configurations is analyzed by varying the number of groups and coordinates. The general conclusions are:

- Lower execution times are obtained with the Group Equations method in comparison with the global formulation. In addition, the former method facilitates the application of parallelism, which can be combined with efficient linear algebra routines for low execution times.
- The exploitation of the sparsity of the matrices gives lower execution times for large matrices, but for small matrices it is preferable to use dense routines.
- The preferred method depends on the number of structural groups and of coordinates, and the speed-up achieved with respect to the global formulation is between four and eight.
- Raspberry Pi seems to be a good alternative to general purpose multicores for small control problems, with similar times and lower price, power consumption and space.

- The massive parallelism of GPUs is not appropriate for the small sizes of the control problem considered.

The use of other computational libraries, dense and sparse, is being analyzed. Other parallelization strategies need to be analyzed. For example, the equations could be grouped in a number of steps so that the computational load at each step is equally distributed to the cores in the system and multilevel parallelism can be exploited in a different way at each step. Auto-tuning techniques should be included in the routines so that they select automatically the best parallel strategy and library together with the values of some parameters (numbers of threads, number of steps). For large multi-body systems or for distributed combinations of such systems, the use of message-passing parallelism needs to be analyzed.

Acknowledgements

This work was supported by the Spanish MINECO, as well as European Commission FEDER funds, under grant TIN2015-66972-C5-3-R.

References

[1] Emmanuel Agullo, Jim Demmel, Jack Dongarra, Bilel Hadri, Jakub Kurzak, Julien Langou, Hatem Ltaief, Piotr Luszczek, and Stanimire Tomov. Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects. *Journal of Physics: Conference Series*, 180(1), 2009.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. J. Dongarra, J. Du Croz, A. Grenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995.

[3] Gregorio Bernabé, Javier Cuenca, Luis-Pedro García, and Domingo Giménez. Tuning basic linear algebra routines for hybrid CPU+GPU platforms. In *ICCS*, 2014.

[4] Gregorio Bernabé, Raúl Hernndez, and Manuel E. Acacio. Parallel implementations of the 3D fast wavelet transform on a Raspberry Pi 2 cluster. *Journal of Supercomputing*, 2016.

[5] Jesús Cámara, Javier Cuenca, Luis-Pedro García, and Domingo Giménez. Auto-tuned nested parallelism: A way to reduce the execution time of scientific software in NUMA systems. *Parallel Computing*, 40(7):309–327, 2014.

[6] Massimiliano Fatica. Accelerating Linpack with CUDA on heterogenous clusters. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-2, pages 46–51, New York, NY, USA, 2009. ACM.

[7] Azzam Haidar, Chongxiao Cao, Asim YarKhan, Piotr Luszczek, Stanimire Tomov, Khairul Kabir, and Jack Dongarra. Unified development for mixed Multi-GPU and Multi-coprocessor environments using a lightweight runtime environment. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, pages 491–500, 2014.

[8] HSL. A collection of fortran codes for large-scale scientific computation, `http://www.hsl.rl.ac.uk`.

[9] Intel MKL web page. `http://software.intel.com/en-us/intel-mkl/`.

[10] Raspberry Pi. `https://www.raspberrypi.org/`.

[11] M. Saura, A. I. Celdrán, D. Dopico, and J. Cuadrado. Computational structural analysis of planar multibody systems with lower and higher kinematic pairs. *Mechanism and Machine Theory*, 71:79–92, 2014.

[12] M. Saura, P. Segado, B. Muñoz, and D. Dopico. Multibody kinematics. A topological formulation based on structural-group coordinates. In *ECCOMAS Thematic Conference on Multibody Dynamics*, June 2015.