Optimal Trajectory Planning of Manipulators  
Subject to Motion Constraints  

Yueshi Shen  
Department of Information Engineering  
Research School of Information Sciences and Engineering  
Australian National University  
Canberra ACT 0200, Australia  
Email: yueshi.shen@anu.edu.au  

Knut Hüper  
National ICT Australia Limited  
†  
Systems Engineering And Complex Systems Program  
Locked Bag 8001  
Canberra ACT 2601, Australia  
Email: knut.hueper@nicta.com.au  

Abstract—This paper presents a novel approach to plan an  
optimal joint trajectory for a manipulator robot performing a  
compliant motion task. In general, a two-step scheme will be  
deployed to find the optimal robot joint curve. Firstly, we approx-  
imate the functional and use Newton’s iteration to numerically  
calculate the joint trajectory’s intermediate discretized points,  
instead of solving a corresponding nonlinear, implicit Euler-  
Lagrange equation. Secondly, we interpolate these points to get  
the final joint curve in a way such that the motion constraints  
will always be sustained throughout the movement. An example  
of motion planning for a 4-degree-of-freedom robot WAM will  
be given at the end of this paper.

I. INTRODUCTION

Generally speaking, the task for a motion planner is to  
specify a motion to be executed by actuators. A properly  
planned motion can have advantages with respect to different  
aspects, e.g., obstacle avoidance, work efficiency optimization,  
better tracking performance etc. More specifically, for multi-  
link robotic systems, the reference trajectory generation can  
be divided into the following two subproblems.

\begin{itemize}
  \item **P1** For a given robot and task, plan a path for the  
    end-effector between two specified positions. Such a  
    path should satisfy either equality (e.g., robot’s  
    end-tip is required to move on a working surface) or  
    inequality (e.g., obstacle avoidance) constraints; in the  
    meantime, it may optimize a performance index.
  \item **P2** For a given end-effector path expressed in the oper-  
    ational space (which usually coincides with the Cartesian  
    space), find its corresponding joint trajectory through inverse  
    kinematics. Similarly, some performance index can be optimized  
    in case of a redundant robot, namely, the robot has more degrees  
    of freedom (DOFs) than necessary to perform the given task.
\end{itemize}

The conventional strategy for robot path planning often  
requires P1 and P2 to be resolved separately. The necessity  
of individually solving P1 comes from the facts that: 1) the  
geometrical shape of end-effector path is very important for  
the automatic task execution, 2) some optimization criterions  
or constraint conditions are naturally easier described in the  
operational space, e.g., the presence of obstacles [1]. For  
P1, it is a widely researched topic to design a collision-free  
configuration (including position and orientation) path for a  
single rigid object travelling in a crowded environment, and  
most of such algorithms can be found in Latombe’s work [2].

However, the drawback of the above approach is its compu-  
tational difficulties or inefficiencies artificially introduced by  
the isolated processions in the operational and joint spaces.  
Firstly, it may be cumbersome to generalize the end-effector’s  
path planning algorithm described in [2] on a multi-rigid-body  
manipulator due to the robot’s feasible configuration space  
constrained by its nonlinear kinematics and joints’ mechanical  
stops. Secondly, the robot’s kinematic model is usually ignored  
at the end-effector path planning stage, so that the resulting  
joint motion may contain some unpredictable behaviors, e.g.,  
when the robot is in the neighborhood of a singularity [3].

Recently, authors have adopted the methodology of solving  
P1 and P2 in one attempt by casting a robot path planning  
problem as an optimal control problem [4],[5], some even  
seek for the optimal time history of joint torques [6],[7].  
Most of these papers are aiming towards the minimum exe-  
cution time under the constraint of drive torque limit (some  
consider obstacle avoidance as well). Such an approach will  
automatically eliminate the necessity of calculating the fea-  
sible configuration space for the manipulator, since the path  
planning is directly performed in the joint space. However,  
path optimization incorporating robot dynamics will often end  
up with a suboptimal result, because apart from the multi-  
rigid-body model, some dynamics factors which are either  
difficult to model (e.g., motor torque ripple) or not a smooth  
function (e.g., Coulomb friction) are usually beyond the scope  
of optimization.

