IASTED Int. Conf. on Robotics and Applications (IASTED RA'99), Santa Barbara, California, USA, Oct. 28-30, 1999.

MANIPULATING DEFORMABLE LINEAR OBJECTS – EFFICIENT SIMULATION OF THE WORKPIECE BEHAVIOR –

AXEL REMDE, DOMINIK HENRICH, SANDER KARL, and HEINZ WÖRN
Institute for Process Control and Robotics (IPR)
Universität Karlsruhe (TH), D-76128 Karlsruhe, Germany
e-mail: [Remde, dHenrich, Sander, Woern]@ira.uka.de
http://wwwipr.ira.uka.de/~paro/

Abstract

In this paper, we investigate the efficient simulation of deformable linear objects. Based on the state of the art, we extend the principle of minimizing the potential energy by considering plastic deformation and describe a novel approach for treating workpiece dynamics. The major influence factors on precision and computation time are identified and investigated experimentally. Finally, we discuss the usage of parallel processing in order to reduce the computation time.

Keywords: Deformable linear object, modeling, simulation, optimization, parallel computing

1 INTRODUCTION

When regarding industrial goods, it is obvious that almost any of them contain deformable workpieces, like seals, insulations, springs and hoses. An important subset are deformable linear objects (DLOs), as hoses, leaf springs, cables or ropes. Accordingly, the challenging problem of handling those objects with robot manipulators has been investigated for several years.

Besides the development of special-purpose grippers and the handling based on sensor information, the computation of the workpiece shape has been addressed. Zheng et al. perform an off-line computation of the gripper trajectory for inserting a flexible beam into a hole and succeed in performing the task without the additional usage of sensors [8]. Though we suppose that the successful and robust execution of handling operations requires additional sensor information in most cases (see Remde et al. [4]), computing the workpiece behavior is useful for diverse reasons. First, having knowledge about the workpiece behavior is necessary for developing and comparing handling strategies. Second, simulation can be used for the off-line generation of coarse gripper trajectories, even if additional sensors may be required for reliable and robust execution.

For these purposes, Hirai et al. give an algorithm for the 2D computation of elastically deformable thin parts based on the principle of minimal potential energy [1]. For DLOs, Wakamatsu et al. extend this approach to 3D-computation [5] and to the consideration of dynamics based on HAMILTON'S principle [6].

These works show the principle possibility of simulating the behavior of DLOs. However, in practical applications it is generally desired to do the simulation efficiently, i.e., to compute the workpiece shape with high precision in a short time. Therefore, we consider the following questions: How should the basic physical approach be formulated with respect to an efficient computation (Section 2)? How are precision and computational effort influenced by parameters and computation algorithms (Section 3)? When is it helpful to employ parallel processing to accelerate the simulation? None of the works mentioned above supply information about these points. However, because the fundamental approach is similar, we suppose that most of the results presented here hold true for these works, too.

2 APPROACH

Our consideration is based on the following main assumptions which hold true for many industrial problems.

The workpiece is *linear*, i.e., the cross section can be neglected. The bending rigidity is independent of the applied force direction, i.e., the cross section of the workpiece is circular. Additionally, the length of the workpiece is constant, i.e., linear extension is not considered. [1]

The behavior of the workpiece is *elastic-perfectly plastic*, i.e., the stress-strain relation can be expressed by the function shown in Figure 1. Plastic deformation is assumed to occur for the entire cross-section simultaneously, i.e., internal stress is neglected. [2] Hardness increase is notregarded.

As stated by Wakamatsu [6], the workpiece behavior is often different for slow (quasi-static) and quick (dynamic) manipulation. Therefore, it should be possible to consider the dynamic workpiece behavior. For the dynamic computation, we assume a *small computation time interval*, i.e.,

[1] However, considering linear extension is no principal problem, see Wakamatsu [4]

[2] In reality, the beginning of plastic deformation depends on the distance from the neutral axis.

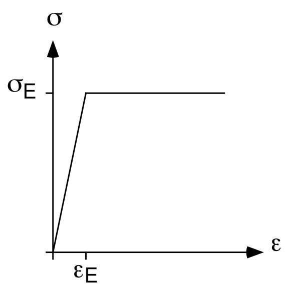

Figure 1: Stress-strain diagram for an elastic-perfectly plastic material (ε: strain, σ: stress)

for one time interval the acceleration of each mass element is constant. As in many works, friction is neglected.

