Robot manipulation of deformable objects, Henrich D., Wörn H. (Eds.),
Springer-Verlag, London, 2000, ISBN: 1-85233-250-6, pp 43-70

Direct and Inverse Simulation of Deformable Linear Objects

Axel Remde and Dominik Henrich

Abstract. In this chapter, the quantitative numerical simulation of the behavior of deformable linear objects, such as hoses, wires and leaf springs is studied. We first give a short review of the physical approach and the basic solution principle. Then, we give a more detailed description of some key aspects: We introduce a novel approach concerning dynamics based on an algorithm very similar to the one used for (quasi-) static computation. Then, we look at the plastic workpiece deformation, involving a modified computation algorithm and a special representation of the workpiece shape. Then, we give alternative solutions for two key aspects of the algorithm, and investigate the problem of performing the workpiece simulation efficiently, i.e., with desired precision in a short time. In the end, we introduce the inverse modeling problem which must be solved when the gripper trajectory for a given task shall be generated.

1. Introduction

In this chapter, we consider the quantitative, numerical simulation of the behavior of a deformable, linear object (DLO) handled by a robot manipulator. In addition to the development of special-purpose grippers and the handling based on sensor information, this problem has been addressed in several works. Zheng et al. perform an off-line computation of the gripper trajectory in order to insert a flexible beam into a hole and succeed in performing the task without the additional usage of sensors [1]. Hirai et al. develop an algorithm for the 2D computation of elastically deformable thin parts based on the principle of minimal potential energy [2]. For

DLOs, Wakamatsu et al. extend this approach to 3D-computation [3] and to the consideration of dynamics based on Hamilton's principle [4].

These works demonstrate that simulating the behavior of DLOs numerically is possible. However, in practice, we need to consider some additional items.

- While dynamics needs to be considered in some cases, this is not necessary in many other cases. Therefore, it is desirable to use an algorithm which allows a changing from static to dynamic computation with little additional effort. Yet, this is not possible when directly employing Hamilton's principle.
- In many practical applications, the workpiece is not only deformed elastically, but also plastically. However, the occurrence of plastic deformation is not being considered in the state of the art. Besides the physical effect itself, we find that an appropriate internal representation of the workpiece shape is necessary when considering plastic deformation.
- When performing simulation on a workpiece, it is generally desirable to perform the computation with sufficient precision in a short time, i.e., to do it in an efficient way. Therefore, we consider different alternatives for some key aspects of the computation algorithm, which are of major influence on the computation time. Additionally, we investigate the influence of some basic computation parameters on both computation time and precision of the results.
- Besides the selection of appropriate computation parameters, parallel processing is an obvious way to reduce the computation time. Thus, we investigate different possibilities of parallizing the shape computation and discuss their advantages and limits.
- One major application field of the simulation of deformable objects is the off-line generation of gripper trajectories for a given (assembly) task. That is, for each time step of the assembly process, the task defines certain boundary conditions which must be fulfilled by the workpiece shape. The goal is to compute a gripper trajectory that fulfills these boundary conditions. Because this problem is just an inverse to the computation of the workpiece shape for given boundary conditions, we call it the "*Inverse Simulation Problem*". In the last part of this section, this kind of task is discussed.

2. Principal Approach

2.1 Physical Principle

In this chapter, we give a short review of the physical approach used for the simulation of DLOs as well as the basic computation principle.

According to the fundamental physical principle of minimal potential energy, dynamic systems assume a minimum of their total potential energy W in any stable state. This holds true not only for systems of discrete elements, e.g., lumped masses and springs, but also for a deformable continuum like DLOs. Based on this

principle, the shape of a deformable object can be computed rather easily, if the boundary conditions are known. Neglecting linear extension, potential energy due to gravity, bending and torsion must be considered. Thus, the following optimization problem has to be solved.

$$W = \int_0^L (W'_{grav} + W'_{bend} + W'_{tor})\, ds \rightarrow \min \tag{1}$$

In Eqn. 1, L is the length of the workpiece, $s \in [0, L]$ is the curve length measured along the workpiece. W'_{grav}, W'_{bend} and W'_{tor} are the potential energy caused by gravity, bending and twisting (per length) respectively. For each point of the workpiece, they are given as follows:

$$W'_{grav}(s) = \rho A |\boldsymbol{g}| z(s) \tag{2a}.$$

Here, ρ and A are the density and the cross section area, $\boldsymbol{g}$ is the acceleration vector due to gravity, and $z(s)$ the coordinate of the workpiece point along $\boldsymbol{g}$.

Being R_{bend} and R_{tor} the (constant) bending and torsional rigidity, and $\kappa(s)$ and $\tau(s)$ the local curvature and twisting, the respective potentials are

$$W'_{bend} = \frac{1}{2} R_{bend} \kappa(s)^2 \tag{2b}$$

$$W'_{tor} = \frac{1}{2} R_{tor} \tau(s)^2 \tag{2c}.$$

2.2 Computation

When computing the workpiece shape, the goal is to determine those functions $W'_{grav}(s)$, $W'_{bend}(s)$, and $W'_{tor}(s)$ that fulfill the condition given in Eqn. 1. In order to perform this computation, the following steps are performed:

- First, a vector $\boldsymbol{q}(s)$ is determined which fulfills the following requirements:
  - $\boldsymbol{q}(s)$ describes the workpiece shape (Cartesian coordinates $\boldsymbol{x}(s)$ of each workpiece points) unequivocally
    $$\boldsymbol{x}(s) = \boldsymbol{f}(\boldsymbol{q}(s)).$$
  - The total potential energy $W'(s)$ per length (having the portions W'_{grav}, W'_{bend}, and W'_{twist}) can be expressed as a function of $\boldsymbol{q}(s)$
    $$W'(s) = f(\boldsymbol{q}(s)).$$
- For a three-dimensional computation, $\boldsymbol{q}(s)$ has three components: $\boldsymbol{q}(s) = [q_1(s), q_2(s), q_3(s)]^T$. Thus, Eqn. 1 is transformed into
  $$W = \int_0^L f(\boldsymbol{q}(s))\, ds \rightarrow \min \tag{3},$$
  and determining the object shape means to compute the components $q_i(s)$ of vector $\boldsymbol{q}(s)$ in order to satisfy Eqn. 3. This is a “calculus of variations” problem, described by a set of partial differential equations (Eulerian equations). Because an analytical solution for these equations can not bee found in most cases, we use the well-known approximation method introduced by Ritz [5].

- In this method, the single components of $\boldsymbol{q}(s)$ are expanded into a series with N_c terms.

$$q_i(s) = \sum_{j=0}^{N_c-1} c_{i,j}\, Q_j(s) \tag{4}.$$

  Here, $Q_j(s)$ are the basis functions of the series and $c_{i,j}$ are the according coefficients. Thus, $\boldsymbol{q}(s)$ is represented by a vector $\boldsymbol{c}$ having $N = 3N_c$ components. By this step, the problem of computing a vector $\boldsymbol{q}(s)$ of functions is reduced to the problem of computing the vector $\boldsymbol{c}$ of coefficients, and Eqn. 1 is finally transformed into

$$W(\boldsymbol{c}) = \int_0^L f(\boldsymbol{c}, s)\, ds \rightarrow \min \tag{5}.$$

- Since this equation can not be solved analytically as well, the integral in Eqn. 5 is computed by numerical integration and the resulting discrete minimization problem is solved by numerical optimization in Multidimensions.

For static computations, this approach is straight forward. In [2, 3], it is used for computing the static shape of DLOs.

3. Consideration of Special Aspects

3.1 Dynamics

When regarding a robot system manipulating a deformable workpiece, the goal is generally not to compute a single workpiece shape with given boundary conditions, rather than to compute the shape of the object at each point of the gripper trajectory. Wakamatsu et al. [4] point out that the resulting object shape is highly sensitive to the velocity of the gripper motion. If this velocity is sufficiently small ($v_{\text{Gripper}} \rightarrow 0$), the workpiece can be regarded as resting in each simulation step. Inertial forces (causing, e.g., workpiece oscillations) are neglected. This behavior is called *quasi-static*.

When the DLO is manipulated fast, the inertia forces caused by the object acceleration can not be neglected. Therefore, the shape in step *i* depends on the results of the position and velocity in step *i*–1 and the acceleration between step *i*–1 and step *i*. This behavior is called *dynamic*.

