Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14195 Berlin

Marc C. Steinbach

Optimal Motion Design Using Inverse Dynamics

Preprint SC 97–12 (March 1997)

Optimal Motion Design Using Inverse Dynamics

Marc C. Steinbach

March 21, 1997

Abstract. Increasing demands on industrial robot operation call for optimal motion planning based on dynamic models. The resulting mathematical problems can be handled efficiently by sparse direct boundary value problem methods. Within this framework we propose a new solution technique that is closely related to the conventional kinematic approach: it eliminates the need for numerical integration of differential equations. First optimization results on a real life transport maneuver demonstrate that the technique may save over 50 % computation time.

Key words: Optimal motion planning, robot dynamics, inverse dynamics, direct boundary value problem approach.

1 Introduction

Robot manipulators play an important role in modern industrial manufacturing. The main reason is probably that robots are freely programmable, and thus offer great flexibility to be used for various tasks. To exploit their full potential, sophisticated motion planning techniques are required.

Modern CAD-based off-line programming systems provide a convenient environment for interactive motion design, including animated simulation of the maneuver and collision checks in a CAD model of the robot workcell. This reduces the need for manual on site corrections when downloading the final program, thus saving time and cutting down production losses. The motion designer can program a maneuver in terms of high level instructions. Typically a sequence of *locations* on the path is specified, each defining a position and orientation for the tool center point (TCP). Additional motion parameters are then used to restrict the TCP speed or acceleration on a path segment, to specify certain motion types between intermediate locations (such as linear or circular), or to influence the motion in other ways.

The motion planning system performs its job based on a *kinematic* robot model which includes the robot geometry and limits on the joint angles, speeds, and accelerations. Slightly simplified, the task of kinematic motion planning can thus be stated as follows: Given a sequence of locations and additional motion parameters, produce a path that connects the locations, then determine a velocity profile that satisfies the speed and acceleration constraints along the path. By default, the velocity profile will usually be chosen so as to minimize the resulting maneuver time.

Nonlinear *dynamic* effects such as centrifugal and Coriolis forces are basically ignored by the motion planning system. Instead, they are handled by the feedback controllers (typically linear controllers with rate feedback) which execute incremental position commands in small time steps to steer the robot along the given path. This makes the kinematic motion design concept inadequate for time-critical high speed maneuvers that are dominated by the nonlinear dynamic forces. Standard speed and acceleration limits will have to be exceeded, and tracking errors will often occur. It is therefore natural to consider trajectory optimization techniques based on dynamic robot models for motion design.

Optimal robot path planning has received considerable attention during the past decades, see, e.g., [9, 2, 15, 18] or the survey [5], and the references therein. The majority of these papers are concerned with time minimization along a prescribed path. A series of studies by the author and his colleagues investigate basic maneuvers of basic 2- or 3-joint robots to gain physical insight [10, 11, 23, 24, 8, 14]. Related work on path planning for satellite mounted robots can be found in [17, 13]. These references are all concerned with *point-to-point* (PTP) trajectory design where the path itself is free for optimization. In real life applications, PTP trajectory planning generally leads to difficult nonlinear optimization problems involving the robot's forward dynamics equations and geometric path constraints. Efficient mathematical algorithms for such problems have recently been developed in an industry project [4, 22]; the results for a typical transport maneuver (cf. Sec. 4) show that optimal, collision-free trajectories can be computed in a few minutes.

The paper is organized as follows. In section 2 we state the trajectory planning problem in *forward dynamics* formulation and outline its numerical treatment by the direct BVP approach. In section 3 we introduce our new technique of *inverse dynamics* optimization which can be seen as a direct generalization of conventional *kinematic* motion design. The new technique is applied to the optimization of a real life robot maneuver in section 4, showing that it can lead to substantial savings of computation time. Conclusions and future directions of research are indicated in section 5.

2 Forward Dynamics Trajectory Optimization

In this paper we consider trajectory optimization problems (TOP) in the following abstract form. Given a fixed interval $I = [0, T]$ and interior points $0 = t_1 < \cdots < t_k = T$, the aim is to find a state function x and a control function u that minimize a Mayer cost function ϕ subject to ordinary differential equations, path inequality constraints, and multipoint boundary conditions:

$$
\text{[TOP]} \qquad
\begin{aligned}
\phi(x(T)) &= \min_{x,u} && \text{(1a)}\\
\dot{x}(t) - f(x(t), u(t)) &= 0 && \text{(1b)}\\
g(x(t)) &\in [g_{\min}, g_{\max}] && \text{(1c)}\\
x(t) &\in [x_{\min}, x_{\max}] && \text{(1d)}\\
u(t) &\in [u_{\min}, u_{\max}] && \text{(1e)}\\
r_1(x(t_1)) + \cdots + r_k(x(t_k)) &= 0 && \text{(1f)}
\end{aligned}
$$

The case with free final time T and other common generalizations can be reduced to this form by simple transformations, see [20].