2.1 PHYSICAL PRINCIPLE

According to the fundamental physical principle of minimal potential energy, the potential energy W_{pot} of the workpiece reaches to a minimum in any stable state. When neglecting linear extension, potential energy due to bending and torsion must be considered. Thus, the equation

$$W_{\mathrm{pot}} = \int_0^L (W'_{\mathrm{grav}} + W'_{\mathrm{bend}} + W'_{\mathrm{twist}})\, ds \overset{!}{=} \min \tag{1}$$

has to be solved. In Eqn. 1, L is the length of the workpiece, $s \in [0, \mathrm{L}]$ is the curve length measured along the workpiece. W'_{grav}, W'_{bend} and W'_{twist} is the potential energy caused by gravity, bending and twisting (per length), respectively. For the quasi-static case, this formulation is straight-forward and the computation can be performed as described by Wakamatsu [5].

If the dynamic behavior of the workpiece shall be regarded, the extension of this approach leads to a computation based on HAMILTON'S principle. However, we propose a different computation method which we suppose to be simpler and more robust. We start with the well-known relation [3]

$$\boldsymbol{a} = -\frac{1}{\Delta m} \nabla W_{\mathrm{pot}} \tag{2}$$

for a single mass element Δm. Being Δt the time step of the computation, $\boldsymbol{x} = \boldsymbol{x}(t)$ its (unknown) position at time t, and $\boldsymbol{x}_0$ and $\boldsymbol{v}_0$ its (known) position and velocity at time $t - \Delta t$, the relation

$$\nabla W_{\mathrm{pot}} + \frac{2\,\Delta m}{\Delta t^2}\big(\boldsymbol{x} - (\boldsymbol{x}_0 + \boldsymbol{v}_0 \Delta t)\big) \overset{!}{=} 0 \tag{3a}$$

can be obtained by simple considerations. This condition is fulfilled if the function U

$$U = W_{\mathrm{pot}} + \Delta m \left(\frac{\boldsymbol{x} - (\boldsymbol{x}_0 + \boldsymbol{v}_0 \Delta t)}{\Delta t} \right)^2 \tag{3b}$$

reaches to a minimum.[4] By comparing Eqn. 3b with Eqn. 1, it is found that the dynamic position $\boldsymbol{x}$ can be computed from $\boldsymbol{x}_0$ by just adding the term

$$W'_{\mathrm{dyn}} = \rho A \left(\frac{\boldsymbol{x} - (\boldsymbol{x}_0 + \boldsymbol{v}_0 \Delta t)}{\Delta t} \right)^2$$

to the integrand in Eqn. 1, with ρ and A being the density and the cross-section of the workpiece. Thus, the equation

$$U = \int_0^L (W'_{\mathrm{grav}} + W'_{\mathrm{bend}} + W'_{\mathrm{twist}} + W'_{\mathrm{dyn}})\, ds \overset{!}{=} \min \tag{4}$$

has to be solved instead of Eqn. 1. [5]

For $\Delta t \to 0$, the solution given above and an numerical integration of the acceleration both converge against the correct solution. However, for any realistic $\Delta t > 0$, dynamic workpiece oscillations increase in time when integrating the acceleration, while they decrease with the approach given here. Therefore, the numerical value of Δt is less critical in this approach. See Karl [2] for details.

2.2 COMPUTATION

The computation of the workpiece shape by evaluating Eqn. 4 (or Eqn. 1) is performed in the following way. First, a vector $\boldsymbol{q}(s)$ of functions $q_i(s)$ is required which uniquely determines the workpiece shape and can be used for computing the potential energy W_{pot}. Second, each function $q_i(s)$ is expanded into a series with N_c terms. Third, a discrete optimization algorithm is used for simultaneously determining a set of coefficients c_i for each series which fulfills Eqn. 4. By these steps, the problem given in Eqn. 4 is transformed into the problem

$$U = f(c_0, c_1, \ldots c_{3N_c - 1}) \overset{!}{=} \min \tag{5}$$

with f being the sum of the energy terms according to Eqn. 4, and $3N_c$ being the total number of coefficients. [6]

