Randomized Parallel Simulation Of Constrained Multibody Systems for VR/Haptic Applications

Antonio Bicchi Lucia Pallottino Marco Bray Pierangelo Perdomi

Centro "E. Piaggio", University of Pisa, Italy

Abstract

In this paper, we consider the problem of efficiently simulating large interconnected mechanical systems. For applications such as haptic rendering of large, complex virtual environments, dynamic simulation software and hardware is still too slow to afford accurate performance in real–time. In particular, mechanisms with closed kinematic chains necessitate solutions to a set of differential equation with algebraic constraints (DAE's), that are often too heavy and stiff to be computed in real–time by present-day single-processor machines. On the other hand, the structure of most state–of–the–art algorithms does not easily lend itself to parallelization. In this paper, we propose and experimentally verify a technique for DAE simulation that profitably uses a degree of randomization to achieve efficient parallelization.

1 Introduction

As the diffusion of Virtual Reality applications becomes more and more pervasive in everyday's life, technical demands to VR systems are also getting more and more challenging at a fast pace. Earlier VR software consisted preminently of graphic environments that could be explored by the user with limited interactions. Initially, environments including moving objects have been incorporated in VR engines using purely kinematic descriptions. However, the realism of movement thus obtained is not quite satisfactory for many applications, and the need for a truly dynamic simulation has been widely recognized. To date, many VR applications include dynamic simulation of more or less complex articulated structures.

The challenge in including dynamic simulation in VR applications is clearly that computations should be (at least) as fast as real time. The development of efficient dynamic simulation algorithms for tree-structured kinematic chains ([11], [9]) and the fast growth of computational power in general purpose machines has helped reaching the goal of simulating rather complex structures in real time ([8], [2], [13], [7], etc.), such as open chains of up to some tens of links. However impressive these advancements have been, they are not yet comparable to what has been achieved in the same time span by graphic software and hardware accelerators. Many complex dynamic systems remain beyond real–time simulation capabilities of present–day software and hardware.

In particular, in this paper we consider simulation of mechanisms with constraints. Multibody systems with constraints arise in a number of different applications, ranging from simple mechanisms with closed kinematic chains, to parallel–actuated platforms, and to robotic telemanipulators in contact with the environment. In applications involving haptic displays, there is very often a closed mechanical chain comprised of the operator and the display device, and a (virtually) closed chain involving the operator and the simulated dynamic environment.

Mathematically, the simulation of constrained mechanical systems calls for the numerical integration of a set of mixed differential and algebraic equations (DAE's). Although much research has been devoted to the study and the efficient numerical solution of DAE's (see e.g. [5], [1]), most such systems are often too heavy and stiff to be computed in real–time by present–day single–processor machines. Clearly, an alternative, cost–effective solution would be the use of parallel and/or distributed computational architectures. This solution is much encouraged by the recently popular trend in high–performance computing, whereby several (order of units or tens) rather sophisticated processors are clustered together on fast buses or net protocols (compare this with earlier massively–parallel philosophies in parallel computing).

Unfortunately, the structure of most state–of–the–art algorithms for simulation of DAE's does not easily lend itself to parallelization or distribution of computation. In this paper, we propose and experimentally verify a technique for DAE simulation that profitably uses a degree of randomization to achieve efficient parallelization. Randomization techniques have proven useful in many domains, such as for instance optimization problems, and, in Robotics, in motion planning for complex systems ([12]). the paper is organized as follows: in 2 we review some of the analytical background involved in simulation of constrained systems; in 3 describe the proposed randomized algorithm, and n 4 we report on some experimental results obtained through application of our proposed algorithm to two case–studies.

2 Background: Simulation of DAE's

Consider a constrained mechanical system comprised of p unconstrained systems, each described by n_i coordinates $\mathbf{q}_i$ with a Lagrangian $L_i(\mathbf{q}_i, \dot{\mathbf{q}}_i) = T_i - U_i$, where

$$T_i = \frac{1}{2}\dot{\mathbf{q}}_i^T \mathbf{B}_i(\mathbf{q}_i)\dot{\mathbf{q}}_i$$

is the kinetic energy and U_i is the potential energy for the i–th system.