In many cases ODE (1b) will be the standard reduction of a higher order ODE, $x^{(p)}(t) = f_p(x(t), u(t))$. Then $f(x, u) = Ax + Bf_p(x, u)$, where

$$
x = \begin{pmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(p-1)} \end{pmatrix}, \qquad A = \begin{pmatrix} 0 & I & & \\ & \ddots & \ddots & \\ & & 0 & I \\ & & & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ I \end{pmatrix}. \tag{2}
$$

For instance, we are principally interested in mechanical systems that are controlled by external forces, with robots as a special case and other important applications in aerospace engineering or vehicle dynamics. Such systems are governed by Newton's law

$$M(q)\ddot{q} = F(q, \dot{q}) + u,$$

from which the second order ODE occurring in [TOP] is readily obtained by inverting the positive definite mass matrix,

$$\ddot{q} = M(q)^{-1}[F(q, \dot{q}) + u].$$

These equations are known as *forward dynamics equations*; they express the accelerations in terms of external forces u and states $x = (q, \dot{q})$. Conversely, the *inverse dynamics equations* express the forces in terms of accelerations $\ddot{q}$ and states $(q, \dot{q})$,

$$u = M(q)\ddot{q} - F(q, \dot{q}).$$

Taking a more abstract point of view, we call $x^{(p)} = f_p(x, u)$ the *forward dynamics equations* of the dynamic process (2) and generalize the notion of inverse dynamics accordingly.

Definition (Inverse dynamics). If for every x

$$f_p(x, \,\cdot\,) \colon u \to f_p(x, u)$$

is a bijection with inverse mapping

$$\bar{u}(x, \,\cdot\,) \colon v \to \bar{u}(x, v),$$

then we call $u = \bar{u}(x, x^{(p)})$ the *inverse dynamics equations* of process (2).

The existence of inverse dynamics for ODE (1b) is precisely the property that is needed to apply the new technique proposed below. It requires a direct influence of the control u on every motion degree of freedom, implying that the dimensions of u and $x^{(0)}$ agree. Note that the mechanical equations above are naturally given in inverse dynamics form, whereas the forward dynamics form involves a matrix inversion.

The Direct Boundary Value Problem Approach

The direct BVP approach for problem [TOP] combines a piecewise control parameterization on some grid with a piecewise state discretization via collocation or multiple shooting on a second grid. In the following we omit technical details and assume that these two grids are identical.

For control parameterization we choose a grid $\Gamma \colon 0 = \tau_1 < \cdots < \tau_m = T$ with $m \geq k$ nodes as a refinement of the grid $\{t_j\}$ on which boundary and interior point conditions are specified. For simplicity, let us assume that [TOP] was already formulated with respect to Γ, hence $k = m$ in (1f) and $r_j = 0$ at the additional nodes. We denote subintervals by $I_j = (t_j, t_{j+1})$ and their respective lengths by $\Delta t_j = t_{j+1} - t_j$.

Next, control functions are specified piecewise via free control parameters v_j and fixed base functions u_j with local support, $u(t) = u_j(t, v_j)$ on I_j. That is, u is restricted to some finite-dimensional space of admissible controls independently on each subinterval. Control jumps are permitted at the nodes t_j. This yields the parameterized trajectory optimization problem

$$
\text{[POP]} \qquad
\begin{aligned}
\phi(x(T)) &= \min_{x,\{v_j\}} && \text{(3a)}\\
\dot{x}(t) - f(x(t), u_j(t, v_j)) &= 0 && \text{(3b)}\\
g(x(t)) &\in [g_{\min}, g_{\max}] && \text{(3c)}\\
x(t) &\in [x_{\min}, x_{\max}] && \text{(3d)}\\
u_j(t, v_j) &\in [u_{\min}, u_{\max}] && \text{(3e)}\\
r_1(x(t_1)) + \cdots + r_m(x(t_m)) &= 0 && \text{(3f)}
\end{aligned}
$$

For the state discretization only multiple shooting will be considered here. We introduce local initial values x_j at the grid nodes and calculate trajectory approximations p_j by numerical integration,

$$p_j(t, x_j, v_j) \approx x_j + \int_{t_j}^{t} f(p_j(s, x_j, v_j), u_j(s, v_j))\, ds \quad \text{on } I_j.$$

Global continuity of the piecewise trajectory representation is ensured by *connection conditions*

$$h_j(x_j, v_j, x_{j+1}) := p_j(t_{j+1}, x_j, v_j) - x_{j+1} = 0, \qquad j = 1(1)m-1.$$

The cost function ϕ and boundary conditions r_j are already given in discrete form, and the path constraints (3c–3e) are finally "discretized" by specifying them at the nodes only.