The principle of minimal potential energy (Eqn. 1) holds true only for static or quasi-static computations. An extension towards the consideration of dynamics leads to Hamilton's principle. With *T* being the kinetic energy, Eqn. 1 is replaced by

$$S = \int_{t_0}^{t_0+\Delta t} L\, dt = \int_{t_0}^{t_0+\Delta t} (T - W)\, dt \overset{!}{=} \min \tag{6a}.$$

A formulation equivalent to Eqn. 6a is given by the Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \stackrel{!}{=} 0 \qquad \text{(6b).}^{1}$$

Wakamatsu et al. [4] present a method that computes the shape of DLOs dynamically based by solving Eqn. 6b.

Hamilton's principle is a straight forward method when considering dynamics, but it requires a significantly extended approach compared to the static computation. However, since dynamic computation (which is rather time consuming compared to static computations) is not required in many cases, it is not advantageous. Therefore, we propose a novel approach that allows to perform static and dynamic computations using the same principle.

We first consider an isolated mass element Δm of the workpiece. Given x_i, $i \in [1..3]$ as its Cartesian coordinates, we obtain (because of $\partial T / \partial x_i \equiv 0$ and $\partial W / \partial \dot{x}_i \equiv 0$) the well-know equation for the motion of a mass element in a potential field from Eqn. 6b

$$\Delta m\, \ddot{x}_i = -\frac{\partial W}{\partial x_i} \qquad \text{(7a),}$$

or, in vectorial form

$$\Delta m\, \ddot{\boldsymbol{x}} = -\nabla_x W \qquad \text{(7b).}^{2}$$

With $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\dot{\boldsymbol{x}}(t_0) = \boldsymbol{v}_0$ and $\boldsymbol{x}(t_0 + \Delta t) = \boldsymbol{x}_1$, $\dot{\boldsymbol{x}}(t_0 + \Delta t) = \boldsymbol{v}_1$, the acceleration $\ddot{\boldsymbol{x}}_1$ in the time interval $[t_0 \ldots t_0 + \Delta t]$ is given by

$$\ddot{\boldsymbol{x}}_1 = \frac{\boldsymbol{v}_1 - \boldsymbol{v}_0}{\Delta t} \qquad \text{(8).}$$

Additionally, we find with $\bar{\boldsymbol{v}}_{0,1}$ being the mean velocity in the considered time interval

$$\bar{\boldsymbol{v}}_{0,1} = \frac{\boldsymbol{x}_0 - \boldsymbol{x}_1}{\Delta t} = \frac{\boldsymbol{v}_0 + \boldsymbol{v}_1}{2} \quad \leftrightarrow \quad \boldsymbol{v}_1 = \; 2\frac{\boldsymbol{x}_1 - \boldsymbol{x}_0}{\Delta t} - \boldsymbol{v}_0 .$$

By combining this relation with Eqn. 4, we obtain $\ddot{\boldsymbol{x}}_1$ as

$$\ddot{\boldsymbol{x}}_1 = \frac{2}{\Delta t^2}\left(\boldsymbol{x}_1 - \boldsymbol{x}_0 - \boldsymbol{v}_0 \Delta t\right).$$

This relation can be inserted into Eqn. 7b for the motion of Δm in potential field W.

$$\nabla_x W(\boldsymbol{x}_1) + \frac{2\,\Delta m}{\Delta t^2}\left(\boldsymbol{x}_1 - \boldsymbol{x}_0 - \boldsymbol{v}_0 \Delta t\right) = 0 \qquad \text{(9).}$$

The left side of this equation is equal to the gradient of

$$U(\boldsymbol{x}) = W(\boldsymbol{x}) + \Delta m \left(\frac{\boldsymbol{x} - \boldsymbol{x}_0 - \boldsymbol{v}_0 \Delta t}{\Delta t}\right)^2 \qquad (10)$$

---

1 In the (quasi) static case, we find that Eqn. 6b is equivalent to the differential formulation of Eqn. 1, because of $T \equiv 0$ and $\partial W / \partial \dot{q}_i \equiv 0$.

2 Eqn. 3b is equivalent to Newton's law of motion $\boldsymbol{F} = m\boldsymbol{a}$.

with respect to $\boldsymbol{x}$. Thus, to solve Eqn. 9, we have to deal with the optimization problem

$$U(\boldsymbol{x}) \rightarrow \min \tag{11}.$$

The position $\boldsymbol{x}$ which satisfying Eqn. 11 is the desired position $\boldsymbol{x}_1$ of the mass element at time $t_0 + \Delta t$. Note that the acceleration in time interval Δt is assumed to be constant here. This is equivalent to $\Delta t \rightarrow 0$. Thus, Δt must be sufficiently small for the numerical computation.

As formulated in Eqn. 1, the static position of the DLO can be determined by minimizing its potential energy W. This holds true for a single lumped mass Δm, too. Therefore, computing the position $\boldsymbol{x}$ of a lumped mass dynamically is reduced to solving the minimization problem for $U(\boldsymbol{x})$ given in Eqn. 11, instead of $W(x)$. The only difference between U and W is the additional term

$$W_{\text{dyn}} = \Delta m \left(\frac{\boldsymbol{x} - \boldsymbol{x}_0 - \boldsymbol{v}_0 \Delta t}{\Delta t} \right)^2 \tag{12}$$

in U. This relation holds true not only for a single lumped mass, but also for a number of N elements moving in a potential field. For considering a continuum, a border crossing $N \rightarrow \infty$ must be carried out. Doing this,

$$W_{\text{dyn}} = A\rho \int_0^L \left(\frac{\boldsymbol{x}(s) - \boldsymbol{x}_0(s) - \boldsymbol{v}_0(s)\Delta t}{\Delta t} \right)^2 ds = \int_0^L W'_{\text{dyn}}\, ds$$

is obtained from Eqn. 12. W'_{dyn} takes the workpiece dynamics per length into account.[3] Therefore, the position $\boldsymbol{x}$ of each workpiece point at time $t = t_0 + \Delta t$ can be computed from the known position $\boldsymbol{x}_0$ and velocity $\boldsymbol{v}_0$ for $t = t_0$ by solving

$$U = \int_0^L (W'_{\text{grav}} + W'_{\text{bend}} + W'_{\text{tor}} + W'_{\text{dyn}})\, ds \overset{!}{=} \min \tag{13}.$$

Switching from quasi-static to dynamic computation can now be done by simply adding the term W'_{dyn} to the integrand.

3.2 Plastic Deformation

3.2.1 Considering Plastic Deformation

So far, we have assumed the workpiece deformation to be totally elastic. However, in many practical applications, the occurrences of considerable plastic deformation are in presence. Because of the workpiece behavior depends on many factors, the exact consideration is rather difficult, even if only linear stress without bending or twisting is assumed. The most important influence factors are:

- direction of force (tensile or compression load)
- duration of force application

---

[3] Though W'_{dyn} represents the workpiece dynamics in the minimization problem, it is not the kinetic energy of the object!

- velocity of force increase and decrease

Additionally, the behavior is generally different for linear stress, bending and twisting. However, an exact consideration is not necessary in many cases, and a rather coarse approximation is sufficient.[4] Thus, we use the following assumptions for significantly simplifying the problem.

The workpiece behavior is assumed to be *elastic-ideal plastic* [6], i.e., the stress-strain relation is given by Figure 1 (left), causing a deformation behavior according to Figure 1 (right). Starting in the stress-free state, the internal stress $\sigma(\varepsilon)$ increases linearly (Hook's law, Phase 1). Besides a certain yield strain ε_E, the stress remains constant for increasing strain (Phase 2). For a subsequent force relief, the stress decreases on a parallel to the original Hook's straight line (Phase 3) with strain ε_p remaining after complete stress release. This behavior implicates especially that the stress-strain relation is independent of the deformation history, which does generally not hold true in reality.

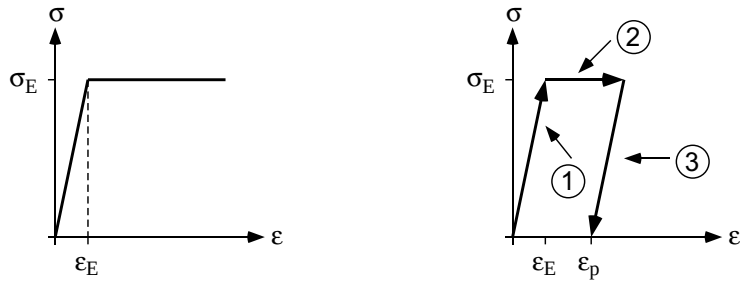

Figure 1: Left: Stress-strain diagram for elastic-perfectly plastic material behavior (σ: stress, ε: strain, σ_E: yield stress, ε_E: yield strain). Right: stress-strain-cycle for tensile load (ε_p: residual strain).

