Two Approaches to Speeding Up Dynamics Simulation for a Low Dimension Mechanical System

Stepan Orlov✉, Alexey Kuzin[1], and Nikolay Shabrov[2]

Computer Technologies in Engineering dept.
Peter the Great St. Petersburg Polytechnic University
St. Petersburg, Russian Federation
✉ majorsteve@mail.ru [1] kuzin_aleksei@mail.ru [2] shabrov@rwwws.ru

Abstract. A dynamical model of continuously variable transmission (CVT) is considered. The model is described by ordinary differential equations (ODE) of motion with about 1800 generalized coordinates, and the same number of generalized speeds. Despite the low dimension of the model, the times of numerical simulations of global dynamics are high due to the properties of the system, namely its stiffness. This work presents our activities aimed on the reduction of simulation time. Two approaches are covered. The first one is to parallelize the code computing ODE right-hand side using OpenMP. The other one is to find or develop a faster numerical integration method. The paper presents results of performance tests of the parallelized algorithm on various computer systems and describes scalability problems related to peculiarities of the NUMA architecture. The second approach is illustrated by the results of application of several explicit and implicit numerical methods.

Keywords: Dynamics simulation · Initial value problem · Numerical integration · Parallel algorithm

1 Introduction

In this paper we consider numerical simulations of global dynamics for a model of continuously variable transmission (CVT).

Mathematical model of CVT has been obtained in the framework of Lagrangian mechanics and contains about 1800 generalized coordinates, plus the same number of generalized speeds, so the total problem dimension is about 3600. To numerically simulate dynamics, one has to solve an initial value problem for ordinary differential equations (ODE).

Taking into account today's sizes of numerical problems solved in the fields of structural mechanics, computational fluid dynamics, and others, we have to state that our problem has a low dimension. Nevertheless, simulation running times are high: sequential code takes several hours of CPU time to simulate one second of real time. Speeding up simulations is important since it enables user to apply new analysis types, such as optimization, based on global dynamics

simulations. Therefore, a significant speedup factor is highly desired for practical applications. To achieve this goal, we use two approaches.

First of all, it is possible to parallelize the code implementing the numerical simulation, aiming on modern multi-core or hybrid hardware architectures. However, the scalability of parallelization is very limited due to low dimension of the problem and the heterogeneity of the model.

The paper is organized as follows. Section 2 presents an overview of the model. Section 3 discusses current results of OpenMP-based parallelization of the code. Section 4 illustrates the behavior of various numerical integration methods applied to the problem of CVT dynamics. Section 5 provides a summary of the results obtained and outlines future work.

2 CVT Model Overview

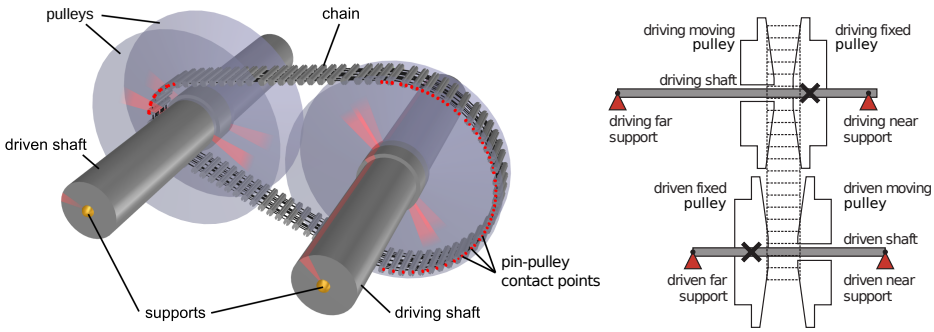

Fig. 1. CVT model general view

The model of CVT includes two elastic shafts, the input and the output one, on nonlinearly elastic supports. There are two pulleys on each shaft, one motionless and one moving (Figure 1). The pulleys have toroidal (almost conical) contact surfaces. There is a chain consisting of *rocker pins* and *plates* (Figure 2). Each pin has two halves that roll over each other during chain motion. End surfaces of pin halves are in contact with the pulleys. The application of clamping forces to pulleys leads to certain chain configuration such that pins are at certain contact radius at each pulley set; the gear ratio can be changed by shifting the moving pulleys along the shafts. The torque is transmitted due to the friction forces at pin-pulley contact points. Mathematical models of CVT parts for global dynamics simulation have to be as simple as possible, while being able to correspond to the reality good enough and represent stressed and deformed state in individual elements, such as pins and links. Therefore, CVT shafts, rocker pins, and plates are modeled as elastic bodies. To describe the state of CVT chain, there are 21 generalized coordinates per link: 10 for each pin half (at each end of pin half axis, there are three coordinates determining its position and two angles determining its small slope; rotation of a pin half about the axis is determined