Let the p systems be connected through m_h holonomic constraints described by $\mathbf{C}(\mathbf{q}) = 0$, and m_n nonholonomic constraints described by $\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = 0$. We assume that all $m \stackrel{def}{=} m_h + m_n$ constraints are scleronomic (i.e., time-invariant), and that they can be jointly written in Pfaffian form as

$$\left[\frac{\mathbf{A}_h(\mathbf{q})}{\mathbf{A}_n(\mathbf{q})}\right]\dot{\mathbf{q}} \stackrel{def}{=} \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0,$$

with $\mathbf{A}_h \in \mathbb{R}^{m_h,n}$, $\mathbf{A}_n \in \mathbb{R}^{m_n,n}$, and $\mathbf{A} \in \mathbb{R}^{m,n}$, or in control form as

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\mathbf{v},$$

where $\mathbf{S}(\mathbf{q}) \in \mathbb{R}^{n\times p}$ is a matrix whose columns annihilate the constraints, $\mathbf{A}(\mathbf{q})\mathbf{S}(\mathbf{q}) = 0$, and whose rank is maximum (i.e., p) for almost all $\mathbf{q}$.

Equations describing the system are thus obtained as

$$\begin{cases} \mathbf{B}\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{A}^T\lambda = \mathbf{Q}, \\ \mathbf{C}(\mathbf{q}) = 0, \\ \mathbf{A}_n\dot{\mathbf{q}} = 0, \end{cases} \tag{1}$$

where $\mathbf{B} = \mathsf{diag}[\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_p]$, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ contains apparent and gravity forces, and $\mathbf{Q}$ stands for all non conservative forces (including external forces). The unknown lagrangian multiplier vector $\lambda \in \mathbb{R}^m$ can be interpreted as a reaction force capable of enforcing the constraints. In (1), the mixed differential and algebraic nature of the problem is apparent. Differential–Algebraic Equations (DAE's) arise in constrained mechanical systems as well as in many other fields, and have been the subject of intense research in the past two decades (see e.g. [5, 1]). One possible approach to the solution of (1) is to apply an implicit numerical method such as backward differentiation formulae (BDF) or implicit Runge-Kutta (IRK) methods directly to the index-three DAE (1), as first suggested in [10]. However, this approach often leads to ill-conditioning problems, and is not recommended with DAE's of index greater than 1 ([5, 1, 8]). The prevalent approach in the literature appears to be that of reducing the index of the DAE system by differentiating the constraints once or twice. In particular, differentiating holonomic constraints twice, and nonholonomic constraints once, a classical index–one form of constrained dynamics can be obtained as

$$\begin{bmatrix} \mathbf{B} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} - \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \\ -\dot{\mathbf{A}}\dot{\mathbf{q}} \end{bmatrix}. \tag{2}$$

Notice that (2) is perfectly equivalent, in its analytical solutions, to the original (1). However, numerical simulation of the two systems may lead to quite different solutions.

Under a nonsingularity condition on the matrix on the left hand side of (2), simulation can proceed by solving the linear system (2) in the unknown accelerations $\ddot{\mathbf{q}}$ and constraint forces λ. Another approach consists in computing, from $\dot{\mathbf{q}} = \mathbf{A}\mathbf{v}$, the second derivative $\ddot{\mathbf{q}} = \mathbf{A}\dot{\mathbf{v}} + \dot{\mathbf{A}}\mathbf{v}$, substituting in (2) and projecting on the constraint (i.e., multiply by $\mathbf{S}^T(\mathbf{q})$) to get

$$\mathbf{S}^T\mathbf{B}\mathbf{S}\dot{\mathbf{v}} = -\mathbf{z}(\mathbf{q}, \mathbf{v}) + \mathbf{S}^T\mathbf{Q}$$

where $\zeta(\mathbf{q}, \mathbf{v})$ incorporates apparent forces, gravity, and constraint derivatives. Such a linear system is then solved (assuming $\mathbf{S}^T\mathbf{B}\mathbf{S}$ to be nonsingular) to get $\dot{\mathbf{v}}$, and this is integrated along with $\dot{\mathbf{q}} = \mathbf{S}\mathbf{v}$. This second approach to the solution of (2) eliminates the algebraic variables λ, and may reduce substantially the number of equations to be solved. However, thie technique relies on the evaluation of the annihilator $\mathbf{S}(\mathbf{q})$, which may be very complex a task, both analytically and numerically.