The local problem functions ϕ, h_j, g, u_j, r_j constitute a large, sparse, constrained nonlinear optimization problem (NLP) in the variables $y = (x_1, u_1, \ldots, u_{m-1}, x_m)$. Starting from a discrete initial trajectory y^0, the NLP can be solved iteratively by sequential quadratic programming (SQP), where each correction Δy^k is the solution of a linear-quadratic subproblem (QP) which approximates the NLP around iterate y^k. First and second derivatives of all problem functions appear in the QP; the latter are usually accumulated by rank-2 updates during the iteration because exact values are inaccessible or too expensive.

The direct BVP approach was first proposed by Bock and Plitt [3, 16] and implemented in the multiple shooting code MUSCOD. Some related methods that were developed later on can be found in [1, 7, 25]. The specific *mulistage block-sparse* QP structure induced by the discretization is fully exploited by the sparse direct SQP method described in [19, 20, 21]; its implementation in the multistage trajectory optimization package MSTOP proved highly efficient in optimal robot motion planning [22].

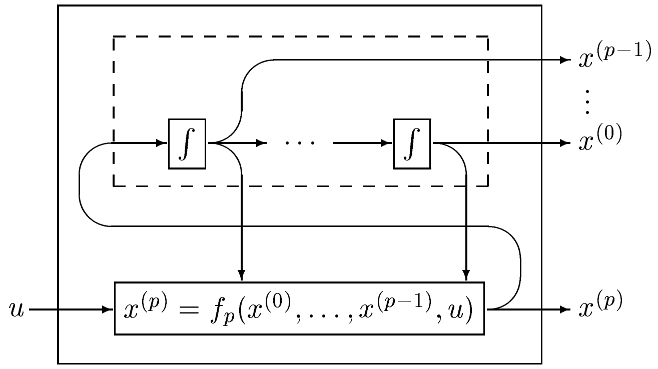

Figure 1: Forward dynamics control flow

Discussion

The great flexibility of the direct BVP approach raises questions as how to choose the grid(s), base functions, and integration schemes. Naturally, the answers depend on the problem (class) to be solved, and on the desired *optimality* and *accuracy* of the solution, but we can give a few guidelines.

The control parameterization basically determines *optimality*, i.e., how close one can get to the true optimum. Under standard regularity assumptions the objective value will improve if the grid is refined, or if the base functions are chosen from higher-dimensional spaces. Time-optimal robot motion typically exhibits bang-bang-type solutions, so it is appropriate to use piecewise constant base functions on relatively fine control grids. In this case the control constraints (3e) can be satisfied globally, $v_j \in [u_{\min}, u_{\max}]$. If jumps of the control torques are undesirable and a higher smoothless level is required, say $u \in C^q$, then we choose $u^{(q+1)}$ piecewise constant and treat the polynomials $u^{(0)}, \ldots, u^{(q)}$ as additional state components. (Otherwise the control coupling would change the QP structure.)

Discretization errors in the state variables are essentially grid-independent; their *accuracy* can be controlled via stepsize and order selection of the integration scheme. Violations of the state constraints are finally influenced by the (state) grid, again with smaller errors on finer grids.

Numerical experience in optimal robot motion planning with MSTOP shows that the integration of forward dynamics equations in evaluating functions and gradients for the QP may be very expensive as compared to the QP solution. This leads us to the idea of reformulating the optization problem as described in the following section.