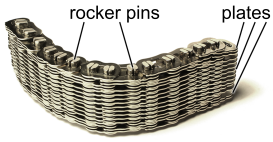

Fig. 2. CVT chain

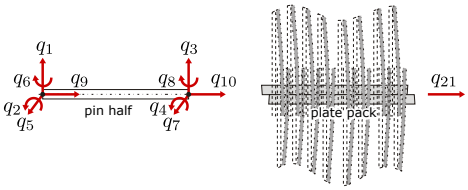

Fig. 3. CVT chain generalized coordinates

by the positions of the neighboring pins) and one to determine the position of pack of plates along pin axis (see Figure 3). Those coordinates fully determine the deformed state of each pin half and each plate in our model of CVT chain.

There are many contact interactions in the CVT model: first of all, we have pin-pulley contact; there are two more types of contact, namely the interactions between pin halves and between a pins and plates (Figure 4).

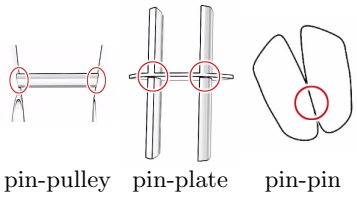

Fig. 4. Types of contact interaction in CVT

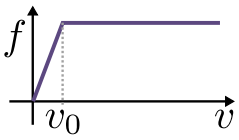

Fig. 5. Friction law for pin-pulley contact

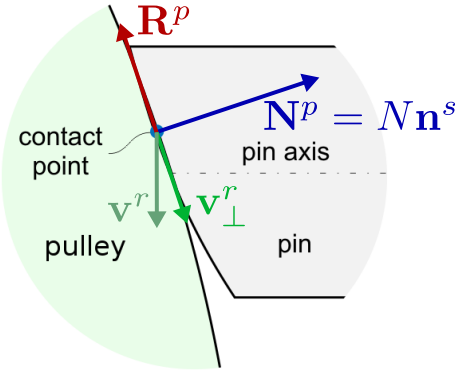

Fig. 6. Pin-pulley contact forces

Special attention should be paid to the contact interaction between pins and pulleys because the torque is transmitted solely due to the friction forces at pin-pulley contact points. The model of contact interaction is physically very simple: for each end surface of a pin half, the interaction is localized at one point; in that point, normal reaction force $\mathbf{N}$ and tangential friction force $\mathbf{R}$ are applied to pin half, and the opposite forces are applied to the pulley (Figure 6). The elastic normal reaction force is computed according to the Hertz' theory [1]; the contact deformation is assumed to be the depth of mutual penetration of contact surfaces, which remain rigid. The friction force is proportional to the normal reaction magnitude and the friction coefficient f. The latter is assumed to be a function of relative tangential speed v, and the dependency corresponds to the Coulomb friction at speeds higher than v_0 (a constant parameter), and to the linearly viscous friction at speeds less than v_0 (Figure 5), so there is no sticking at contact point. The accepted friction law can be interpreted as a kind

of regularization of the Coulomb dry friction; the value of v_0 is quite small, which is a source of numerical stiffness of resulting ODEs.

To resolve contact point kinematics, contact surfaces are locally approximaed with quadratic functions, which allows to determine the right position of contact point on pin half end surface. This is important because the contact point positions ultimately determine the deformed state of pins and plates.

The model of CVT is described in more detail in [2].

Differential equations of motion for the CVT model are obtained in the framework of Lagrangian mechanics, so they have the following form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = \tilde{Q}_k, \quad k = 1, \ldots, n, \tag{1}$$

where t is the time, q_k are generalized coordinates, L is the Lagrangian, $\tilde{Q}_k$ are non-potential applied generalized forces, and n is the number of degrees of freedom.