First, it is necessary to choose an appropriate object representation. Wakamatsu [5,6] uses Eulerian angles for describing the local coordinate system at each workpiece point with respect to a global coordinate system

$$\boldsymbol{q}(s) = \left[\varphi(s), \theta(s), \psi(s)\right]^{\mathrm{T}}$$

and expresses curvature κ and torsion τ by the Eulerian angles and their derivatives with respect to s. This is a good representation if elastic deformation is considered. However, it is not favorable for describing plastic deformation since the elastic and plastic portion of curvature and torsion are distributed over all of the Eulerian angles. Please note that the occurrence of plastic deformation means a change of the stressfree object shape. Thus, the plastic deformation computed in each step must be stored and considered in all subsequent steps while the elastic deformation is computed newly in each step. Storing just the plastic portion of the deformation is not possible if Eulerian angles are used to represent the object shape.

---

[3] Generally, we refer to positions as $\boldsymbol{x}$, velocities as $\boldsymbol{v}$, and accelerations as $\boldsymbol{a}$.

[4] Please note that Eqn. 3a is the gradient of Eqn. 3b with respect to $\boldsymbol{x}$.

[5] Please note that W'_{dyn} represents the dynamic behavior in the minimization problem but is not the kinetic energy of a mass element!

[6] The total number of coefficients is $3N_c$ because 3 functions are required for representing the workpiece shape in space. See also below!

Therefore, we propose to use curvature κ and torsion τ directly for the representation.

For representing the DLO shape in space unequivocally, a third function is required. For completing the representation, we choose the local direction of curvature $\delta(s)$. Thus, the workpiece representation is given by

$$\boldsymbol{q}(s) = \left[\kappa(s), \tau(s), \delta(s)\right]^{\mathrm{T}}$$

Using this representation, the separation of curvature and torsion into their elastic and plastic portions can be performed by simple geometric considerations. Being κ_e and τ_e the elastic portions of curvature and torsion, respectively, W'_{bend} and W'_{twist} can be computed easily by

$$W'_{\text{bend}} = \frac{1}{2} R_{\text{bend}} \kappa_e^2$$

$$W'_{\text{twist}} = \frac{1}{2} R_{\text{twist}} \tau_e^2$$

with R_{bend} and R_{twist} as bending and twisting rigidity. The transformation to a global Cartesian coordinate system can be performed by simple trigonometric considerations. More information about the consideration of plastic deformation is given in Karl [2].

Second, a series expansion is used for approximating $\boldsymbol{q}(s)$. The type of series and the number N_c of series terms are of major influence on both computation time and precision. Generally, the series should be a good approximation of the $q_i(s)$ with N_c being as small as possible. Because the workpiece may principally take any shape, it is not possible to find a series which guarantees a good approximation under all circumstances. A typical problem of almost any series is the representation of sharp bendings. Hirai, Wakamatsu et al. succeed in using Fourier series. An interesting alternative are Chebyshev polynomials which are often a good choice for the approximation of unknown functions (See Press et al. [3]). The series expansion Q of function q with N_c Chebyshev polynomials can be expressed by

$$Q(s) = \sum_{j=0}^{N_c-1} c_j \cos\left(j \arccos(\frac{2s}{L} - 1)\right).$$

We generally presume

$$Q(s) \equiv q(s),$$

for $N_c \to \infty$, i.e., the accuracy is increased with N_c. However, the effort for solving the minimization problem is increased with N_c, too.

Third, a discrete optimization algorithm is required for determining a set of coefficients c_j which fulfills Eqn. 5. Generally, any algorithm for non-linear optimization in multidimensions could be employed. We investigate the Downhill Simplex (DS) algorithm and the DAVIDON-FLETCHER-POWELL (DFP) algorithm as Quasi-NEWTON Method. The major advantages and disadvantages of the DFP algorithm hold true for all kinds of gradient methods (see Press et al. [3]). On the one hand, the Downhill Simplex algorithm is easy to implement. On the other hand, gradient methods are generally more powerful.