The idealized consideration of plastic deformation according to Figure 1 holds true for a linear (tensile) load. For bending and twisting a similar behavior can be assumed. In these cases, strain ε must be replaced by curvature κ or torsion τ, respectively. For the consideration of bending, the following additional simplification is used.

For bending loads, the amount of strain is different for the single fibres, depending on their distance from the neutral axis. Thus, for a circular workpiece of radius R, the deformation has to be considered separated for all distances $r \in [0, R]$ from the neutral axis. As simplification, we assume the curvature κ to be identical for the total cross section, with plastic deformation occurring for κ being greater

4 An exact consideration often fails because of the missing material parameters.

than the threshold curvature κ_E. This simplification can be derived directly from the assumption of a one-dimensional object of negligible cross-section.[5]

3.2.2 Modification of the Computation Algorithm

The simulation of plastic deformation requires the following modifications in the computation algorithm.

For computing the potential energy due to bending and twisting, the proportional relations $W_{bend} \cong \kappa^2$ and $W_{tor} \cong \tau^2$ (Eqn. 2) are not valid. Instead, the relation

$$W'_{bend} = \int_0^{\kappa} \sigma(\tilde{\kappa})\, d\tilde{\kappa}$$

must be used for computing the bending energy (and, correspondingly, the torsional energy), with $\sigma(\tilde{\kappa})$ given by the stress-strain-diagram.

The occurrence of plastic deformation implies a change in the stress-free workpiece shape. Thus, the stress-free workpiece shape used for the computation in simulation step *i* is given by the total plastic deformation of the previous simulation steps 0, 1, ..., *i*-1. The additional deformation computed in step *i* contains two portions. Its plastic portion (according to the stress-strain-relation) must be added to the plastic deformation computed in the previous steps, its elastic portion of step *i* is ignored in the subsequent simulation steps.

Thus, the computation algorithm for step *i* is as follows:

```
1    D_New := 0;
     Optimum := false;
2    repeat
3       D := D_plastic_{i-1} + D_New;
4       W_grav := W_grav(D);
5       W_bend := W_bend(D_New);
6       W_tor := W_tor(D_New);
7       W := W_grav + W_bend + W_tor;
8       if Minimum(W) then
9             Optimum := true
10      else D_New := D_New + ΔD;
11   until Optimum;
12   D_plastic_i := D_plastic_{i-1} + PlasticPortion(D_New);
```

In this algorithm, the vector ***D*** represents the workpiece deformation due to bending and twisting[6], and ***D_plastic*** is its plastic portion according to the stress-strain-relation. ***D_New*** is the (additional) deformation computed in simulation step *i*

---

5 In reality, the different curvature (and thus, bending stress) of the single fibres causes internal stress within the workpiece.

6 For the adding of deformations, please refer to Section 3.3.2.

and $\Delta\boldsymbol{D}$ is the variation of ***D**_New* in each iteration of the numerical optimization. The optimization itself is represented by the **repeat** ... **until**-loop.

In each step of the numerical optimization, only ***D**_New* is used for computing the energy due to bending and twisting, while both ***D**_New* and the plastic deformation of the previous simulation steps are used for computing the potential due to gravity. This is due to the fact that the energy required for plastical bending and twisting is 'lost' irreversibly and, thus, does not have to be considered for the future simulation steps. However, the stress-free workpiece shape (and, thus, the gravity potential) is influenced by the previous plastic deformation.

3.3 Workpiece Representation

3.3.1 Representation by Curvature and Torsion

In line 3 and line 12 of the algorithm given in Section 3.2.2, different deformation portions have to be added. The addition of deformations causes some restrictions to the internal representation of the workpiece shape.

According to Section 2, the workpiece shape is represented by a vector $\boldsymbol{q}(s)$ of three functions q_i which must be suited for computing both the Cartesian position of each workpiece point and the potential energy. However, we did not give any further information on what kind of functions should be used.

One approach is to directly use the Cartesian coordinates $\boldsymbol{x}(s) = [x(s), y(s), z(s)]^T$ of each workpiece point. However, it is found that computing the potential energies due to bending and twisting is rather complicated. Therefore, this representation is not desirable.

Another approach is to describe the accompanying trihedron (i.e., a "local" Cartesian coordinate system consisting of tangent vector $\boldsymbol{t}(s)$ and two normal vectors $\boldsymbol{n}$, $\boldsymbol{b}$), $\{\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}\}$ for each point $s \in [0, L]$ of the DLO with respect to a global Cartesian coordinate system. With this representation, the global coordinates of each point on the workpiece can be computed easily by just integrating the tangent vector over s, and there exists a rather simple relation between these vectors[7] and the local amount of curvature and torsion. This method is used by Wakamatsu et al. [3, 4], describing the orientation of the accompanying trihedron by three Eulerian angles φ, θ, ψ. Thus, the vector of functions representing the workpiece shape is $\boldsymbol{q}(s) = [\varphi(s), \theta(s), \psi(s)]^T$.

The main drawback of Eulerian angles (or, similarly, roll, pitch and yaw angles) is that curvature and torsion have impact not only on one, but on all of them. Thus, the relation between workpiece deformation (given by curvature and torsion) on the one hand, and Eulerian angels on the other hand, is not invertable. In conclusion, given a set of Eulerian angles (as functions of s), it is not possible to compute curvature and torsion, and given two sets of Eulerian angles, it is not possible to add the corresponding deformations.

---

7 and their first derivatives with respect to s

This problem will not occur as long as only elastic deformation is regarded. But according to the algorithm given in Section 3.2.1, considering plastic deformation requires the addition of deformations (line 3 and line 12). Thus, a representation by Eulerian angles is not possible.

To avoid this problem, we choose a different approach, in which curvature and torsion are directly used as internal representation of the DLO. Being $\{\boldsymbol{t}(s), \boldsymbol{n}(s), \boldsymbol{b}(s)\}$ the accompanying trihedron at position s, we use the following functions $q_d(s)$, $q_\kappa(s)$ and $q_\tau(s)$ with $\boldsymbol{q}(s) = [\, q_d(s), q_\kappa(s), q_\tau(s)]^T$ to compute the accompanying trihedron at position $s+\Delta s$.

$q_d(s)$ represents the local direction of curvature, while q_κ is the local amount of curvature. In order to describe the object bending from point s to point $s+\Delta s$ on the DLO, the accompanying trihedron is rotated by the (infinitesimal) angle

$$dw_{\kappa} = q_{\kappa}(s)\,ds$$

about the axis which is formed by rotating one of the normal vectors by q_d about tangent vector $\boldsymbol{t}$.[8] Figure 2 (top) shows a DLO section with the accompanying trihedron and the rotation axis. The result of this rotation is a new trihedron $\{\boldsymbol{b}'(s), \boldsymbol{n}'(s), \boldsymbol{t}'(s)\}$, as shown in Figure 2 (bottom).

$q_\tau(s)$ is the local amount of torsion. Torsion is performed by rotating the trihedron $\{\boldsymbol{b}'(s), \boldsymbol{n}'(s), \boldsymbol{t}'(s)\}$ by the angle

$$dw_{\tau}(s) = q_{\tau}(s)\,ds$$

about the $\boldsymbol{t}'$-axis.[9] As result of this second rotation, the accompanying trihedron $\{\boldsymbol{t}(s+ds) = \boldsymbol{t}'(s), \boldsymbol{n}(s+ds), \boldsymbol{b}(s+ds)\}$ at position $s+ds$ is obtained.

3.3.2 Adding Deformations

With the method derived above, adding deformations according to Section 3.2.2 is rather simple. When adding two deformation portions, (I) and (II), each of them has a value for q_d, q_κ, and q_τ.