3 Parallelization With OpenMP

Initially CVT simulation application was rather complex sequential code written in C++ that is why OpenMP was treated as preferable technology of parallelization. OpenMP's important advantage is its relative simplicity when it is applied to existing sequential code. Of course, it does not exclude abilities of the code restyling if necessary.

Due to the problem pecularities the most obvious way of introduction of parallel computing is the parallelization of each step of numerical integration procedure of differential equations of motion. As one can see in Figure 7 most of the time of integration step in sequential application is spent on calculation of the right-hand side of the system of ODEs and foremost for the chain forces calculation, contact forces and time-dependent inertia matrix evaluation and factorization. So these fragments are to be parallelized first.

Current work represents results of parallelization of chain and contact forces evaluation. Due to the chain's periodicity the pins and the links of the chain are natural candidates to parallelization cells. These approaches to the cell definition are both used now, depending on the force being calculated. Therefore the chain forces calculation block is organized as a sequence of two parallel loops (`pragma omp for`) with static scheduling across the links and pins respectively, combined in common parallel section (`pragma omp parallel`). For example, the link-based loop contains evaluation of the forces of the link plates deformation and elastic and damping forces of pin halves deformation. At the other hand, the pin-based loop contains calculation of pin halves interaction of the same pin, such as pin halves contact and friction forces. It also contains evaluation of damping forces in joints.

As it has been mentioned above, the problem has a low dimension so it is possible to use thread-local buffers for the vector of generalized forces of the chain.

Due to usage of thread-local buffers OpenMP threads require few synchronization and the procedure of chain force calculation can be scaled well. But this approach has drawback too. The buffers should be initialized with zeros before each forces calculation takes place and also the results should be gathered into common forces vector after it. In the worst case these steps can not be scaled at all, because the amount of arithmetical operations per thread does not depend on the thread count. Of course, it is worth taking into account that the count of non-zero elements in each thread-local buffer decreases when the thread count increases and, namely, has an order of N/n, where N is the length of the vector of the forces of the chain and n is the count of threads.

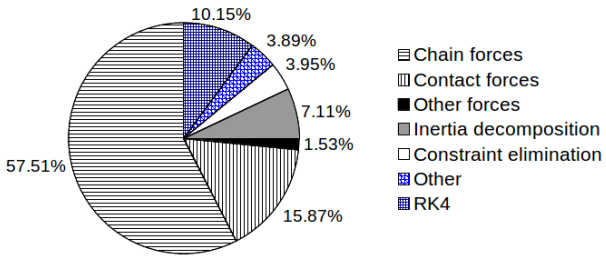

Fig. 7. CPU time consumption in CVT simulation. Sequential code

Therefore, the sequence of chain force calculation at each simulation time point consists of the next steps: thread-local buffers initialization, chain force calculation, gathering of the thread-local results into common force vector. All the steps are being executed in parallel inside `pragma omp parallel` block.

The results of simulation of the same CVT model are presented in Figure 8 and 9. The simulation is performed on *Tesla* computer of Computer Technologies in Engineering dept. (CTM). It consists of 2 NUMA nodes and its hardware and software parameters are presented in Table 1. CPU affinity was managed with environment variable `GOMP_CPU_AFFINITY` so that when $n \leq 6$ only one NUMA node is in use. And only when $n > 6$ the cores of the second node are used. Therefore, the influence of non-uniformity of memory access in NUMA architecture becomes more explicit in this case. The values of Y axis of Figure 8

Table 1. Hardware parameters and OS/GCC versions of computers used in simulations

	Tesla	Tornado
Cores per socket	6	14
NUMA nodes	2	2
CPUs	Intel Xeon X5660 2.80GHz	Intel Xeon E5-2697 v3 2.60GHz
Linux	Ubuntu 12.04.05 LTS	CentOS Linux release 7.0.1406 (Core)
GCC version	4.6.3	4.8.2