3 SIMULATION EXPERIMENTS

3.1 PRECISION

For investigating the influence of the series expansion, the two benchmark problems (a) shown in Figure 2 are used:

First, a copper wire of length $L = 1$ m and diameter $d = 1$ mm is fixed in a Cartesian world coordinate system at $x = 0$ m, $z = 1$ m with horizontal orientation at one end and bends due to gravity (Benchmark A). Second, an additional load of 1 kg is mounted at the free end to increase the bending (Benchmark B). [7]

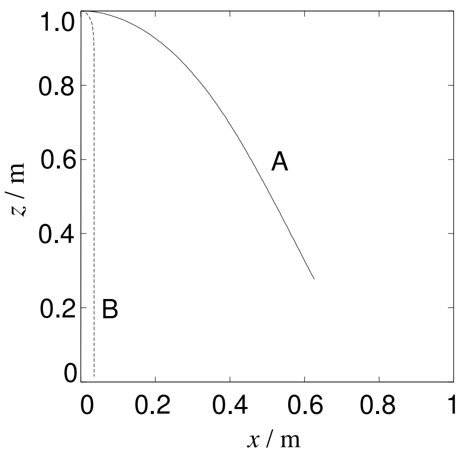

Figure 2: Two Benchmarks A and B for investigating the computation precision

Figure 3 shows the maximum error Δx_{max} of the computed shape as a function of the number of series terms for Chebyshev polynomials and Fourier series, respectively. As reference, a computation with $N_{c,\,ref} = 32$ series terms is used. The number of elements for the discretization of the wire length is $N_L = 64$.

As expected, the computed shape converges against the reference shape for $N_c \to N_{c,\,ref}$. The deviation increases with the bending of the wire. However, the number of series terms required for obtaining good accuracy is considerably lower for Chebyshev terms. In this case, the maximum accuracy (given by the computational accuracy, dashed horizontal line in the figures) is obtained for $N_c \approx 10$ even for sharp bendings. If Fourier series are used, the number of required coefficients is considerably higher. Thus, we suppose that Chebychev polynomials converge faster for typical cases.

Besides the number of series terms, the precision depends on the discretisation Δs (given by object length L and number of curve elements N_L) of the object for computing the energy integral given in Eqn. 4. For benchmark A described above, the maximal error is shown as a

[7] In reality, the deformation is mainly elastic for benchmark A while it is plastic for benchmark B. However, we consider the deformation to be purely elastic in both cases in order to have equal conditions. The plastic deformation is not relevant for the question considered here.

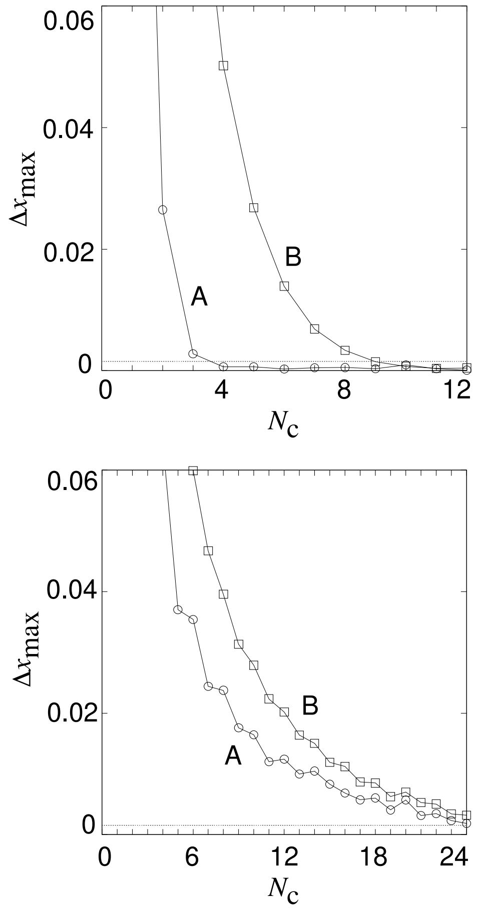

Figure 3: Maximal deviation Δx_{max} between computed and reference workpiece shape as a function of the number N_c of series terms for Chebyshev polynomials (above) and Fourier series (below)