Because twisting is always performed around the tangent vector (i.e., the rotation vector is identical for both portions), the rotations can be simply added

$$dw_{\tau} = (q_{\tau,\mathrm{I}} + q_{\tau,\mathrm{II}})\,ds \qquad (14a).$$

For the curvature, the directions of both portions are generally different. Here, the corresponding rotation vectors $d\boldsymbol{w}_{\kappa,\mathrm{I}}$ and $d\boldsymbol{w}_{\kappa,\mathrm{II}}$ are added.[10] These vectors lay in the plane given by $\boldsymbol{n}(s)$, $\boldsymbol{b}(s)$, with the absolute values $dw_{\kappa,\mathrm{I}}$ and $dw_{\kappa,\mathrm{II}}$. The directions of curvature are given by $q_{d,\mathrm{I}}$ and $q_{d,\mathrm{II}}$. According to Figure 3, direction q_d and absolute value dw_κ of the resulting curvature are given by

$$\begin{aligned} dw_{\kappa}^{\,2} &= dw_{\kappa,\mathrm{I}}^{\,2} + dw_{\kappa,\mathrm{II}}^{\,2} + 2\,dw_{\kappa,\mathrm{I}}\,dw_{\kappa,\mathrm{II}}\cos(q_{d,\mathrm{I}} - q_{d,\mathrm{II}}) \\ \tan(q_d) &= \frac{dw_{\kappa,\mathrm{I}}\sin(q_{d,\mathrm{I}}) + dw_{\kappa,\mathrm{II}}\sin(q_{d,\mathrm{II}})}{dw_{\kappa,\mathrm{I}}\cos(q_{d,\mathrm{I}}) + dw_{\kappa,\mathrm{II}}\cos(q_{d,\mathrm{II}})} \end{aligned} \qquad (14b).$$

---

8 Without loss of generality, we assume that the rotation is performed about $\boldsymbol{n}(s)$.

9 $q_\kappa(s)$ and $q_\tau(s)$ are the local amount of curvature and torsion at position s.

10 Adding the rotation vectors is permissible because both rotation angles are infinitesimal.

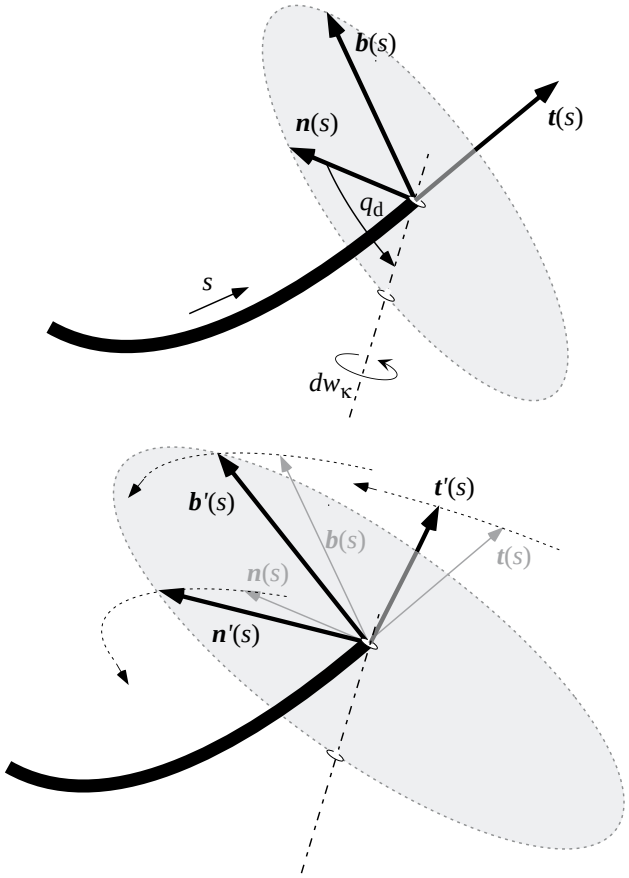

Figure 2: Expression of DLO curvature by its direction q_d, and amount, $dw_\kappa = q_\kappa ds$. Top: Rotation of trihedron {$\boldsymbol{t}(s)$, $\boldsymbol{n}$(s), $\boldsymbol{b}(s)$}. Bottom: Resulting trihedron {$\boldsymbol{t}'(s)$, $\boldsymbol{b}'(s)$, $\boldsymbol{n}'(s)$}.

3.3.3 Transformation into Global Coordinates

The vector $\boldsymbol{q}(s)$ given above is the internal representation of the workpiece shape for solving the minimization problem given in Eqn. 1. However, the final goal is to compute the coordinates of each point on the workpiece in a global Cartesian coordinate system. For this purpose (and for computing the gravity potential in Eqn. 2a), a transformation that transforms the internal representation $\boldsymbol{q}(s)$ into the Cartesian coordinates $\boldsymbol{x}(s)$ for each point on the DLO is necessary.

Generally, $\boldsymbol{x}(s)$ is given by

$$\boldsymbol{x}(s) := \int_0^s \boldsymbol{t}(\tilde{s})\, d\tilde{s} \tag{15},$$

with as the unit tangent vector of the DLO at $\tilde{s}$ with respect to the global Cartesian system.[11]

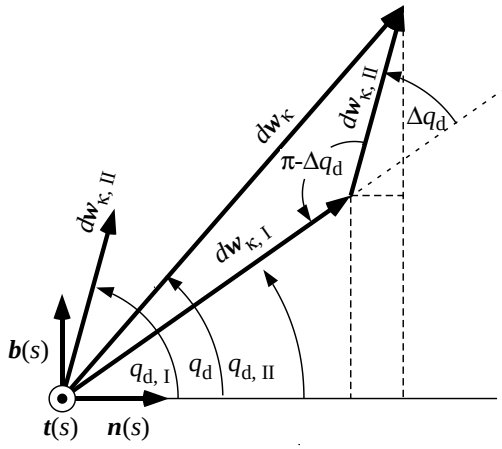

Figure 3: Adding of rotation vectors $d\boldsymbol{w}_{\kappa\,I}$ and $d\boldsymbol{w}_{\kappa,\,II}$, both lying in the ***b*-*n***-plane of the accompanying trihedron at DLO position *s*

According to Section 3.3.1, the accompanying trihedron at position *s*+*ds* on the DLO is derived from the accompanying trihedron at position *s* by performing two rotations, with the first one representing bending and the second one representing twisting. For computing ${}_{\text{Global}}\boldsymbol{t}(s)$, it is helpful to draw up the transformation matrix ${}_s\boldsymbol{T}_{s+ds}$, with

$$ {}_s\boldsymbol{t}(s+ds) = {}_s\boldsymbol{T}_{s+ds} \cdot {}_{s+ds}\boldsymbol{t}(s+ds), $$

i.e., to express the orientation of the tangent vector at *s*+*ds* with respect to the accompanying trihedron at *s*.[12]

For this purpose, the rotation representing the bending of the DLO is further divided into three rotations:

1. First, the accompanying trihedron $\{\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}\}$ at *s* is rotated by q_d about $\boldsymbol{t}$, aligning $\boldsymbol{n}$ with the axis of rotation in Figure 2 (top). The resulting trihedron is $\{\boldsymbol{t}^{I} \equiv \boldsymbol{t}, \boldsymbol{n}^{I}, \boldsymbol{b}^{I}\}$.
2. Second, the trihedron obtained in step 1 is rotated by $d\boldsymbol{w}_\kappa$ about $\boldsymbol{n}^{I}$. The resulting trihedron is $\{\boldsymbol{t}^{II}, \boldsymbol{n}^{II} \equiv \boldsymbol{n}^{I}, \boldsymbol{b}^{II}\}$.
3. Finally, the trihedron obtained in step 2 is rotated by $-q_d$ about $\boldsymbol{t}^{II}$. The resulting trihedron is $\{\boldsymbol{t}' \equiv \boldsymbol{t}^{II}, \boldsymbol{n}', \boldsymbol{b}'\}$, as shown in Figure 2 (bottom).

By additionally taking torsion (rotation around $\boldsymbol{t}'$) into consideration, we finally find

---

[11] In the following, a leading subscript to a vector gives the coordinate system in which the vector is expressed.

[12] Please note that ${}_s\boldsymbol{t}(s) \equiv [1, 0, 0]^T$.

$$ {}_s\boldsymbol{T}_{s+ds} = \boldsymbol{R}(\boldsymbol{t}, q_\mathrm{d})\, \boldsymbol{R}(\boldsymbol{n}^\mathrm{I}, dw_\kappa)\, \boldsymbol{R}(\boldsymbol{t}^\mathrm{II}, -q_\mathrm{d})\, \boldsymbol{R}(\boldsymbol{t}', dw_\tau) \qquad (16). $$

$\boldsymbol{R}(\boldsymbol{a}, \alpha)$ is the rotation matrix which transforms ${}_2\boldsymbol{x}$ into ${}_1\boldsymbol{x}$ with coordinate system (2) being rotated with respect to system (1) by α about axis $\boldsymbol{a}$. In the case of Eqn. 16, all rotations are performed about elementary vectors of the trihedra. Thus, the single rotation matrices $\boldsymbol{R}$ are very simple.