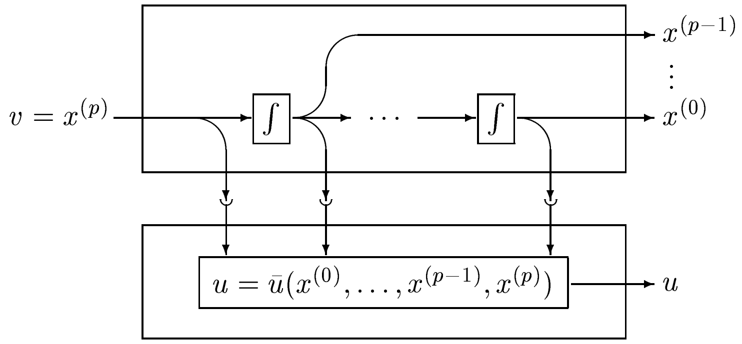

Figure 2: Inverse dynamics control flow

3 Inverse Dynamics Trajectory Optimization

In the original trajectory optimization problem [TOP], the physical control u is the system input which produces $x^{(p)} = f_p(x, u)$ as an output, see Fig. 1. Our new technique employs the inverse dynamics equations to exchange these roles. We consider the derivative $v = x^{(p)}$ as the system input and $u = \bar{u}(x, v)$ as an output, see Fig. 2. The new differential equation

$$\dot{x} = Ax + Bv$$

is then *linear* with constant coefficients A, B from (2), and [TOP] is transformed into the *equivalent* inverse dynamics optimization problem

$$\text{[iTOP]} \qquad \begin{aligned} \phi(x(T)) &= \min_{x,v} && \text{(4a)} \\ \dot{x}(t) - Ax(t) - Bv(t) &= 0 && \text{(4b)} \\ g(x(t)) &\in [g_{\min}, g_{\max}] && \text{(4c)} \\ x(t) &\in [x_{\min}, x_{\max}] && \text{(4d)} \\ \bar{u}(x(t), v(t)) &\in [u_{\min}, u_{\max}] && \text{(4e)} \\ r_1(x(t_1)) + \cdots + r_m(x(t_m)) &= 0 && \text{(4f)} \end{aligned}$$

We have paid the price of replacing simple control bounds by mixed nonlinear path constraints (4e) involving the inverse dynamics equations. However, this creates only a slight complication and one may hope that it is outweighed by the great simplification of the differential equation. Note also that we obtain a formal statement of the *kinematic* motion planning problem with no dynamics at all, if the state-dependent control constraints (4e) are replaced by suitable constant bounds $v \in [v_{\min}, v_{\max}]$. This corresponds exactly to the upper part in Fig. 2.

For the linear ODE (4b) one can immediately write down the explicit IVP solution in terms of $x_0 = x(0)$ and the control function v,

$$x(t) = \exp(tA) \left[x_0 + \int_0^t \exp(-sA) B v(s)\, ds \right],$$

where

$$\exp(tA) = \begin{pmatrix} I & tI & \cdots & \frac{t^{p-1}}{(p-1)!} I \\ & I & \cdots & \frac{t^{p-2}}{(p-2)!} I \\ & & \ddots & \vdots \\ & & & I \end{pmatrix} =: G^x(t).$$

Parameterization

Now let us restrict parameterizations for [iTOP] to piecewise constant controls, $v(t) \equiv v_j$ on I_j. Then it is easy to calculate the control influence on the state variables by successive integration, yielding

$$\exp(tA) \int_0^t \exp(-sA) B\, ds = \begin{pmatrix} \frac{t^p}{p!} I \\ \vdots \\ \frac{t^2}{2} I \\ tI \end{pmatrix} =: G^u(t).$$

This makes a state discretization obsolete since we have exact QP data

$$h_j(t_{j+1}, x_j, v_j) = G^x(\Delta t_j) x_j + G^u(\Delta t_j) v_j - x_{j+1}.$$

Note that the remaining problem functions $\phi, g, r_j, \bar{u}$ are likely to be given analytically as well, so that the elimination of dynamics from the differential equation enables us to supply exact values for all data needed in the NLP, "exact" meaning the absence of discretization errors. Note also that piecewise polynomials $v \in \Pi^{q+1}$ may be used for parameterization: the derivative $v^{(q+1)}$ is piecewise constant, and ODE (4b) remains formally unchanged if we regard $v^{(0)}, \ldots, v^{(q)}$ as additional state components $x^{(p)}, \ldots, x^{(p+q)}$.

Interpretation

Basically, the inverse dynamics problem [iTOP] decouples the nonlinear system dynamics from the differential equation (see Fig. 2) which now expresses

Figure 3: The robot KUKA IR 761.

a purely geometric relation between the state $x^{(0)}$ and its derivatives $x^{(i)}$. From this point of view we may assume that the position $x^{(0)}$ is controlled directly; the inverse dynamics problem [iTOP] is then seen as a kinematic motion design problem with the usual limits on derivatives, and additional dynamic constraints. This view immediately suggests that one might solve the purely kinematic problem first (with (4e) replaced by appropriate bounds on v), and use the result as initial trajectory for the iterative solution of [iTOP]. The kinematic optimization can be expected to converge quickly, as is confirmed by the numerical results in Sec. 4.