Incidentally, let us observe at this point that in both techniqus above, nonsingularity assumptions are tantamount to assuming that the constraint matrix $\mathbf{A}(\mathbf{q})$ is full rank. When such condition is not verified, constraints are redundant, and the system is sometimes termed *hyperstatic*. Determination of lagrangian multipliers is not unique in this case: if the computation of constraint forces is of interest in simulation (as e.g. in the evaluation of dynamically induced mechanical stresses), then it is necessary to use a more refined model for the system. This can be done by introducing a model of compliance (and possibly damping) in the system, modify the potential term accordingly, and redo the analysis. However, if the actual interest in simulation is just to derive the correct motion of the multibody system, than hyperstaticity can be simply resolved by simply suppressing all dependent rows in $\mathbf{A}$. We will henceforth consider the case that the rank of $\mathbf{A}(\mathbf{q})$ (and hence of $\mathbf{S}(\mathbf{q})$) is constant.

Practical integration of (2) will rely on different schemes to numerically approximate derivatives (forward and backward Euler, Runge-Kutta, BDF, etc.). Because at this point we want to abstract from the specific details of the numerical integration algorithm to be employed, we will most simply model effects of numerical approximation of derivatives with noise terms. Specifically, we assume that the numerically approximated velocity $\tilde{\dot{\mathbf{q}}}$ equals the analytic derivative additively perturbed as $\tilde{\dot{\mathbf{q}}} = \dot{\mathbf{q}} + \delta_v$, with perturbation bounded norm as $\|\delta_v\| \leq \Delta_v$. Analogously for accelerations, we set $\tilde{\ddot{\mathbf{q}}} = \ddot{\mathbf{q}} + \delta_a$, with $\|\delta_a\| \leq \Delta_a$. Besides, it has to be considered that initial conditions $(\mathbf{q}(0), \dot{\mathbf{q}}(0))$ used for simulation of (2) can not be assumed in general to match perfectly the constraint conditions. Let E_p and E_v be bounds on the maximum errors in initial conditions such that $\|C(\mathbf{q}(0))\| \leq E_h$, and $\|A_n(\mathbf{q}_0)\dot{\mathbf{q}}(0)\| \leq E_n$.

It is of interest to study the dynamics of the constraint errors, defined as $\mathbf{e}_h = \mathbf{C}(\mathbf{q})$ and $\mathbf{e}_n = \mathbf{A}_n(\mathbf{q})\dot{\mathbf{q}}$. For both the above solution methods of (2), one gets

$$\ddot{\mathbf{e}}_h = d_h,\ \mathbf{e}_h(0) = \mathbf{C}(\mathbf{q}(0)), \quad (3)$$

$$\dot{\mathbf{e}}_n = d_n,\ \mathbf{e}_n(0) = \mathbf{A}(\mathbf{q}(0))\dot{\mathbf{q}}(0) \quad (4)$$

where, for small errors, $d_h \simeq \mathbf{A}_h\delta_a + \frac{\partial \dot{\mathbf{A}}_h\dot{\mathbf{q}}}{\partial \dot{\mathbf{q}}}\delta_v$ and $d_n \simeq \mathbf{A}_n\delta_a + \frac{\partial \dot{\mathbf{A}}_n\dot{\mathbf{q}}}{\partial \dot{\mathbf{q}}}\delta_v$. We let such perturbation terms be bounded in norm as $\|d_h\| \leq D_h$, $\|d_n\| \leq D_n$. Error dynamics are thus linear, with one or two poles in the origin for nonholonomic and holonomic constraints, respectively, and a random forcing term of bounded intensity. Observe that, for given errors of the numerical scheme (δ_a, δ_v), the bounds on the error forcing terms increase with the derivatives of $\mathbf{A}(\mathbf{q})$: in other words, the larger is the curvature of the constraints, the more dramatically numerical noise affects simulation.