Based on Eqn. 16, the transformation ${}_\mathrm{Global}\boldsymbol{T}_s$ of the accompanying polyhedron at s into global coordinates is defined recursively by

$$ \boldsymbol{T}(s) = {}_\mathrm{Global}\boldsymbol{T}_s = \boldsymbol{T}(s - ds)\ {}_{s-ds}\boldsymbol{T}_s \qquad (17). $$

Together with Eqn. 15, the global coordinates of point s on the DLO are finally given by

$$ \boldsymbol{x}(s) = \int_0^s \boldsymbol{T}(\tilde{s}) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} d\tilde{s} \,. $$

Please note that Eqn. 17 is meaningful only for $s > 0$. Therefore, we define $\boldsymbol{T}(0)$,i.e., the DLO point held by the gripper, as the transformation matrix which transforms the accompanying trihedron of the gripper into global coordinates, expressed, e.g., by Eulerian angles.

3.4 Series Expansion

Since the number $N = 3N_\mathrm{c}$ of series coefficients that must be determined by the optimization algorithm has major influence on the computation time, the series expansion for the $q_i(s)$ should be a good approximation with few series terms. The more severely the workpiece is being deformed from its stress-free shape, the more complicated becomes this problem. Since the workpiece may generally take an arbitrary shape, it is not possible to find a series expansion that meets all possible situations. In [2, 3], Fourier series are proposed. As an alternative, we investigate the usage of Chebyshev polynomials which are often found to be a good choice for approximating unknown function analytically [7]. The polynomial of order j has the form[13]

$$ Q_j(x) = \cos(j \arccos(x)) \,, $$

and the series expansion for the q_j is then given by Eqn. 4. In order to meet the definition range of the arccos-function, the curve length $s \in [0 \ldots L]$ must be normalized by $x = 2s/L - 1$. If Fourier series are used, a similar normalization to the range $[-\pi \ldots \pi]$ is required.

3.5 Optimization Algorithm

In order to solve the minimization problem in determining the set $\boldsymbol{c}$ of coefficients for the series expansion, any nonlinear optimization algorithm in Multidimensions can be used. We implemented two algorithms of different complexity.

---

[13] Alternatively, the terms Q_j can be expressed by recursively defined polynomials.

The first one is the downhill simplex (DS) algorithm invented by Nelder and Mead. Here, a simplex is the geometric figure consisting of $N + 1$ points (vertices) in N dimensions including their interconnecting line segments, faces, etc. For example, in two dimensions a simplex is formed by a triangle.

The basic principle of the DS algorithm for minimizing the function $U(\boldsymbol{c})$ is seen as follows: At the beginning, an initial simplex consisting of the N+1 coefficient vectors $\boldsymbol{c}_0, \boldsymbol{c}_1, ...\boldsymbol{c}_N$, is chosen arbitrarily. Then, the minimum of U is computed as follows: In each iteration, the vector $\boldsymbol{c}_{worst}$ with the highest function value U is modified according to diverse rules. In this process, the simplex is stepwise contracted and moves downhill towards the minimum of U.

Even though the downhill simplex algorithm is not very efficient, it has the following two advantages: First, it only evaluates the function U itself and does not need its (partial) derivatives. Second, it is easy to implement and is generally a good choice if the aim is "to get something working quickly" [7].

The second algorithm is the Davidon-Fletcher-Powell (DFP) algorithm as a standard variable metric method. Comparing to the DS algorithm, variable metric methods are more powerful. However, they require the evaluation not only of U, but also the vector of its first partial derivatives, ∇U, and the inverse matrix of its second partial derivatives, i.e., the inverse Hessian matrix $\boldsymbol{H}^{-1} = [\nabla U]^{-1}$. Especially the computation of $\boldsymbol{H}^{-1}$ can be very time consuming. Therefore the DFP algorithm computes $\boldsymbol{H}^{-1}$ in each optimization step approximately from $\boldsymbol{H}^{-1}$ computed in the last step.

Both the DS and the DFP algorithms are standard algorithms which are described and discussed in more detail, e.g., by Press et al. [7].

4. Efficient Simulation

In this section, we investigate the computation time and the accuracy of the simulation with dependence on optimization algorithm, series expansion, and the main parameters workpiece discretization, N_L, and number of series terms, N_c. Based on this investigation, it is possible to do the simulation of DLOs efficiently, i.e., to compute the shape of the workpiece with sufficient precision in a (rather) short time. Since the simulation is based on the same principle approach as the simulation software described in [2, 3], we assume that most of the results presented here hold true for these works, too.

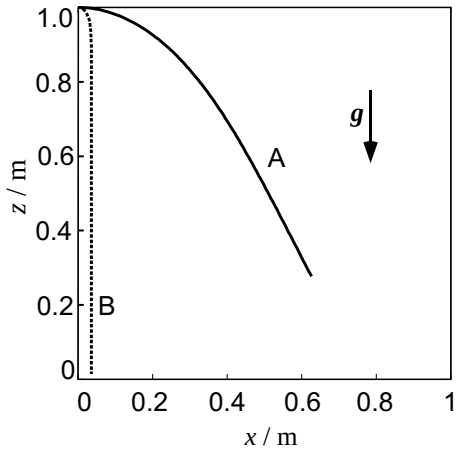

Figure 4: Two benchmarks A and B for investigating the computation precision

4.1 Computation Precision

In investigating the influence of the series expansion, the two benchmark problems shown in Figure 4 are used:

First, a copper wire of length $L = 1$ m and diameter $d = 1$ mm is fixed at one end in a Cartesian world coordinate system at $x = 0$ m, $z = 1$ m with horizontal orientation and bends due to gravity (benchmark A). Second, an additional load of 1 kg is mounted at the free end to increase the degree of bending (benchmark B). [14]

Figure 5 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 with the reference shape for $N_c \rightarrow N_{c,\,ref}$. The deviation increases with the degree of bending of the wire. However, the number of series terms required for obtaining high accuracy is considerably lower for Chebychev 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 Chebyshev polynomials converge faster for typical cases.

---

[14] 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.

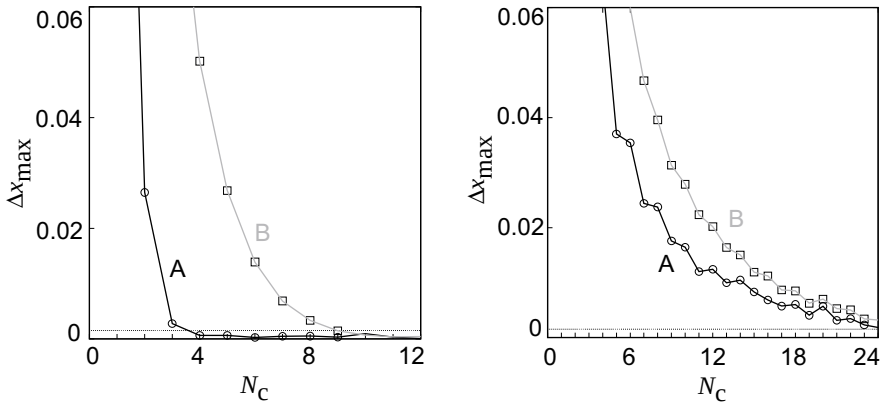

Figure 5: Maximal deviation Δx_{max} between computed and reference workpiece shape as a function of the number N_c of series terms for Chebyshev polynomials (left) and Fourier series (right)

Besides the number of series terms, the precision also depends on the discretization Δs (given by object length L and number of curve elements N_L) of the object in computing the energy integral given in Eqn. 1. For benchmark A described above, the maximal error is shown as a function of N_L in Figure 6 (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. With $N_L = 15$, the maximal accuracy is obtained. A further increase of N_L does not improve the accuracy.[15]

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. For the numerical integration, we approximate the object between the nodes by circular arcs. [16] Between the nodes, the difference to the reference shape may be higher than shown in Figure 5 and Figure 6.