To conclude this section we observe that the inverse dynamics problem [iTOP] (parameterized with piecewise constant control v_j) is obtained as the result of a special (implicit) control parameterization for [TOP], namely the unique one that produces the simplest possible form of the ODE: $u_j(t, v_j) = \bar{u}(x(t), v_j)$. Any other parameterization of v for [iTOP] yields a corresponding parameterization of u for [TOP] in the same way. (Of course, defining base functions implicitly in terms of the states x does not make sense in any other situation.)

4 A Case Study in Robot Motion Planning

In the following we consider a robot motion design problem that is described in detail in [4, 22]. The task under consideration is a time-critical transport maneuver on a Mercedes-Benz production line consisting of about ten hydraulic presses. A KUKA IR 761 robot (Fig. 3) equipped with an arm extension and a pneumatic gripper carries partially processed car body parts such as doors or mudguards between two adjacent presses; we call this maneuver

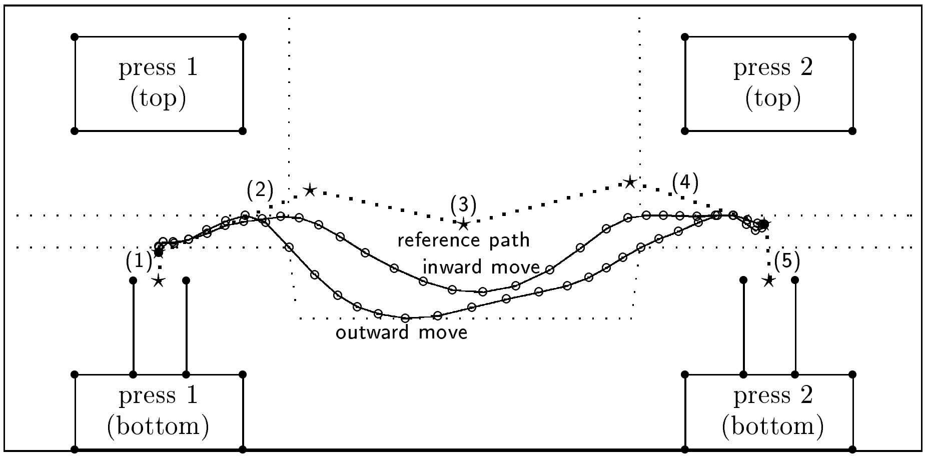

Figure 4: Front view of the press connection workcell (not to scale)

the *press connection*. Figures 4 and 5 depict vertical and horizontal cross sections through the robot workcell, respectively.

The robot's equations of motion can be written

$$M(q)\ddot{q} = F(q, \dot{q}) + u = C(q)(\dot{q}, \dot{q}) + \gamma(q) + u, \tag{5}$$

where $q \in \mathbb{R}^6$ are the joint angles, $u \in \mathbb{R}^6$ are the actuator torques, and $C(q)(\dot{q}, \dot{q})$ and $\gamma(q)$ denote the position-dependent gyroscopic and gravitational torques, respectively. This model includes the multibody dynamics for six links and the gripper with load, the rotor inertias (scaled up by gear ratios), and a passive pneumatic weight compensation system on the KUKA IR 761 robot. Friction is taken into account by reducing the nominal torque limits, $u_{i,\max} = (1 - c_i)u_{i,\mathrm{nom}}$. The empirical loss coefficients c_i are supplied by KUKA; they can be seen as safety margins allowing the feedback controllers to compensate for friction (and possibly other disturbances).

Based on model (5) we can compute an optimal torque profile $u(t)$ and optimal trajectory values $q(t), \dot{q}(t), \ddot{q}(t)$, where the joint angles and speeds are limited by simple bounds. Nonlinear geometric restrictions of the form $g(q) \in [g_{\min}, g_{\max}]$ ensure a safety distance of about 10 cm to prevent collisions of the load with the presses or the robot; the resulting feasible region for the tool center point is bounded by the fine dotted lines in Figs. 4 and 5.