While system (4) is barely marginally stable, (3) is unstable. As a consequence, simulation of (2) does not absorb initial condition mismatches, nor damps out numerical perturbations. This leads to violation of constraints, which is more severe in the proximity of highly curved segments of the trajectory. For holonomic constraints, errors in initial conditions increase linearly in time, so that simulated trajectories will typically go adrift with respect to the constraints.

The problems generated by index reduction techniques in DAE's are well known in the literature, and several approaches have been proposed to solve them. These include for instance Baumgarte stabilization ([3]), penalty methods ([14]), and augmented lagrangian techniques ([4]). All these methods share the fundamental characteristic of adding to the original equations a (linear) combination of the constraint errors $\mathbf{e}_h$, $\mathbf{e}_n$, and derivatives thereof. A typical such scheme for stabilizing simulation errors consists in modifying the constrained dynamics equation by adding the term in the box as

$$\begin{bmatrix} \mathbf{B} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} - \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \\ -\dot{\mathbf{A}}\dot{\mathbf{q}} + \boxed{\mathbf{f}(\mathbf{e}_h, \dot{\mathbf{e}}_h, \mathbf{e}_n)} \end{bmatrix}. \quad (5)$$

Selecting for instance

$$\mathbf{f} = \begin{bmatrix} -K_v & 0 \\ 0 & -K_n \end{bmatrix} \begin{bmatrix} \dot{\mathbf{e}}_h \\ \mathbf{e}_n \end{bmatrix} + \begin{bmatrix} \mathbf{K}_p \\ 0 \end{bmatrix} \mathbf{e}_h,$$

the error dynamics become

$$\begin{aligned} \ddot{\mathbf{e}}_h + \mathbf{K}_v\dot{\mathbf{e}}_h + \mathbf{K}_p\mathbf{e}_h &= \mathbf{d}_h \\ \dot{\mathbf{e}}_n + \mathbf{K}_n\mathbf{e}_n &= \mathbf{d}_n. \end{aligned}$$

Hence, by choosing $\mathbf{K}_p, \mathbf{K}_v, \mathbf{K}_n$ as symmetric and positive definite matrices, the constraint is stable. A typical choice for these matrices is to take them diagonal, with $\mathbf{K}_v(i,i) = 2\mathbf{K}_p(i,i)$, so that holonomic constraint errors are critically damped decoupled second order systems. The larger the values of $\mathbf{K}_p(i,i)$ and $\mathbf{K}_n(i,i)$, the faster the transients due to initial condition mismatch converge to zero, and the more effective is the attenuation of the input noise.

3 Randomized Parallel Simulation of D.A.E.'s

The problem with real–time simulation of large/complex constrained mechanical systems by the stabilization methods described in the above section is that, if gain matrices $\mathbf{K}_p, \mathbf{K}_n$ are chosen large enough to have quick convergence and good attenuation – hence small simulation errors – system (5) becomes very stiff. Indeed, the dynamics of the constraint error represents in this case the fastest dynamics in the system, and hence the simulation bottleneck.

Although the literature on numerical integration of ODE's abounds with techniques for solving stiff problems, in VR/Haptic applications we are confronted with an additional problem: simulation has to be done in real–time. Moreover, haptic interactions with the user make it necessary to include in the simulation devices to detect events such as contacts, and allow force exchanges with the haptic probe.

Real–time simulation implies that the computational time must be smaller than the integration step size, hence severely limits the complexity of computations that can be done in each step. Furthermore, the computational time must be predictable, which fact makes adaptable multistep methods with error control unapplicable. García de Jalon and Bayo [8] have a interesting discussion on these problems, and indicate that implicit, stiffly-stable or A-stable fixed multistep methods appear to adapt best to the problem at the present state-of-the-art. Explicit linear multistep methods, on the other hand, are computationally inexpensive and would have the important advantage of avoiding any iterative procedure during the step computations. Unfortunately, their stability has to be guaranteed by the choice of small enough integration steps, which makes these methods hardly suited to stiff problems. We will see below how these problems can be alleviated by the use of our proposed simulation scheme.