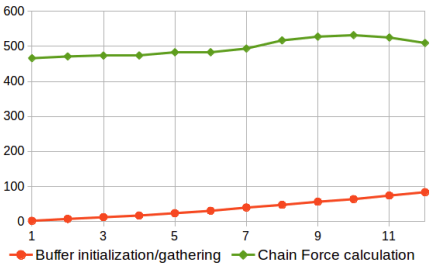

Fig. 8. CPU time

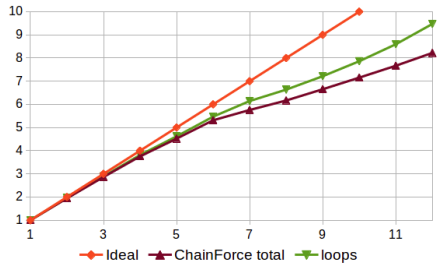

Fig. 9. Relative speedup, chain forces, Tesla

is time and the X axis shows number of threads. The data of the curves in the Figure 8, refered as "CPU time" is calculated in the following way. Let us denote as $T_{i,t}^{(init)}$, $T_{i,t}^{(calc)}$ and $T_{i,t}^{(g)}$ durations of buffer initialization, chain force calculation and gathering respectively, measured in i-th thread with help of `omp_get_wtime` function at simulation time moment t. Then the data points of *Buffer initialization/gathering* and *Chain Force calculation* curves in Figure 8 are evaluated with the formulas:

$$T_1 = \sum_t \sum_{i=1}^{n} \left(T_{i,t}^{(init)} + T_{i,t}^{(g)} \right), \qquad T_2 = \sum_t \sum_{i=1}^{n} T_{i,t}^{(calc)} \tag{2}$$

and represent overall amount of time spent in all threads for buffers initialization/gathering and forces calculation respectively over all simulation steps. This chart demonstrates scalability of the code: in ideal case both curves should be straight horizontal lines, which means that there are no extra CPU time consumption when number of threads grows. One can see that the time of initialization/gathering grows faster than the time of the forces calculation and it will degrade efficiency of the code when the number of threads becomes large. But contribution of initialization/gathering is relatively small in the interval of thread numbers considered from 1 to 12.

The values shown in Figure 8 do not take into account time that has been spent on parallel section creation/closing (`pragma omp parallel` block creation). This fraction can be significant, as it is shown in Figure 9. The speeding up of chain forces parallel calculation with respect to single-thread case is presented there. The Y axis contains ratios of calculation time at 1 core to calculation time at n cores and X axis is the numbers of cores. Both curves refer to the same simulation but use time evaluated in the different way. The time used in curve *ChainForceTotal* evaluation takes into account time consumed on `omp parallel` section. This time has been measured as a difference of `omp_get_wtime` calls after and before `omp parallel` section, therefore it shows real time of calculation. The time used in curve *loops* evaluation is the sum of curves 1 and 2 from Figure 8 divided by corresponding number of threads and characterizes calculation time with `omp parallel` excluded. One can see that

the difference between these curves becomes especially significant when both NUMA nodes are involved ($n > 6$).

This CVT configuration has been also simulated on another machine described in the second column of the Table 1. This is two nodes of computational cluster "Polytechnic RSK Tornado" of Supercomputer Center "Polytechnic" of SPbPU. Further it is referred as *Tornado*. Unlike the simulation on *Tesla* this computation does not use explicit thread binding with `GOMP_CPU_AFFINITY` variable, so the system assigns threads to cores implicitly.

Figure 10 represents chart analogous to Figure 9: relative speed up in dependency on the number of threads. The curve *Tesla* is the same as *ChainForceTotal* in Figure 9, i. e., relative speedup of chain forces calculation on *Tesla*. *Tornado* curve is analogous result obtained at *Tornado*. One can see that in the second case the scalability is much worse and there is no speed up since the level of 11–12 threads is reached. This dependency on architecture of hardware used is subject to future investigation.