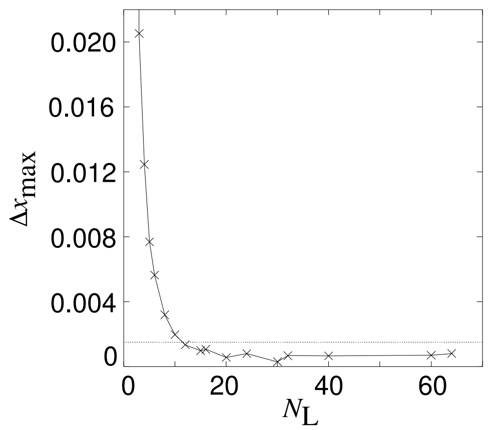

Figure 4: Maximal deviation Δx_{max} between computed workpiece shape and reference as a function of the number N_L of curve elements for benchmark A.

function of N_L in Figure 4 (for benchmark B, a similar result is obtained). The reference shape is computed with $N_{L,ref} = 960$. In this example, Chebychev series with $N_c = 16$ terms are used for approximating the q_i. For $N_L = 15$, the maximal accuracy is obtained. A further increase of N_L does not improve the accuracy. [8]

Please note that we assign one node to every discrete workpiece element. The maximum deviation Δx_{max} considered here is the maximum deviation between the computed node positions and the node positions of the reference computation. Between the nodes, we approximate the object shape by circular arcs. Here, the difference to the correct shape may be higher.

3.2 COMPUTATION TIME

The effort required for computing the workpiece shape is mainly determined by the combination of following factors: Number of curve elements N_L, number of series terms N_c, and optimization algorithm for computing the energy minimum according to Eqn. 5.

For investigating the computational effort, the benchmark C shown in Figure 5 is used: The copper wire described above is gripped at one end point with gripper position $x = 0$ m, $z = 1$ m. Starting with a horizontal orientation, the gripper is rotated by 180° around the y-axis with a stepsize of 10° and back into the initial position. In each experiment, the total time for simulating the 36 object positions is measured. The computation is performed on a 133 MHz Pentium PC with 64 Mbytes RAM

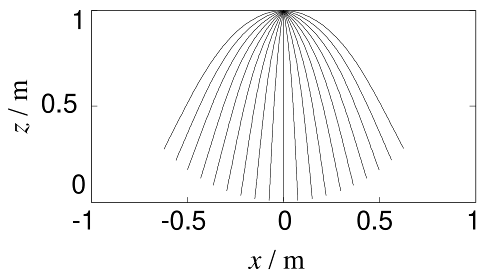

Figure 5: Benchmark C for investigating the computation time

using LINUX as operating system.

For both optimization algorithms, Figure 6 shows the computation time as a function of the number of curve elements N_L with the number of series terms being $N_c = 8$. In both cases, the computation time increases approximately linearly with N_L. Compared to the Downhill Simplex algorithm, the DFP algorithm requires more evaluations of the energy integral in each iteration. Therefore, even a small difference in the number of iterations has a significant influence on the total computation time. Thus, the computation time as a function of N_L is less smooth for the DFP algorithm than for the Downhill Simplex algorithm. [9]

---

[8] The software stores all numbers as 64 Bit floating points (standard doubles for PCs).

[9] Please note that the computation time is not generally smaller for the downhill simplex algorithm. This also depends on the number N_c of series terms.

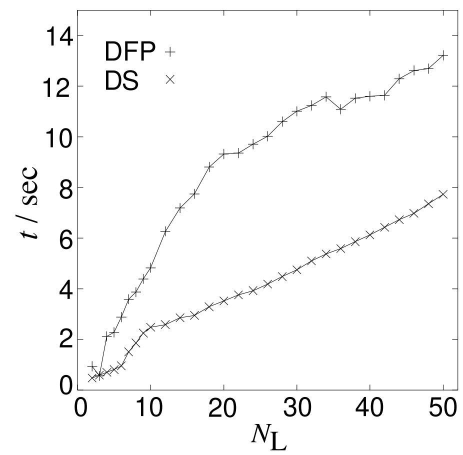

Figure 6: Computation time as a function of the number of curve elements N_L for Downhill Simplex (DS) and DAVIDON-FLETCHER-POWELL (DFP) algorithm