Real–time simulation of complex multibody systems represents such a challenge to current computer technology that makes exploitation of computational parallelism compelling. However, how to employ parallelism profitably is not a trivial question. Although parallel computation for ODE's ([6]) and DAE's ([15]) has been considered in the literature, these methods do not apply to real–time simulation of VR/Haptic systems because of the motivations above.

To start with, parallel computing would not be of much help if adapted to constrained mechanical systems in the naïve way: *simulate each unconstrained system on a processor (diastole), collect data and compute reaction forces (systole), and distribute reactions.* Indeed, the systole would coincide with computations involving the fastest dynamics in the system, and hence the slowest compuational part.

The Randomized Parallel Simulation (RPS) algorithm is based on a probabilistic solution of constrained system with randomized perturbations, and on periodic polling of solutions to choose a representative one.

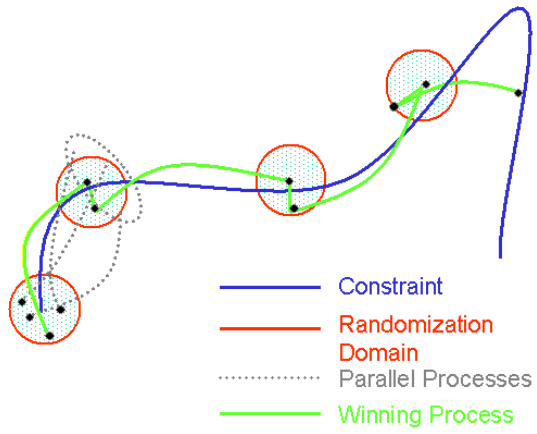

Figure 1: Graphical illustration of the RPS procedure.

The basic idea is extremely simple, and applies to any fixed–step numerical integration method (NIM). Consider a time interval of length T larger than the integration step τ, and proceed as follows (see also fig.1:

1. Initialize N instances of simulation by a NIM of the entire constrained system (2) on each processor, with initial conditions $\mathbf{q}_j(kT^+), j = 1, \dots N$ randomly distributed in a neighborhood of the current value of coordinates $\mathbf{q}(kT^-)$ at time kT.

2. Proceed with simulation as in the nominal case, without introducing constraint stabilization. This achieves much faster simulation (same order as unconstrained simulation) on each processor (diastole);

3. Stop simulation at time $(k+1)T$. Get simulation results $\mathbf{q}_{j,k} \stackrel{def}{=} \mathbf{q}_j(kT+T)$, $\dot{\mathbf{q}}_{j,k} \stackrel{def}{=} \dot{\mathbf{q}}_j(kT+T)$, and residuals $r_{j,k} = \|\mathbf{C}(\mathbf{q}_{j,k})\| + \alpha\|\mathbf{A}_h(\mathbf{q}_{j,k})\dot{\mathbf{q}}_{j,k}\| + \beta\|\mathbf{A}_n(\mathbf{q}_{j,k})\dot{\mathbf{q}}_{j,k}\|$ from each processor. Poll them centrally to elect the one with smallest residual and set

$$\begin{aligned} \mathbf{q}(kT+T)^- &= \mathbf{q}_{\ell,k} \\ \dot{\mathbf{q}}(kT+T)^- &= \dot{\mathbf{q}}_{\ell,k} \\ \ell &= \arg\min_j r_{j,k} \end{aligned}$$

4. Possibly adapt T based on current results.

The principle on which the algorithm is based is a well known result of the theory of differential and difference equations:

Theorem 1 (Generalized Liouville's Theorem) *Consider the one-parameter group induced by the ODE* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, *with* $\mathbf{x} \in \mathbb{R}^n$. *Take a domain* $D \in \mathbb{R}^n$ *of volume* V_D *and its image* $D(t)$ *under the action of the phase flow. Then for the volume of* $\hat{D}$ *it holds*

$$\frac{d}{dt} V_{\hat{D}} = \int_{D(t)} div\, \mathbf{f} dx$$

Applying this theorem to the unperturbed constraint error equation,

$$\ddot{\mathbf{e}} = 0$$

i.e. setting

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{e} \\ \mathbf{x}_2 &= \dot{\mathbf{e}} \end{aligned}$$

and

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} \stackrel{def}{=} \mathbf{M}\mathbf{x},$$

because div $\mathbf{M}\mathbf{x} \equiv 0$, the volume of the initial domain in which initial conditions have been randomly distributed is kept constant. Notice that the polling phase of the algorithm simply consists of passing a $2n$-dimensional vector $(\mathbf{q}_{j,k}, \dot{\mathbf{q}}_{j,k})$ and a scalar residue from each slave processor to the master, finding the smallest in a list of N numbers, evaluating N samples of a randomized distribution of given average, and passing back these samples to the slave processors.

The RPS algorithm above described is extremely simple. However, there are many variants and extensions that can be conceived, whose effects have still to be studied in detail. Some possibilities are described below.

1) Although the volume of the control domain remains constant, its second moment (covariance) may actually grow in time by applying NIM's to unconstrained equations. To correct this, it will be sufficient to simulate the N parallel instances with a *slightly* stabilized algorithm with small $\mathbf{K}_p, \mathbf{K}_n$, so that the eigenvalues of the error dynamics are in the left half complex plane. In this case, div $\mathbf{M}\mathbf{x} =$ Trace $\mathbf{M} < 0$, so that the control volume decreases in time, contracting on the exact solution. This also achieves attenuation of the effects of numerical noise accuracy. Typically much lower values of $\mathbf{K}_p, \mathbf{K}_n$ will suffice than necessary in a single processor implementation, thus avoiding excessive stiffness of equations;

2) Compensation of uncertainties on initial conditions by ramdomization can be extended to numerical noise effects on the error dynamics. This could be achieved by intentionally adding to the right hand sides of (3), (4), a different instance of a zero–average random function for each processor;

3) RPS can be applied in principle with any fixed step NIM. Its advantages however are particularly evident with explicit multistep methods, whereby stiffness reduction is crucial. A variant would be that different processors implement different NIM's (with the same step length), thus introducing another degree of randomization;

4) In the presence of kinematic singularities where $\mathbf{A}(\mathbf{q})$ looses rank, bifurcations typically arise in simulation. The randomization method appears to be inherently suited to tracking multiple solutions in this case. This can be done by setting up a simple mechanisms for detection of near–singularities, which may trigger a multi–hypotheses mode of RPS whereby the poll elects more than one solution to be kept trace of.

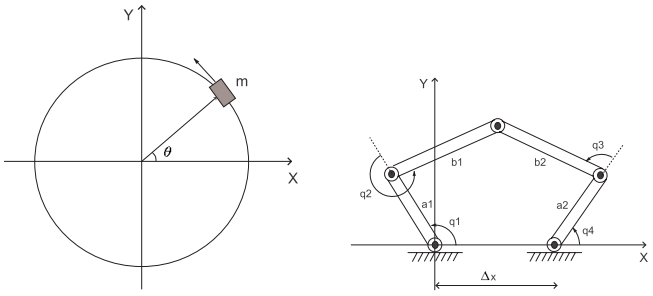

Figure 2: A wheeled cart on a circular rail (left) and a five–bar mechanism (right) used as examples.

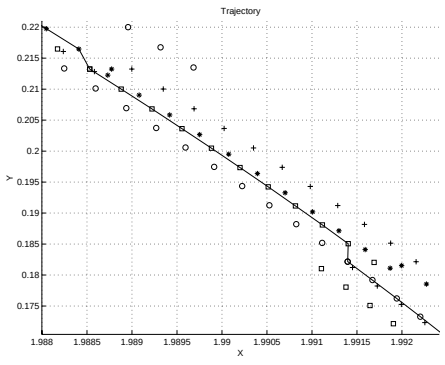

Figure 3: A close-up view of a RPS run for the cart example. Outputs from four processors are represented by a circle, square, cross, and star, respectively. On the the interval $20T < t < 21T$, corresponding to the bottom-right part of the diagram, the "circle" processor is elected, and its trajectory establishes the corresponding segment of the RPS solution (described by the continuous line). At time $21T$ a randomization occurs and simulations are restarted (middle part of the diagram) and, based on results at time $22T$, the "square" processor is elected.