For both optimization algorithms, Figure 8 shows the computation time as a function of the number of curve elements N_L with the number of series terms $N_c = 8$. In both cases, the computation time increases approximately linearly with N_L. Comparing with the downhill simplex algorithm, the DFP algorithm requires more evaluations of the energy integral in each iteration. Therefore, even a small differ-

---

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

16 For curved objects, an approximation by circular arcs (i.e., elements of constant curvature) are better suited than linear segments (of curvature 0). The kind of approximation especially influences the center of gravity of each segment, and, thus, the potential due to gravity (Eqn. 2a).

ence in the number of iterations has a significant impact 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.[17]

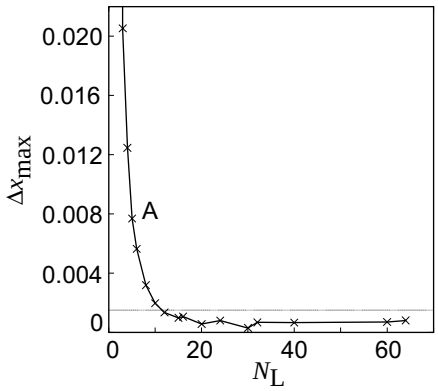

Figure 6: Maximal deviation Δx_{max} between computed workpiece shape and reference as a function of the number N_L of curve elements for benchmark A

4.2 Computation Time

The effort required for computing the shape of the workpiece is mainly determined by the combination of the 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. 1.

For investigating the computational effort, benchmark C shown in Figure 7 is used: The copper wire described above is gripped at one end point with gripper position $x = 0$ m, $z = 1$ m. With the gripper orientation being initially horizontal, the gripper is rotated by 180° about the y-axis with a stepsize of 10° and back to 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 using LINUX as operating system.

Figure 9 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 large. However, in the previous section 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.

---

[17] 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.

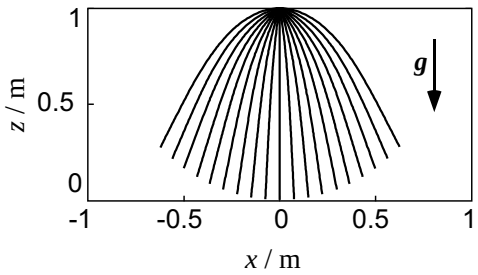

Figure 7: Benchmark C for investigating the computation time

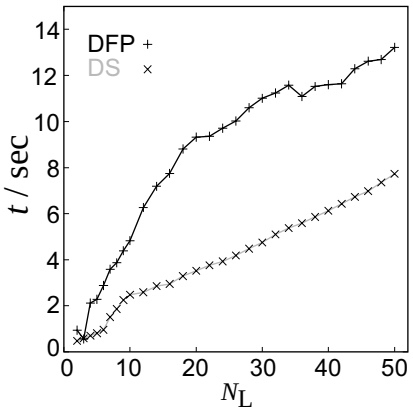

Figure 8: Computation time as a function of the number of curve elements N_L for downhill simplex (DS) and Davidon-Fletcher-Powell (DFP) algorithm

4.3 Parallel Computation

The previous section 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, we need to distinguish two different situations, which are discussed in the following.

1. Different shapes of the workpiece 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, consisting 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 [8] for details).

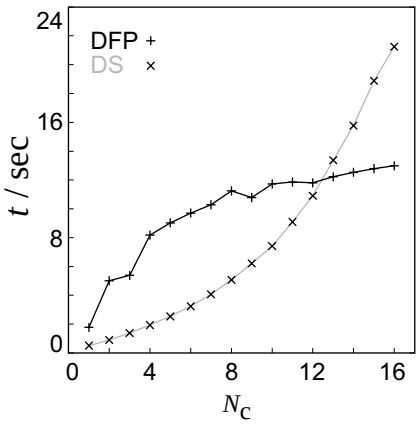

Figure 9: Computation time as a function of the number of series terms N_c for downhill simplex (DS) and Davidon-Fletcher-Powell (DFP) algorithm

4.3.4 Independent Computation

For independent computation, at least the following two basic conditions must be met: 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 discrepancy to the actual shape is typically smaller in this case. As expected, the resulting speedup is almost linear, as shown in Figure 10 for the benchmark C given in Figure 7.

However, if any interactions between workpiece and obstacles have to be considered, the computed object shapes may not be valid, even if the conditions given above are met. This problem is demonstrated in Figure 11 (left), with the following benchmark D. The object is moved downwards (into direction MD, direction of gravity) and collides with an obstacle. All steps in the simulation are computed independently from each other, the minimization algorithm is always started with an undeformed workpiece as initial guess. For the first four steps, the object shape is correctly simulated, but in the fifth step the object “jumps” to the lower side of the obstacle, 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.

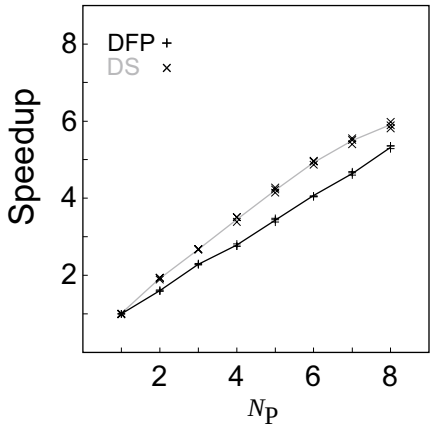

Figure 10: Measured speedup for independent computations using downhill simplex (DS) and Davidon-Fletcher-Powell (DFP) algorithm with N_p being the number of processors (each experiment performed three times)

The correct result is obtained if the object shape computed in step *i*–1 is taken as initial guess in step *i*, as shown in Figure 11 (right). 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.

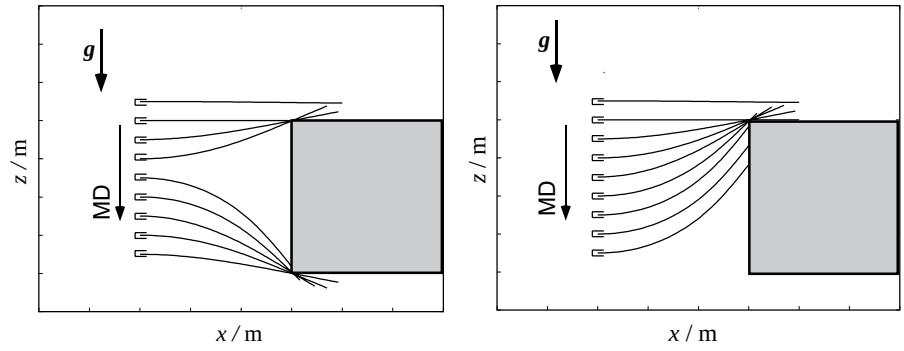

Figure 11: Benchmark D: Obstacle interaction of a deformable linear object while being moved into direction MD for independent computation of all steps (Steps 1 to 9 upwards down). Left: false result for step 5 to 9. Right: correct result

It is not always 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 is likely to occur.

4.3.5 Dependent Computation

If the object is deformed plastically while being handled by the robot, or if the simulation needs to be performed dynamically, the simulation of step i requires the results obtained in step $i-1$ (for example the velocity of each mass element) as input data for the computation. Therefore, the approach given above is not feasible.

Generally, it may be assumed that the final solution $\boldsymbol{Q}_i$ in simulation step i (with $t = i\,\Delta t$) can be computed fast, if the difference between the actual minimal energy and the initial guess used for the numerical optimization is small.[18] Based on this idea, we can compute a good initial guess for the steps i+1, i+2, ..., while computing the correct result $\boldsymbol{Q}_i$ for step i.

Let us assume the case of performing a dynamic simulation and having two independent computation tasks T_1 and T_2. Given a known initial value $\boldsymbol{Q}_0$ (i.e., the position and velocity of each workpiece point) of the object at time $t = 0$, we use the following algorithm:

Based on $\boldsymbol{Q}_0$, T_1 and T_2 compute simultaneously two shapes $\boldsymbol{Q}_{1,\,\Delta t}$ and $\boldsymbol{Q}_{2,\,2\Delta t}$ in simulation step $i = 1$. $\boldsymbol{Q}_{1,\,\Delta t}$ is computed with timestep Δt, $\boldsymbol{Q}_{2\,2\Delta t}$ is computed with timestep $2\Delta t$. While $\boldsymbol{Q}_{1,\,\Delta t}$ is the correct solution for $t = \Delta t$, i.e., $\boldsymbol{Q}_{1,\,\Delta t} = \boldsymbol{Q}_1$, $\boldsymbol{Q}_{2\,2\Delta t}$ is an approximation of $\boldsymbol{Q}_2$ at (shape of the workpiece at $t = 2\Delta t$). It is only an approximation, since the correct computation of $\boldsymbol{Q}_2$ requires $\boldsymbol{Q}_1$ as input data.

In simulation step $i = 2$, task T_2 continues its computation for $\boldsymbol{Q}_2$, using $\boldsymbol{Q}_{2\,2\Delta t}$ as inital guess and $\boldsymbol{Q}_1$ as input data. The result, $\boldsymbol{Q}_{2,\,\Delta t} = \boldsymbol{Q}_2$, is the correct solution for $t = 2\Delta t$. Simultaneously, T_1 computes an approximation $\boldsymbol{Q}_{3,\,2\Delta t}$ for $\boldsymbol{Q}_3$, and so on.