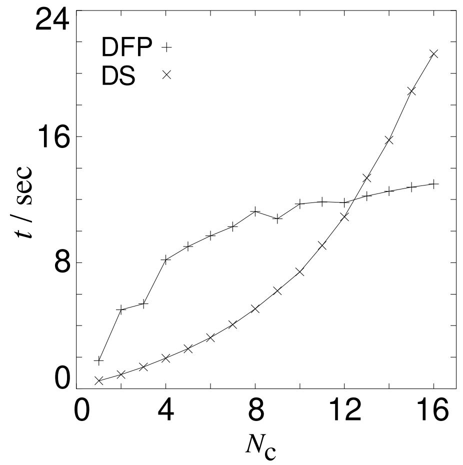

Figure 7: Computation time as a function of the number of series terms N_c for Downhill Simplex (DS) and DAVIDON-FLETCHER-POWELL (DFP) algorithm

Figure 7 shows the measured computation time as a function of the number of series terms N_c with object discretization $N_L = 32$. Obviously, gradient methods as the DFP algorithm are especially powerful if the number of coefficients to be determined is high. However, in Section 3.1 it is shown that approximately 10 series terms are generally sufficient if $\boldsymbol{q}(s)$ is approximated by Chebychev polynomials. Therefore, the Downhill Simplex algorithm is not only easier to implement but also faster in typical cases.

4 PARALLEL COMPUTATION

Section 3 shows that a short computation time can be obtained by an appropriate selection of computation parameters and optimization algorithm. If a further reduction of the computation time is required, e.g., for real-time computation in combination with sensor evaluation, parallel computation can be considered.

In this context, it is two different situations must be distinguished, which are discussed in the following.

1. Different workpiece shapes are independent from each other (*independent computation*).
2. The shape computed in each step depends on the shape computed in the previous step (*dependent computation*).

We implemented a parallel version of the simulation software on a workstation cluster which consists of 9 PCs, each with 133 MHz Intel Pentium processors and 128 Mbytes memory. The parallel communication is established by an Ethernet based bus network (see Wurll [7] for details).

4.1 INDEPENDENT COMPUTATION

For independent computation, at least the following two conditions must be fulfilled: The simulation is performed (quasi-)static and there is no plastic deformation. Under these circumstances, the workpiece shape can be computed in parallel for different positions of the gripper trajectory (starting the optimization algorithm always with the same initial guess for $\boldsymbol{q}(s)$). However, it is more favorable to use the result of a previous step as initial guess since the difference to the correct shape is typically smaller

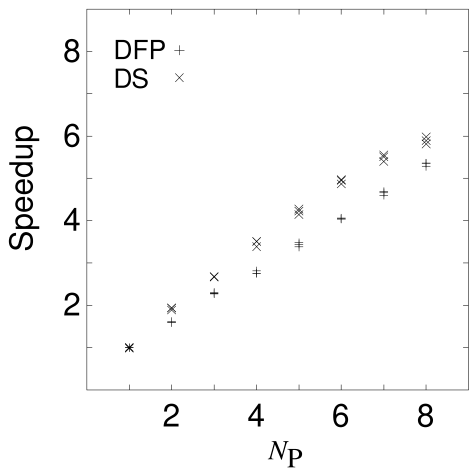

Figure 8: Measured speedup for independent computations using Downhill Simplex (DS) and DAVIDON-FLETCHER-POWELL (DFP) algorithm with N_P being the number of processors (each experiment performed three times)

in this case. The resulting speedup is linear, as shown in Figure 8 for the benchmark C given in Figure 5.

However, if any interaction between workpiece and obstacles has to be considered, the computed object shapes may not be valid, even if the conditions given above are fulfilled. This problem is demonstrated in Figure 9 with the following benchmark D. The object is moved downwards (into direction MD, direction of gravity) and collides with an obstacle. All steps of the simulation are computed independently from each other, the minimization algorithm is always started with an undeformed workpiece as initial guess. While the object shape is simulated correctly for the first four steps, the object 'jumps' to the lower side of the obstacle in step five which is obviously incorrect. This is caused by the fact that the optimization algorithm seeks for the minimum energy which is next to the initial guess. Starting with an undeformed object, the algorithm always finds the global minimum in this example.

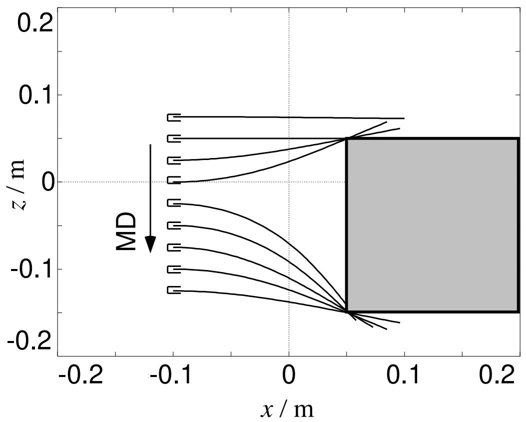

Figure 9: Benchmark D: Obstacle interaction of a deformable linear object while being moved into direction MD for independent computation of all steps. The result is false for step 5 to 9.