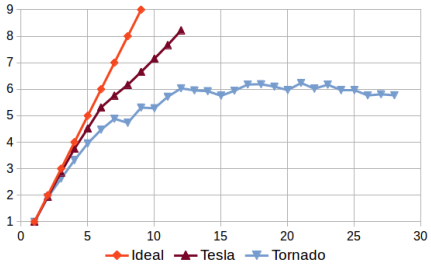

Fig. 10. Relative speedup, chain forces

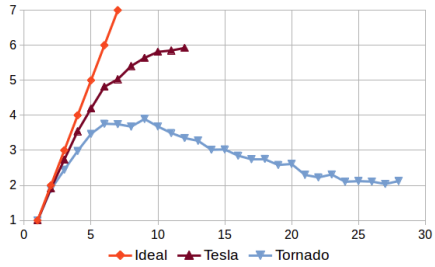

Fig. 11. Relative speedup, contact forces

The calculation of contact forces between chain and pulleys takes place in a separate parallel block and because it is being performed faster relatively to chain forces calculation, the contribution of parallel section creation/closing in this case is more significant. Contact forces calculations on both machines are presented in Figure 11. Meanings of the curves are the same as in Figure 10 with respective replacement of "chain forces" with "contact forces". One can see the loss of performance on *Tornado* with the number of threads growing. Code of contact forces calculation does not contain explicit synchronization structures, therefore the genesis of such slowing down is not obvious and requires additional investigation. Measuring of time of contact forces calculation without taking into account parallel section creation/closing shows much better scalability so the problem might be in rather significant contribution of `pragma omp parallel` code.

4 Investigation of Numerical Methods

Production version of the CVT simulation code has always been using the Runge–Kutta numerical integration scheme of fourth order (RK4) [3] to solve

the initial value problem of CVT dynamics. It is known that the RK4 scheme, as well as other explicit numerical integration schemes, have a step size limitation due to the stability requirement: in general, for a linear system the value $h\lambda$, where h is the step size and λ is an eigenvalue of ODE right-hand side Jacobian, must belong to the stability region, which for an explicit scheme is always a bounded area in the complex plane; for nonlinear ODEs, it is usually the same.

The step size used for CVT numerical simulations with the RK4 scheme has to be quite small, between 10^{-8} and 10^{-7} due to the above mentioned stability limitation. As a consequence, CPU time required for a simulation is high. The analysis of ODE system Jacobian has shown that without friction, maximum eigenvalue magnitude is about 10^6 and corresponds to pin axial vibrations; due to the friction, there are also real negative eigenvalues up to -10^8. Therefore, the ODE system can be considered mildly stiff.

While the actual goal of the entire investigation is to decrease CPU time of simulations, in this section we try to achieve a different goal: find a method that can be applied with significantly larger step sizes than those currently in use. Once such a method is found, its performance has to be further optimized.

Sections below illustrate our attempts to apply different numerical methods to CVT dynamics simulation; we cover explicit methods (sec. 4.2), semi-implicit methods (sec. 4.3), and one completely implicit method (sec. 4.4).

4.1 Numerical Experiment Setup

For each numerical integration scheme, two tests have been done. In the first test, the dependency of step local error on the step size is investigated. The error is computed simply by comparison with the "exact" solution obtained with a very small step size of $2 \cdot 10^{-9}$ using the RK4 scheme. The test can be used, in particular, to verify scheme order of accuracy. In the second test, a dynamics simulation is performed during 0.005 second of real time; a sample history curve is obtained (namely, the axial force in a pin half entering a pulley set) as an evident indicator of the acceptability of numerical results.

To illustrate the impact of nonsmoothness of friction law on the accuracy of numerical results, we also included the results of testing for smooth friction law $f = f_0 \arctan \frac{v}{v_0 f_0}$, where f_0 is the saturation value of friction coefficient.

Sections below present the results of first test for the original and smoothed friction law, and the results of second test for both friction laws and for selected step sizes.

4.2 Explicit Methods

Explicit methods covered in this subsection are the following classical ones.

– Three embedded Dormand–Prince schemes with step size control [3]. The step size control was disabled in test simulation. An embedded scheme provides two solutions of different orders of accuracy at each step, which can be used to control the step size; those orders are encoded in the name of the

scheme. The three schemes are DOPRI45 (orders 4, 5), DOPRI56 (orders 5, 6), and DOPRI78 (orders 7,8).
– Gragg–Bulirsch–Stoer method (GBS) with smoothing step [3]. It is an extrapolation method with the symmetric Gragg's scheme used as the reference scheme. We tried this method with a fixed number of extrapolation stages (2, 4, 6) and the harmonic step size sequence (the schemes are referred to as GBS2, GBS4, GBS6 below).
– Extrapolated explicit Euler scheme, with 2 extrapolation stages and the harmonic step size sequence (referred to as Euler-x2 below).