This approach can be easily extended to an arbitrary number of independent computation tasks. While one task computes the correct solution for $t = i\,\Delta t$ (using the timestep Δt), the other tasks compute approximations for $t = \{(i+1)\,\Delta t, (i+2)\Delta t, ...\}$.

However, the assumption that a good initial guess results in a short computation time for 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, such as DFP, on the other hand, come close to the minimum rather fast even if the initial guess is bad, but it requires many iterations to finally determine the minimum with the desired 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 shape of the workpiece, the DFP algorithm terminates immediately. This is the case if there is no plastic deformation in a step or if the acceleration of all mass elements is constant in time, respectively. Here, the speedup is lower than for independent computation, but also linear.

---

[18] $\boldsymbol{Q}_i(s)$ consists of the functions representing the workpiece shape $q_1(s)$, $q_2(s)$, $q_3(s)$, and the (Cartesian) velocity $\boldsymbol{v}(s)$ of each workpiece point in step i.

Figure 12 shows the measured speedup for a dynamic computation of the benchmark C as shown in Figure 7 for both minimization algorithms. Due to the characteristics of the downhill simplex algorithm described above, the speedup is almost negligible. For the DFP algorithm, a maximum speedup of about two is obtained for three computation tasks. If the number of tasks is further increased, the speedup decreases 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 benefit of the tasks becomes smaller from task to task. Second, the effort required for communication increases with the number of computation tasks.

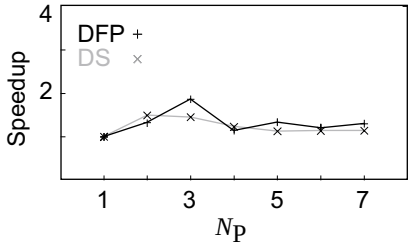

Figure 12: Measured speedup for dynamic (dependent) computation using downhill simplex (DS) and Davidon-Fletcher-Powell (DFP) algorithm with N_p being the number of processors (each experiment performed three times)

Comparing with the resulting speedup, it is found that the effort for the parallel computation is 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 we expected speedup to be poor.

5. Inverse Simulation

5.1 Approach

So far, we have discussed the problem of simulating the behavior of the workpiece if the gripper trajectory (and possibly other boundary conditions like obstacles) are given. We call such problems "*Direct Simulation Problems*". They occur, e.g., when different handling strategies are compared or when the impact of the material parameters is studied.

However, if we think of using a simulation system for robot programming (i.e., generating the gripper trajectory for a given task), the problem is just inverse: Given some boundary conditions concerning the position and shape of the workpiece, the

gripper trajectory shall be computed. We call this kind of problems "*Inverse Simulation Problems*". Till now, it has not been investigated systematically.

The simplest example is the threading of a DLO through a cut-out, e.g., in a sheet metal. The optimal solution for this task is a gripper trajectory which meets the following conditions for the compete threading process, guaranteeing maximal tolerance to all kinds of uncertainties or distortions:

- The DLO pierces the sheet metal plane at the center point $\boldsymbol{P}_{\text{Goal}}$ of the cut-out, and
- the orientation of the DLO at $\boldsymbol{P}_{\text{Goal}}$ is aligned with the normal $\boldsymbol{n}_{\text{Goal}}$ of the sheet metal.

To solve this problem, we recall the solution to the direct simulation problem, as described above. To do this, we introduce a penalty function U_{Penalty}, describing the deviation between given boundary conditions and actual DLO shape. By changing the position $\boldsymbol{P}_{\text{Gripper}}$ and orientation $\boldsymbol{n}_{\text{Gripper}}$ of the gripper, U_{Penalty} is iteratively minimized. This approach leads to the following algorithm.

```
1    P_Gripper := P_Gripper, 0;  {Initial guess for  gripper position & orientation}
     n_Gripper := n_Gripper, 0;
2    repeat  {Main loop for inverse simulation problem}
3        Solved := false;
4        Deviation := U_Penalty(Q, P_Goal, n_Goal);
5        if Deviation = 0 then  {solved}
                Solved := true
         else   {not solved}
                P_Gripper := P_Gripper + ΔP_Gripper;  {varying gripper position
                n_Gripper := n_Gripper + Δn_Gripper;    and orientation}
6    until Solved;
```

In this algorithm, $\boldsymbol{Q}$ is the shape of the workpiece, given by the vectors of all line elements, $\Delta\boldsymbol{P}_{\text{Gripper}}$ and $\Delta\boldsymbol{n}_{\text{Gripper}}$ are the change of the gripper position and orientation in each iteration.

In this approach, we use two interlaced optimization processes. The "inner" one for computing the DLO shape for a given gripper position (direct simulation problem), and the "outer" one for determining the gripper position which solves the inverse problem. With this approach, it is possible to consider any kind of boundary conditions by an appropriate selection of U_{Penalty}.

5.2 Solution for Important Special Cases

However, there are important special cases in which the problem can be solved much easier. As long as no interaction between workpiece and environment has to be considered, the shape of the DLO is only affected by the orientation of the gripper with respect to gravity, but it is independent of the absolute gripper position.

Therefore, the following strategy can be used for solving the inverse simulation problem:

Given $\boldsymbol{P}_{\text{Goal}}$, $\boldsymbol{n}_{\text{Goal}}$ and the line element $n \in [0, 1, ..., N_{\text{L}}-1]$ of the DLO which shall pierce the cutout, $\boldsymbol{P}_{\text{Gripper}}$ is fixed at an arbitrary position, e.g., the origin of the global coordinate system, and $\boldsymbol{n}_{\text{Gripper}}$ is varied until the DLO tangent is aligned with $\boldsymbol{n}_{\text{Goal}}$. With $\boldsymbol{P}_n'$ being the position of DLO element n, the gripper must then be displaced to

$$\boldsymbol{P}_{\text{Gripper}} = \boldsymbol{P}_{\text{Goal}} - \boldsymbol{P}_n'$$

in order to finally solve the problem. With this approach, the algorithm given above is applied only to determine the gripper orientation, while the gripper position can be computed directly. As long as the curve describing the shape of the DLO lies within a plane (which holds approximately true in many cases), the problem does not have to be regarded in three dimensions, but is two-dimensional. In this case, only one angle θ is required for determining the gripper orientation. Thus, the "outer" optimization problem in the algorithm given above is an one-dimensional optimization for θ. Being $\boldsymbol{t}_n$ the tangent vector of the DLO in point n, a quadratic penalty function

$$U_{\text{Penalty}} = c\,(\arccos(\boldsymbol{t}_n \boldsymbol{n}_{\text{Goal}}))^2$$

is found to be suited for the optimization, with c being an adjustment constant.

The left column of Figure 13 shows simulation examples for the threading of flexible beams with different bending rigidity through a cut-out with the normal of the sheet metal plane being perpendicular to gravity. The increase of bending due to gravity for decreasing bending rigidity (from top to bottom) is obvious.

Please note that in Zheng et al. [1], the task of inserting a flexible beam into a bore hole of approximately equal diameter is regarded. In that case, the gripper trajectory is found to be equivalent to the deflection curve of the bending beam. However, this is no general solution for the inverse simulation problem. For the task described in [1], the already inserted portion of the beam does not have to be considered any longer, because its weight is compensated by the walls of the bore hole. In other tasks, e.g., the threading task considered here, the complete beam is deflected by gravity while the task is performed. The relation between the gripper trajectory for the threading task and the insertion into a bore hole is shown in the right column of Figure 13. In each figure, the upper curve represents the DLO shape for the beginning of the threading process, i.e., the free end of the DLO being at position $\boldsymbol{P}_{\text{Goal}}$, having the orientation $\boldsymbol{n}_{\text{Goal}}$. For the insertion task described by Zheng et al., this curve defines the complete gripper trajectory. In the contrary, the lower curves in the figures give the gripper trajectory for the threading tasks shown in the left column. Obviously, the trajectory is different for both tasks, though both $\boldsymbol{P}_{\text{Gripper}}$ and $\boldsymbol{n}_{\text{Gripper}}$ are identical in the beginning.