The correct result is obtained if the object shape computed in step i–1 is taken as initial guess in step i. In this case, the algorithm finds the local minimum which is next to the shape computed in the previous step. The parallel computation of several points of the trajectory is obviously not possible in this case.

It is generally not necessary to use the result of step i – 1 as initial guess in step i in those cases. It is also possible to use a shape computed a few steps in the past. Thus, correct results can be achieved without completely loosing the advantage of parallel computation. However, the number of computations to be computed in parallel must be chosen carefully if interaction with obstacles may occur.

4.2 DEPENDENT COMPUTATION

If the object is deformed plastically while being handled by the robot or the simulation shall be performed dynamically, the simulation of step i requires the results obtained in step i – 1 (for example the velocity of each mass element). This holds true not only for initializing the optimization algorithm, but also as input data for the computation. Therefore, the approach discussed above is not feasible.

Generally, it should be assumed that the final computation of the object shape $\boldsymbol{q}^i$ in simulation step i can be performed fast if the difference between the actual minimum and the initial guess is small. Based on this idea, we can compute a good initial guess for the steps $\boldsymbol{q}^{i+1}$, $\boldsymbol{q}^{i+2}$, ... while computing the correct result $\boldsymbol{q}^i$.

Let us assume the case of performing a dynamic simulation and having two independent computation tasks T_1 and T_2. Having a known initial shape $\boldsymbol{q}^0$ of the object, we use the following algorithm: Based on $\boldsymbol{q}^0$, T_1 and T_2 compute the shapes for the next two steps, $\boldsymbol{q}^1$ and $\boldsymbol{q}^2$ simultaneously. While the obtained result is valid for $\boldsymbol{q}^1$, it is an approximation for $\boldsymbol{q}^2$ (since the final computation of $\boldsymbol{q}^2$ requires $\boldsymbol{q}^1$ as input). In the next step, $\boldsymbol{q}^1$ can be used for finally computing the final position $\boldsymbol{q}^2$ and an approximation for $\boldsymbol{q}^3$, and so on. This method can be easily employed for any number of independent computation tasks.

However, the assumption that a good initial guess results in a fast computation of the energy minimum is not always true, but depends on the optimization algorithm. On the one hand, the Downhill Simplex algorithm requires $3N_c + 1$ independent guesses for each of the $3N_c$ parameters. Having just one good (maybe almost optimal) guess from the previous steps does not significantly simplify the problem. Accordingly, the possible speedup is rather low.

Gradient-based algorithms as DFP, on the other hand, come close to the minimum rather fast even if the initial guess is bad, but require many iterations to finally determine the minimum with the required accuracy. Therefore, the influence of a good initial guess is rather small, resulting in a low speedup. However, if the guess computed in step i–1 is the actual workpiece shape, DFP algorithm terminates immediately. This is the case if there is no plastic deformation in a step or the acceleration of all mass elements is constant in time, respectively. Here, the speedup is lower than for independent computation (Section 4.1), but also linear.

Figure 10 shows the measured speedup for a dynamic computation of the benchmark C described in Section 3.2 for both minimization algorithms. Due to the characteristics of the Downhill Simplex algorithm described above, the speedup is almost neglectable. For the DFP algorithm, a maximum speedup of about two is obtained for three computation tasks. If the task number is further increased, the speedup is reduced due to the following reasons: First, the more computation tasks we use the more guesses we compute for future simulation steps. However, if a guess for step $i+k$ is computed in step i, the significance of the guess decreases with increasing k. Therefore, the additional use of the tasks becomes smaller from task to task. Second, the effort required for communication increases with the number of computation tasks.

Compared with the resulting speedup, it is found that the effort for the parallel computation is generally too high in the case of dependent computations. Some additional possibilities for parallelizing sub-tasks, e.g., the computation of the energy integral, have been considered but have not been implemented because the expected speedup is too low.