4 Experimental results

We will describe results obtained in the simulation of two different systems, see fig.2. Consider first the simulation of motion of a cart on a circular rail, as depicted on the right in fig.2. We first compare solutions obtained by using a one–step, explicit Euler NIM with fixed step τ. Trajectories simulated by different processors, along with the final resulting trajectory simulated by RPS, are shown in fig.3. The stabilization coefficients have been chosen as $K_v = 2K_p$, so that the constraint dynamics are critically damped. Figure fig.4 shows the effects of increasing K_p on accuracy, for a fixed value of $\tau = 5 \cdot 10^{-4}$. Results of implementations with a single processor, and of RPS with 5 and 10 processors, are reported. From fig.4, it can be observed that the increase in accuracy is not uniform for 5 processors, while it tends to regularize with increasing the number of processors and hence the degree of randomization. The effect of increasing the number of

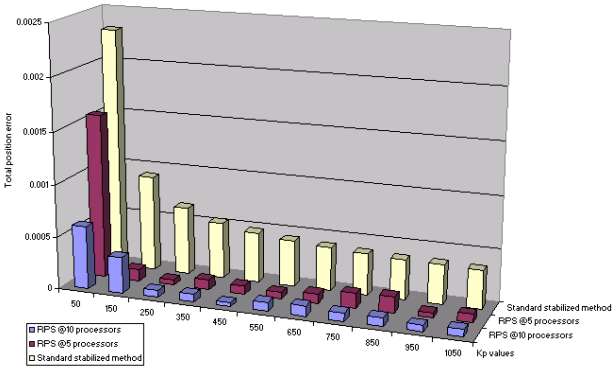

Figure 4: Different accuracies obtained at varying the stabilization parameter K_p with a single processor and with RPS with 5 and 10 processors.

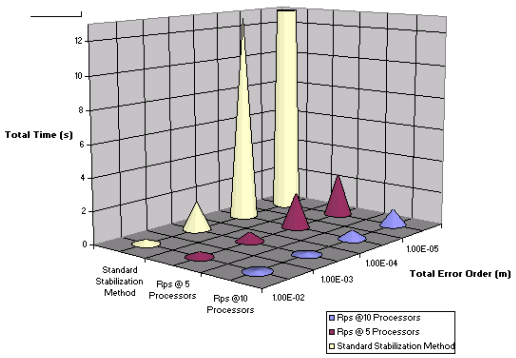

Figure 5: Simulation times for achieving comparable accuracies with one, 5, and 10 processors.

processors above 10 is however rather small in simulations of this example. It can be observed that, to obtain an accuracy higher than $6 \cdot 10^{-3}$ with a single processor, a value of $K_p = 350$ is needed, while it suffices to take $K_p = 150$ with 5 processors, and $K_p = 50$ with 10 processors in parallel. On the other hand, to obtain accuracies of the order of say $1 \cdot 10^{-4}$, the single–processor implementation of this explicit Euler NIM would require much higher K_p, which led to numerical instability for the chosen value of τ. This is not the case with RPS, which achieves that accuracy already with $K_p = 250$. This effect is stressed in fig.5, which has been obtained by choosing, for a given level of desired accuracy, the shortest step τ compatible with stability. From this experiment, it results that the decrease in computational time obtained by RPS is superlinear for high enough accuracies. The rather impressive performances are partially explained by the fact that effects of increasing K_p on algorithmic stability limits is nonlinear, because of the usage in this examples of a simple fixed–step NIM. More analysis is needed to establish performance increase in general cases. Analogous results for the 5–bar case, depicted in

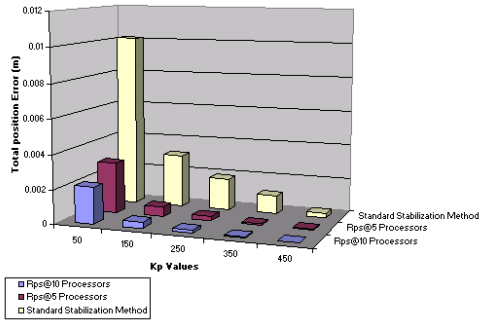

Figure 6: Different accuracies at varying K_p.