The feedback controllers of a typical robot will usually not be able to execute the optimal trajectory $q(t)$ directly; therefore we have to back-calculate

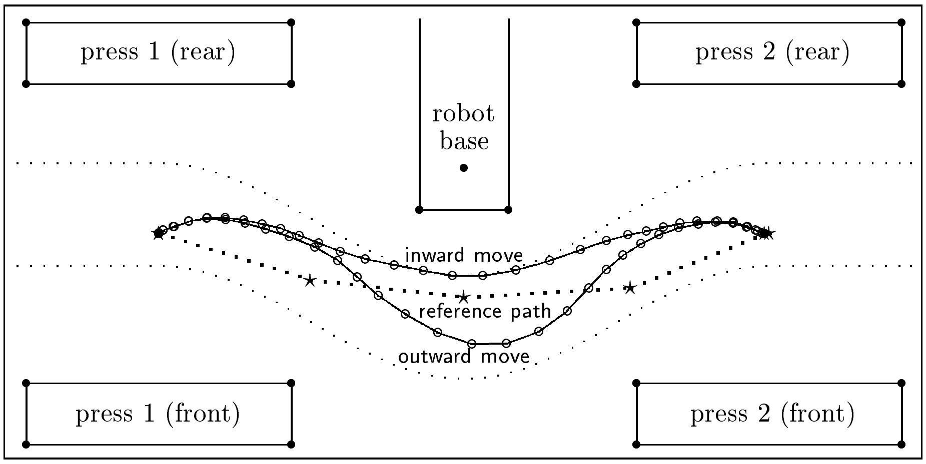

Figure 5: Top view of the press connection workcell (not to scale)

a trajectory $q_c(t)$ that inverts the controller and motor dynamics [12]. When commanded to the real robot, $q_c(t)$ will produce the desired torque profile $u(t)$ and optimal trajectory $q(t)$ with high accuracy, in certain cases requiring that $u \in C^0$ or even $u \in C^1$.

Let us now turn to the optimization problem. In [22] we compute optimal trajectories using the multistage trajectory optimization package MSTOP. The optimization runs are performed with piecewise constant torque profiles on uniform grids consisting of 20, 32, or 40 intervals, and with different termination accuracies (measuring the expected objective decrease in the next SQP step plus a weighted sum of constraint violations). A single step of the classical order four Runge-Kutta method is performed on each interval. A comparison with a high accuracy integration shows that this is just accurate enough, yielding position errors smaller than 5 mm in all cases. Even for such a low accuracy the integration of forward dynamics equations (and differentiation with respect to initial values and control torques) dominates the computations by far. To be precise, the evaluation of functions and gradients for the QP subproblems takes about 90 % of the total CPU time in all cases, while QP solution needs less than 10%! (The overhead of the SQP method is negligible.) Thus, the evaluation of forward dynamics limits the speed of computations. As it is well known that inverse dynamics calculations are significantly less expensive than forward dynamics [6], our new technique appears particularly promising in the present situation.

Optimization runs are again performed with the package MSTOP on an Iris Indigo workstation with a 100 MHz R4000/R4010 processor. We compare (1) forward dynamics results from [22], (2) inverse dynamics results using the same initial data, and (3) inverse dynamics results using as initial data the optimal solution of the kinematic problem (as suggested in Sec. 3). From the two possible motion types found in [22] (see Figs. 4 and 5) we consider only the "inward move" which is structurally identical to the maneuver on the real production line; it keeps the load closer to the robot base than the "outward move" and produces smaller centrifugal forces. For direct comparison we just copy the earlier optimization results for case (1) with termination accuracies $\tau \in \{10^{-3}, 10^{-4}\}$ and add a new set of runs with $\tau = 10^{-2}$. The corresponding value in cases (2) and (3) is always set to $\tau' = \tau/4$ for reasons to be explained below. Table 1 lists the complete set of results.

A view at the total CPU times shows that the inverse dynamics variants (2), (3) always outperform the forward dynamics optimization, achieving the most drastical savings for large τ on coarse grids. The kinematic optimization for initial trajectory generation is always beneficial except for one case on the list. Reductions of computation time range from 25 % to over 40 % in case (2), and up to 60 % in case (3).

In [22] we noticed a nearly constant average ratio of about 10 minor iterations (for the QP solution) per SQP iteration. Interestingly, the same ratio is also observed in the inverse dynamics cases (2) and (3), but now the time fraction for function and gradient evaluation is 15 % only and QP solution dominates the computation, consuming about 85 % of the total time.