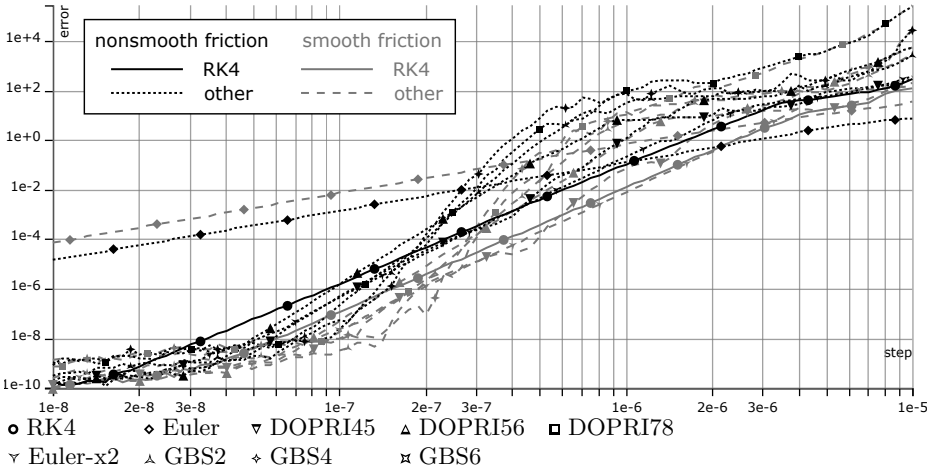

Fig. 12. Step local error for explicit methods

The step local error test (Figure 12) shows that the local error is generally less for smooth friction law; further data processing also indicates that the local error behaves according to scheme order only in a limited step size ranges, different for different schemes; some schemes (DOPRI45, DOPRI78, GBS4, GBS6) do not show the expected behavior at all, although they do in tests with simple ODEs.

The dynamics test (Figure 13) confirms that all explicit schemes considered have severe step size limitation that is about 10^{-7} for nonsmooth friction law and schemes GBS2, DOPRI56, and is less for other schemes; for smooth friction law, the limit is higher yet it is less than $5 \cdot 10^{-7}$. We can also conclude that low order schemes (2–4) are preferrable in model with non-smooth friction low; in model with smooth friction law, higher order schemes may be preferrable.

4.3 Semi-implicit Methods

There was a hope that a W-method [4] is capable of producing acceptable numerical solution at steps much greater than 10^{-7}, because those methods generally have better stability properties than explicit ones. However, in our case all W-methods tested failed for some reason, though they worked good in tests with simple ODEs.