5 CONCLUSIONS

In this work, the efficient simulation of deformable linear objects is proposed and investigated. We extend the

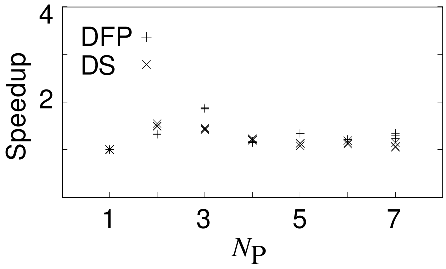

Figure 10: Measured speedup for dynamic (dependent) computation using Downhill Simplex (DS) and DAVIDON-FLETCHER-POWELL (DFP) algorithm with N_P being the number of processors (each experiment performed three times)

principle of minimal potential energy for a treating of plastic deformation and give an novel approach for considering dynamics. The discretization of the workpiece length, the series expansion used for approximating the workpiece shape, and the algorithm for computing the minimal energy are identified to be of major influence on precision and computation time. For these parameters, an experimental investigation is performed. Additionally, we discuss the usage of parallel processing.

For approximating the set of functions describing the workpiece, Chebyshev series turn out to be well-suited. The accuracy of the computed object nodes is found to be rather high even if the discretization of the object is rather coarse.[10] For solving the minimization problem, the Downhill Simplex algorithm is found to be sufficient. Based on these results, it is possible to assess the parameters for the efficient simulation of a given task. Parallelizing the computation is efficient if the single steps of the trajectory can be computed independently from each other, otherwise the speedup is rather poor.

In this paper, we discuss the computation of the workpiece shape if the trajectory of the gripper is known a priori. However, if the goal is to perform a distinct (assembly) operation, the problem is generally inverse: given the task which shall be performed, a suited gripper trajectory must be computed. The investigation of this *Inverse modeling problem* will be our next step.

ACKNOWLEDGMENT

The work described in this paper was funded by the European Commission in the framework of Brite-EuRam project BE-3323 HANDFLEX (Integrated modular solution for HANDling of FLEXible materials in industrial environments).

[10] Please note that this holds true only for the nodes themselves. Between the nodes, the difference to the correct workpiece shape is generally higher.

REFERENCES

[1] Hirai, S., Wakamatsu, H., and Iwata, K.: "Modeling of deformable thin parts for their manipulation". In: Proc. 1994 Int. Conf. on Robotics and Automation (ICRA'94), vol. 4, pp. 2955-2960, San Diego, USA, May 1994.

[2] Karl, S.: "Physikalische Modellierung des mechanischen Verhaltens von deformierbaren, linearen Objekten" (Physical modeling of the mechanical behavior of deformable linear objects). Master Thesis, Universität Karlsruhe (TH), Germany, 1999.

[3] Press, W. H., et al.: "Numerical recipes in C". Second edition, Cambridge University Press, 1992.

[4] Remde A., Henrich D., and Wörn H.: "Manipulating deformable linear objects: Contact state transitions and transition conditions". In: 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99), Kyongju, Korea, October 1999.

[5] Wakamatsu, H., Hirai, S., and Iwata, K.: "Modeling of linear objects considering bend, twist and extensional deformations". In: Proc. 1995 Int. Conf. on Robotics and Automation (ICRA'95), vol. 1, pp. 433438, Nagoya, Japan, May 1995.

[6] Wakamatsu, H., et al.: "Dynamic analysis of rodlike object deformation towards their dynamic manipulation". In: Proc. 1997 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems" (IROS'97), pp. 196ff, Grenoble, France, Sept. 1997.

[7] Wurll, C., Henrich, D.: "Ein Workstation-Cluster für paralleles Rechnen in Robotik-Anwendungen" (A workstation cluster for parallel processing in robotic applications). In: Proceedings der 4. ITG/GI-Fachtagung Arbeitsplatz-Rechensysteme (APS'97), Universität Koblenz-Landau, Germany, 21.-22. May, 1997, pp. 187 - 196.

[8] Zheng, Y. F., Pei, R., and Chen, C.: "Strategies for automatic assembly of deformable objects". In: Proc. 1991 Int. Conf. on Robotics and Automation (ICRA'91), vol. 3, pp. 2598-2630, Sacramento, USA, April 1991.