Finally, we note that in cases (2) and (3) the SQP method needs significantly more iterations to converge than in case (1). On the 40 interval grid the inverse dynamics optimization does not converge at all in case (2). The slower convergence explains why we have set the termination accuracy to $\tau' = \tau/4$ in these cases: otherwise the SQP method would terminate with larger values of the maneuver time than in case (1) due to the smaller objective decrease per iteration. To find a reason for the slow convergence itself, recall that second derivatives are approximated by update techniques. The accumulation of curvature information takes apparently longer with the inverse dynamics formulation, even after the kinematic optimization in case (3). This indicates that nonlinearities from the robot's multibody dynamics (being evaluated four times per Runge-Kutta step in case (1) but only once per node else) have a great influence in the press connection maneuver. In accordance with the sensitivity analysis in [22], geometric restrictions play a minor role only.

grid size	accuracy τ	optimal time	iterations SQP	QP	CPU time [s] total	QP	method
20	10^{-2}	1.78	5	39	31.7	2.5	(1)
20	10^{-2}	1.78	11	111	17.8	15.1	(2)
20	10^{-2}	1.79	$3+8$	$33+64$	13.2	11.1	(3)
32	10^{-2}	1.78	5	42	51.0	4.2	(1)
32	10^{-2}	1.78	13	135	33.3	28.2	(2)
32	10^{-2}	1.79	$4+7$	$49+56$	20.0	17.2	(3)
40	10^{-2}	1.78	5	47	67.3	5.9	(1)
40	10^{-2}	—					(2)
40	10^{-2}	1.79	$4+10$	$50+92$	35.4	30.3	(3)
20	10^{-3}	1.767	10	81	62.1	5.1	(1)
20	10^{-3}	1.767	25	231	36.4	30.3	(2)
20	10^{-3}	1.769	$7+16$	$67+144$	28.3	24.2	(3)
32	10^{-3}	1.770	10	90	100.6	8.7	(1)
32	10^{-3}	1.764	31	301	73.6	61.5	(2)
32	10^{-3}	1.767	$9+25$	$101+227$	68.6	58.4	(3)
40	10^{-3}	1.767	13	121	165.9	14.5	(1)
40	10^{-3}	—					(2)
40	10^{-3}	1.765	$9+33$	$104+327$	115.1	97.8	(3)
20	10^{-4}	1.7656	17	140	106.2	8.6	(1)
20	10^{-4}	1.7629	52	481	75.2	62.5	(2)
20	10^{-4}	1.7648	$9+33$	$86+284$	52.4	44.2	(3)
32	10^{-4}	1.7639	21	196	214.7	19.2	(1)
32	10^{-4}	1.7609	62	619	153.7	128.1	(2)
32	10^{-4}	1.7595	$11+61$	$117+596$	160.0	135.7	(3)
40	10^{-4}	1.7634	24	227	307.7	27.7	(1)
40	10^{-4}	—					(2)
40	10^{-4}	1.7613	$12+60$	$128+586$	200.5	169.8	(3)

Table 1: Optimization results for the press connection using (1) forward dynamics, (2) inverse dynamics, (3) kinematics and inverse dynamics

5 Conclusions

For trajectory optimization problems with inverse dynamics we have introduced a new solution technique that can be interpreted as a special control parameterization in the direct BVP approach, or as kinematic motion planning with dynamic constraints. Its most appealing property is that it renders the numerical integration of differential equations unnecessary, thus enabling "trajectory optimization without simulation". Discretization errors are completely eliminated from the state variables to remain only in the path constraints.

First computational results demonstrate that inverse dynamics optimization can drastically reduce computation times in robot motion planning, especially when preceded by a purely kinematic trajectory optimization. On this background we may assume the point of view that we can control the robot position directly while satisfying various equality and inequality restrictions on joint angles, speeds, and accelerations. This is basically the kinematic point of view taken in commercial CAD-based motion design systems, but we can easily handle complex nonlinear restrictions (including multibody dynamics) that are left otherwise as disturbances to the feedback controllers, and we add the potential of efficient numerical algorithms to compute optimal trajectories instead of just feasible ones.

An additional benefit of the new technique is the availability of exact first and second derivatives at reasonable cost. Even if accuracy is not an important issue (as in our application example), both first and second derivatives might be useful to calculate reliable error estimates. In addition, exact second derivatives may resolve the problem of slow accumulation of curvature information. However, the QP can then become indefinite during the SQP iteration, which raises a number of mathematical questions concerning globalization strategies etc. Such issues should be addressed in future investigations.

6 Acknowledgements

This research was supported by the federal ministery of education, science, research and technology (BMBF) under grant 03-BO7HEI-6/3.0M750. The author would further like to thank KUKA GmbH, Augsburg, for supplying data of the robot and the press connection maneuver.