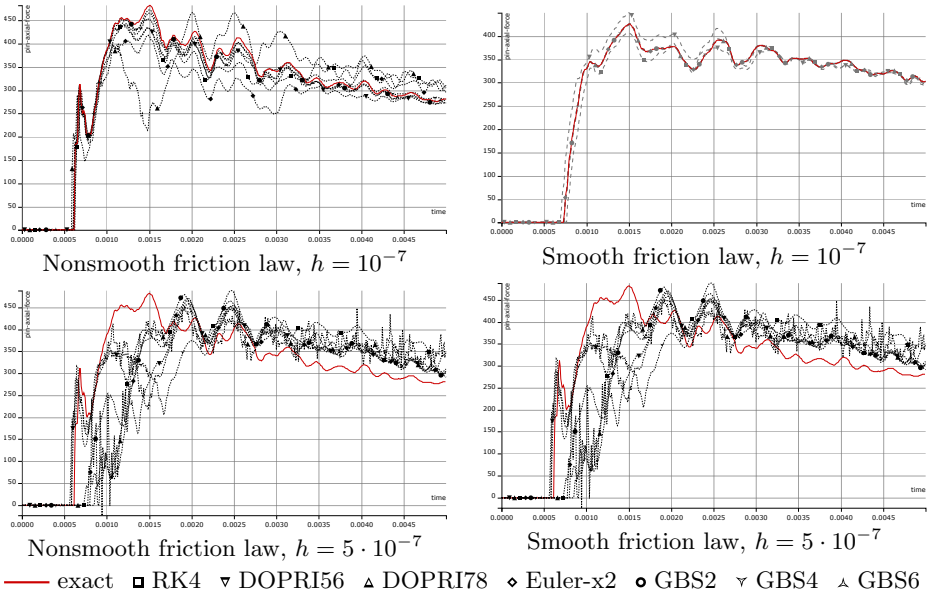

Fig. 13. Sample curves for explicit methods

The schemes considered in this subsection are W24 [4] and the W1 method extrapolated according to the Richardson's procedure [3]. The W1 scheme is as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}(t_k, \mathbf{x}_k) + hd\mathbf{A}(\mathbf{x}_{k+1} - \mathbf{x}_k), \tag{3}$$

where $\mathbf{x}$ is the numerical solution vector, the subscript k denotes the step number, h is the step size, t is the time, $\mathbf{f}$ is the ODE right-hand side vector, d is a parameter (usually between 0 and 1), and $\mathbf{A}$ is the matrix approximating the ODE system Jacobian $D\mathbf{f}/D\mathbf{x}$. W-methods are attractive compared to Rosenbrock methods [5] due to the ability to keep $\mathbf{A}$ constant during many time steps, thus eliminating the necessity to compute it and factorize the matrix $\mathbf{I} - hd\mathbf{A}$ at each time step.

Figure 14 shows that all schemes tested have much greater local step error than explicit schemes. The expected order of schemes is observed at step sizes less than 10^{-7}; the higher the order, the less the range in which scheme order is obeyed.

Figure 15 shows that all W-method schemes produce inacceptable solution even at step 10^{-7}. We have to conclude that they didn't work in our case.

4.4 Trapezoidal Rule Method

Among many implicit methods, we chose the trapezoidal rule (2-nd order scheme):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}[\mathbf{f}(t_k, \mathbf{x}_k) + \mathbf{f}(t_k + h, \mathbf{x}_{k+1})], \tag{4}$$