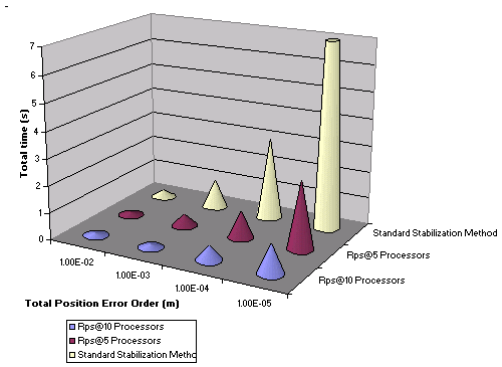

Figure 7: Simulation times for fixed accuracy.

fig.2, are reported in fig.6 and fig.7, which show similar behaviours as observed in the previous example.

5 Conclusion

In this paper, we have described a novel algorithm for parallel (or distributed) simulation of complex mechanical systems. The basic idea is based on a randomization technique, and on concurrent simulations of different instances of the same system. Numerical results obtained confirm the validity of the approach at least when compared with fixed step, explicit numerical integration methods, which are prevalent in real–time problems. Some open questions remain in the theoretical assessment of performance, and in comparison with implicit and variable–step methods. Further study is also necessary to establish optimal choices for several parameters used in RPS, such as e.g. the length of the re–randomization interval, and the radius of the randomization domain.

References

[1] U. M. Ascher and L. R. Petzold. *Computer Methods for ordinary differential equations and differential–algebraic equations.* SIAM, 1998.

[2] D. Baraff. Linear–time simulation using lagrange multipliers. In *Proc. SIGGRAPH*, pages 137–146, 1996.

[3] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Math. Appl. Mech. Engrg.*, 1:1–16, 1972.

[4] E. Bayo, J. García de Jalon, and M. A. Serna. A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer methods in applied mechanics and engineering*, 71:183–195, 1988.

[5] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations.* North Holland, 1989.

[6] K. Burrage. *Parallel and sequential methods for ordinary differential equations.* Oxford University Press, Oxford, UK, 1995.

[7] U. M. Ascher D. K. Pai and P. G. Kry. Forward dynamics algorithms for multibody chains and contact. In *Proc. IEEE Int. Conf. robotics and Automation*, pages 857–862, 2000.

[8] J. García de Jalon and E. bayo. *Kinematic and Dynamic Simulation of multibody Systems: the real time challenge.* Springer–Verlag, 1994.

[9] R. Featherstone. The calculation of robot dynamics using articulated body inertia. *The Int. Journal of Robotic Research*, 2:13–30, 1983.

[10] C. W. Gear. The simultaneous numerical solution of differential–algebraic equations. *IEEE Trans. Circuit Theory*, CT–18:89–95, 1971.

[11] J. M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Trans. Systems, Man, and Cybernetics*, 10:730–736, 1980.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics & Automation*, 12(4):566–580, June 1996.

[13] K. W. Lilly. *Efficient dynamic simulation of robotic mechanisms.* Kluwer, 1993.

[14] P. Lötstedt. On a penalty function method for the simulation of mechanical systems subject to constraints. Technical Report 7919, Royal Institute of Technology, Stockholm, Sweden, 1979.

[15] A. Skjellum, S. Mattison, M. Morari, and L. Peterson. Concurrent dassl: Structure, application, and performance. In *Proc. Fourth Conf. on Hypercube Concurrent Computers and Applications*, Monterey, CA, 1989.