References

[1] J. T. Betts and W. P. Huffman. Path constrained trajectory optimization using sparse sequential quadratic programming. *AIAA J. Guidance*, 16(1):59–68, 1993.

[2] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robotics Research*, 4(3):3–17, 1985.

[3] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of constrained optimal control problems. In *Proc. 9th IFAC World Congress*, pages 242–247, Budapest, Hungary, 1984. Pergamon Press.

[4] H. G. Bock, J. P. Schlöder, M. C. Steinbach, H. Wörn, V. H. Schulz, and R. W. Longman. Schnelle Roboter am Fließband: Mathematische Bahnoptimierung in der Praxis. In K.-H. Hoffmann, W. Jäger, T. Lohmann, and H. Schunck, editors, *Mathematik – Schlüsseltechnologie für die Zukunft*, pages 539–550. Springer Verlag, 1997.

[5] N. N. Bolotnik and F. L. Chernousko. Optimization of manipulation robot control. *Sov. J. Comp. and Syst. Sci.*, 28(5):127–169, 1991.

[6] R. Featherstone. *Robot dynamics algorithms*. Kluwer Academic Publishers, Boston, Dordrecht, Lancaster, 1987.

[7] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance*, 10(4):338–342, 1987.

[8] B. Hartel, M. C. Steinbach, H. G. Bock, and R. W. Longman. The influence of friction on time optimal robot trajectories. Submitted for publication, Nov. 1996.

[9] M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated kinematic chains. *ASME J. Dyn. Syst., Meas., and Contr.*, 93:164–172, 1971.

[10] J. Konzelmann, H. G. Bock, and R. W. Longman. Time optimal trajectories of elbow robots by direct methods. In *Proc. 1989 AIAA Guid., Nav., and Contr. Conf.*, pages 883–894, Boston, MA.

[11] J. Konzelmann, H. G. Bock, and R. W. Longman. Time optimal trajectories of polar robot manipulators by direct methods. *Modeling and Simulation*, 20(5):1933–1939, 1989.

[12] R. W. Longman, J. Li, M. C. Steinbach, and H. G. Bock. Issues in the implementation of time-optimal robot path planning. In *Proc. AIAA Nonl. Dyn. Syst. Symp.*, Reno, NV, Jan. 6–9 1997. To appear.

[13] R. W. Longman, V. H. Schulz, and H. G. Bock. Path planning for satellite mounted robots. In C. L. Kirk and D. J. Inman, editors, *Dynamics and Control of Structures in Space III*, pages 17–32. Computational Mechanics Publications, Southampton, Boston, 1996.

[14] M. Mössner-Beigel, M. C. Steinbach, H. G. Bock, and R. W. Longman. Time optimal path planning in polar robots with joint flexibility. Submitted for publication, Nov. 1996.

[15] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *IEEE J. Robotics and Autom.*, 3(2):115–123, 1987.

[16] K.-J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, University of Bonn, 1981.

[17] V. H. Schulz. Optimal paths for satellite mounted robots. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 76(S3):291–294, 1996.

[18] Z. Shiller. Time-energy optimal control of articulated systems with geometric path constraints. *ASME J. Dyn. Syst., Meas., and Contr.*, 118(1), 1995.

[19] M. C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *Int. Series Numer. Math.*, pages 213–222. Birkhäuser Verlag, Basel, 1994.

[20] M. C. Steinbach. *Fast Recursive SQP Methods for Large-Scale Optimal Control Problems.* Ph. D. dissertation, University of Heidelberg, 1995.

[21] M. C. Steinbach. Structured interior point SQP methods in optimal control. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 76(S3):59–62, 1996.

[22] M. C. Steinbach, H. G. Bock, G. V. Kostin, and R. W. Longman. Mathematical optimization in robotics: Towards automated high speed motion planning. Submitted to Surveys on Mathematics for Industry, Jan. 1997.

[23] M. C. Steinbach, H. G. Bock, and R. W. Longman. Time optimal control of SCARA robots. In *Proc. 1990 AIAA Guid., Nav., and Contr. Conf.*, pages 707–716, Portland, OR.

[24] M. C. Steinbach, H. G. Bock, and R. W. Longman. Time-optimal extension and retraction of robots: Numerical analysis of the switching structure. *J. Optim. Theory and Appl.*, 84(3):589–616, 1995.

[25] O. von Stryk. *Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen.* Ph. D. dissertation, TU München, 1994.