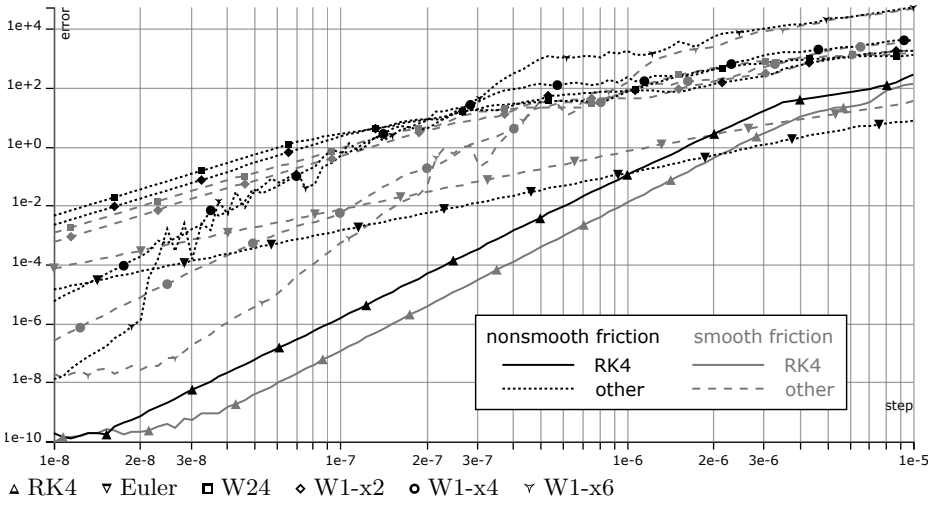

Fig. 14. Step local error for semi-implicit W-methods

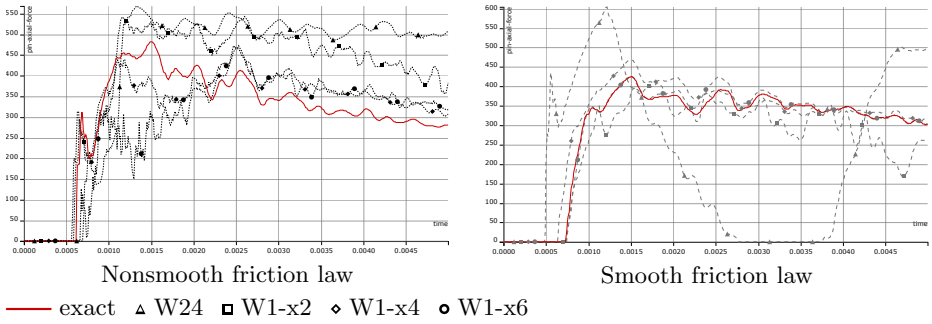

Fig. 15. Sample curves for W-methods, $h = 10^{-7}$

Figure 16 shows the step local error for the trapezoidal rule. Notice that it is less than for any other scheme tested at steps greater than $4 \cdot 10^{-7}$.

Sample curve shown in Figure 17 is obtained at step size $2 \cdot 10^{-6}$ and practically coincides with the exact solution. It is possible to use larger step sizes, but only with step size control because the Newton's method used at a time step may fail to converge.

5 Conclusions and Future Work

The paper considers two approaches for speeding up the numerical integration of about 3600 ODEs of CVT dynamics. The ODE right-hand side is quite numerically expensive, and the ODE system is mildly stiff.

The first approach is to parallelize the computation of ODE right-hand side. Usage of pins and chain links as cells of parallelization allows to calculate forces

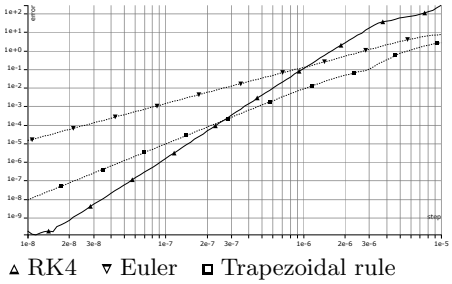

Fig. 16. Step local error for trapezoidal rule, nonsmooth friction law

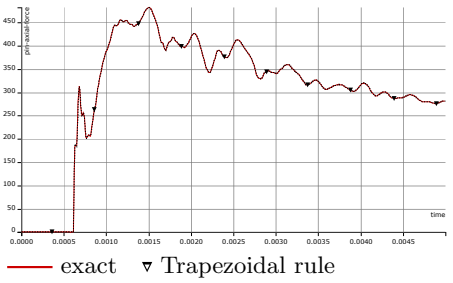

Fig. 17. Sample curves for trapezoidal rule, $h = 2 \cdot 10^{-6}$, nonsmooth friction law

in the chain in natural way. But scalability of present implementation strongly depends on parameters of machine used and may be rather poor. The cause of it is the goal of future investigation.

The second approach is to find a numerical method faster than RK4 currently used in the production version of CVT software. The investigation has shown that traditional explicit numerical integration schemes and W-methods don't work in our case. Implicit methods give good results; however, to make those methods run faster than RK4, additional efforts are required: for example, ODE right-hand side Jacobian could be computed much faster but it requires tedious programming (the idea is to combine the approach presented in [6] with the decomposition of the ODE right-hand side into a sum and providing faster code for the Jacobian of contact forces).

Future plans include performance improvements for implicit schemes. In addition, we are planning to test so called stabilized explicit Runge–Kutta methods because they have not been covered in this research, while seem to be quite apropriate for the case of Jacobian eigenvalues that we really have.

References

1. Johnson, K.: Contact mechanics. Cambridge University Press, Cambridge (1987).
2. Shabrov, N., Ispolov, Yu., Orlov, S.: Simulations of continuously variable transmission dynamics. ZAMM 94 (11), 917–922 (2014).
3. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I (2Nd Revised. Ed.): Nonstiff Problems. Springer-Verlag, New York, Inc (1993).
4. Steihaug, T., Wolfbrandt, A.: An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations. Math. Comp. 33, 521–534 (1979).
5. Rosenbrock, H.H.: Some general implicit processes for the numerical solution of differential equations. Comput. J. 5, 329–330 (1963).
6. Ypma, T.J.: Efficient estimation of sparse Jacobian matrices by differences. Journal of Computational and Applied Mathematics, vol. 18, issue 1, 17–28 (1987).