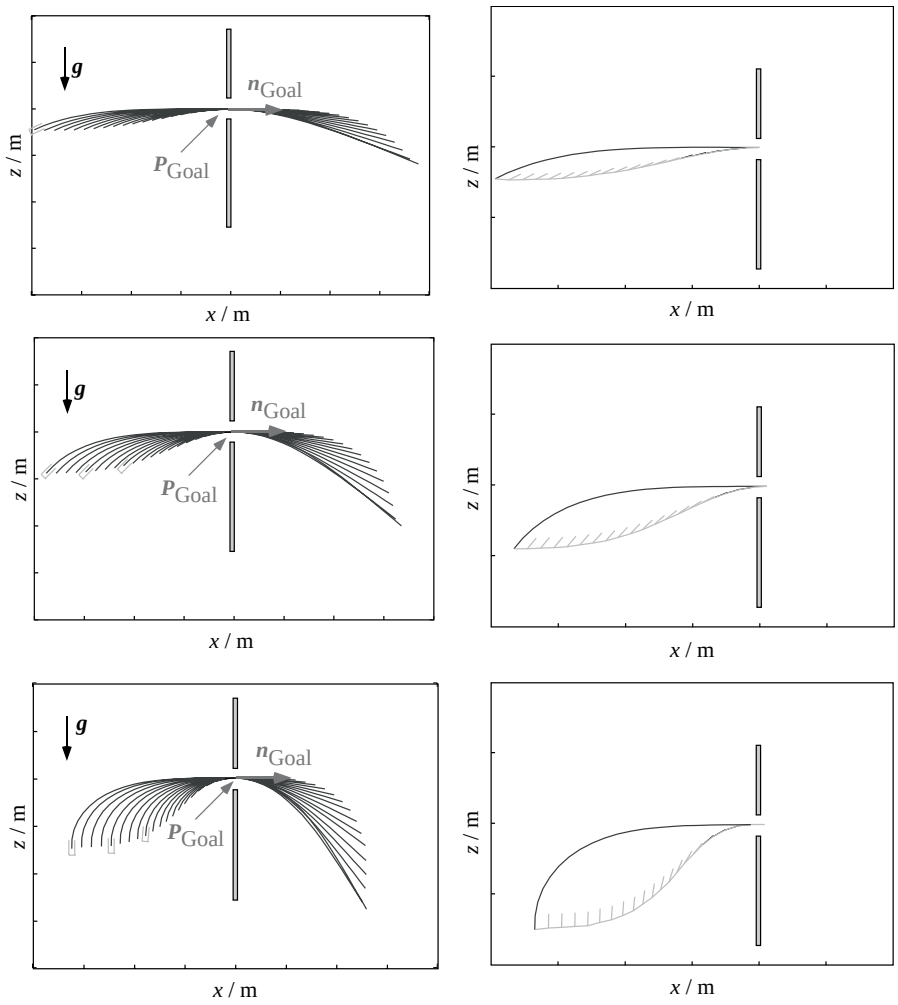

Figure 13: Threading of a bending wire of length L = 1 m through a cut-out with center x_{Goal} = 0 m, z_{Goal} = 1 m for different bending rigidities (decreasing from top to bottom). Figures left: Wire shape during the threading process. Figures right: Initial wire shape (upper curve) and gripper trajectory with gripper orientation indicated by the short lines (lower curve).

In the task of threading through a cut-out with a given orientation, the required position and orientation of the gripper are defined unequivocally for each point of its trajectory. Instead of giving the position and orientation of one point as constraints, it is also possible to give two goal positions $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ without orientation. An example for this kind of task is the threading through two cut-outs. In this case, it is desired to guide the DLO simultaneously through the centers of both cut-outs. Again, it is not necessary to determine both position and orientation by means of

numerical optimization, but the gripper displacement can be computed directly. An example is shown in Figure 15.

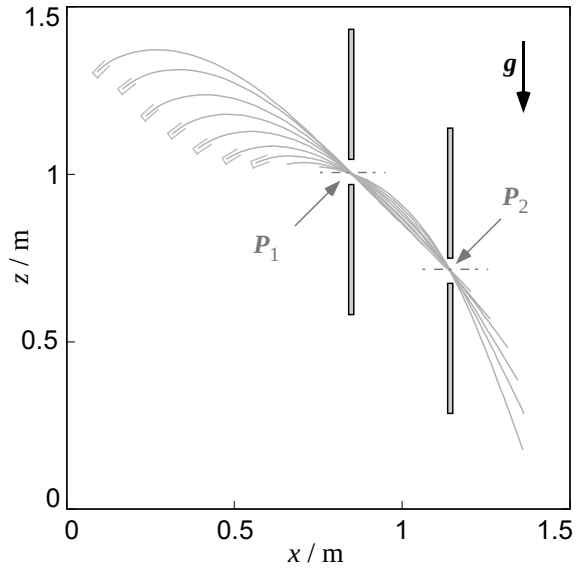

Figure 14: Example for inverse simulation with two desired goal positions $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$: Threading through two cut-outs

5.3 Outlook

Comparing with the general algorithm described above, the computation can be simplified in the cases discussed here because the workpiece shape is independent from the gripper position. This holds true as long as the DLO is deformed only by gravity and no direct contact between DLO and environment occurs. If this condition is not fulfilled, the situation becomes more complex.

As example, Figure 15 shows the numerical simulation of an experiment described by Henrich et al. [9]: The DLO is in contact with an obstacle and is being deformed by the contact force. The gripper is moved on a trajectory which iteratively reduces the DLO length between gripper and contact point, while the DLO orientation $\boldsymbol{n}_{\text{Goal}}$ is kept constant.[19]

However, another experiment described in [9] using the same setup states that for an arbitrary contact point along the DLO length and a given DLO orientation in the contact point, a gripper trajectory exists which

- retains the contact during the motion, and

[19] In this example, the gripper trajectory is given by a straight line, connecting the initial gripper position and the contact point. The gripper orientation is constant [9].

- ensures that neither the contact point along the DLO length nor the orientation of the DLO in the contact point is altered.

This means that for any given contact point along the DLO and given orientation at the contact point, the solution of the inverse problem is not unambiguous, but the number of solutions is infinite. This is demonstrated in the example shown in Figure 16 for a horizontal orientation $\boldsymbol{n}_{\text{Goal}}$ of the DLO in the contact point.

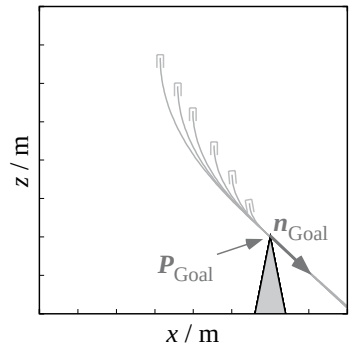

Figure 15: Example for inverse modeling problem if contact between workpiece and environment must be considered. The DLO length between gripper and contact point $\boldsymbol{P}_{\text{Goal}}$ is iteratively reduced while the orientation in $\boldsymbol{P}_{\text{Goal}}$ is kept constant.

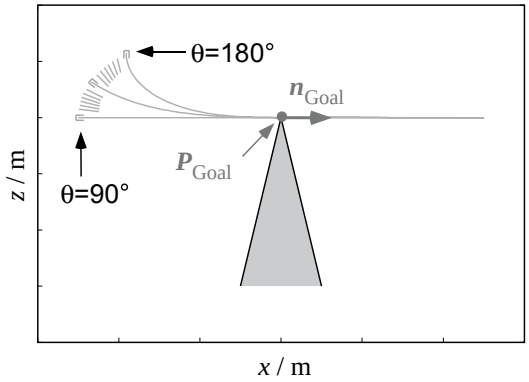

Figure 16: Different solutions for the inverse modeling problem with contact between workpiece and obstacle. All of the gripper positions shown here (and all points on the trajectory connecting them) are valid solutions for the inverse simulation problem.

Because it is generally not clear what solution the algorithm given above will find, it is required to add additional boundary conditions in such cases, which guarantee an unequivocal solution. The problem of solving the inverse simulation problem with contact between workpiece and environment is currently being investigated.

6. References

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

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

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

[4] 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.

[5] Ritz, W.: "Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik". Journal für die reine und angewandte Mathematik", vol. 135, 1909.

[6] Nakagaki, H., et al., "Study of deformation and insertion tasks of a flexible wire", In: Proc. 1997 Int. Conf. on Robotics and Automation, vol. 3, pp. 2397-2402, Albuquerque, USA, April 1997.

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

[8] 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.

[9] Henrich D., Ogasawara T., and Wörn H.: "Manipulating deformable linear objects: Contact states and point contacts". In: Proc. 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99), Porto, Portugal, July 21-24, 1999.