This paper studies the optimization of a robot path in the  
joint space (also solving P1 and P2 at the same time) with  
regard to some synthetical geometric performance index for  
the motions of both robot’s joint and end-effector as well.
The idea is to optimize not only the joint trajectory but also its resulting end-effector path in the eye of the joint space, since the motion of robot’s end-effector is ultimately driven by that of its joints through a surjective forward kinematic map, for which a fairly accurate model is usually possible to obtain. Our ‘lump-sum’ path planning approach will have the advantage on achieving such a synthetical optimality, as the anterior step of the conventional strategy is usually blind to the final joint trajectory’s optimality.

Furthermore, rather than the free space motion, we consider the situation that the robot’s end-effector is subject to some motion constraints. A typical example is that the manipulator is performing a compliant motion task, i.e., its end-effector is only allowed to move on some working surface. The extreme case in this category is that the end-effector path is totally determined before we carry out the path planning. Then our algorithm will downgrade to a solver for only P2 as in [8],[9], except that we allow the cost function to contain second order derivatives and we avoid solving the Euler-Lagrange differential equation.

The rest of this paper is organized as follows: Section 2 gives the mathematical formulation of our optimal robot path planning problem; Section 3 presents a two-step calculation scheme which numerically computes the optimal joint trajectory; Section 4 shows an example of applying our algorithm on planning the motion for a 4-DOF robot WAM with its end-effector constrained to move on a sphere.

II. PROBLEM DESCRIPTION

First of all, we are considering the situation that a robot is performing some compliant motion task, i.e., the manipulator is subject to l end-effector constraints.

\[ C_i(p(t), R(t)) = C_i(\kappa_p(q(t)), \kappa_R(q(t))) = 0 \]

[i = 1, \cdots, l]

where \( q : \mathbb{R} \to \mathbb{R}^p, t \mapsto q(t) \) is the p-dim joint curve, \( p : \mathbb{R} \to \mathbb{R}^n, t \mapsto p(t) \) is the path of end-effector’s linear position, and \( R : \mathbb{R} \to SO_3, t \mapsto R(t) \) is the path of end-effector’s orientation. Moreover, \( \kappa_p : \mathbb{R}^p \to \mathbb{R}^p, q(t) \mapsto p(q(t)) \) and \( \kappa_R : \mathbb{R}^p \to SO_3, q(t) \mapsto R(q(t)) \) are the manipulator’s forward position and orientation kinematic map, respectively.

For example, suppose the manipulator’s end-effector is required to move on a 2D plane in \( \mathbb{R}^3 \) (Fig.1), e.g., the manipulator is wiping a flat window. Then the corresponding motion constraint can be expressed as:

\[ C(q(t)) = \kappa_p(q(t)) \, ^\top n = p(t) \, ^\top n = 0 \]

where \( n \in \mathbb{R}^3 \) is the plane normal.

Another situation is that the path of the end-effector is directly determined by the given task itself, e.g., the robot performs cutting, welding and so on. Then the motion constraint functions may be given as:

\[
\begin{aligned}
C_1(q(t)) &= \kappa_p (q(t)) - p_d(t) = 0 \\
C_2(q(t)) &= \kappa_R (q(t)) - R_d(t) = 0
\end{aligned}
\]
Consider a regular partition of the time interval \([t_0, t_n]\):

\[
t_k = kh, \quad k \in \{0, 1, \ldots, n - 1\}
\]

where \(h = (t_n - t_0)/n\) is the step size.

Let

\[
q_k^* := q(t_k^*), \quad \dot{q}_k^* := \dot{q}(t_k^*), \quad \mu_{i,k} := \mu_i(t_k)
\]

\(k \in \{0, 1, \ldots, n - 1\}\)

(12)

where \(q_k^*\) can be approximated by \(q_{k-1}, q_k, q_{k+1}, \xi_0\) and \(\xi_n\).

\[
q_0 \simeq 2q_1 - 2q_0 - \frac{2}{h} \xi_0
\]

\[
q_k \simeq q_{k-1} - 2q_k + q_{k+1}
\]

\[
q_n \simeq 2q_{n-1} - 2q_n + \frac{2}{h} \xi_n
\]

(13)

In addition, we define \(t_k^*\) as:

\[
t_k^* := kh + \frac{h}{2}, \quad k \in \{0, 1, \ldots, n - 1\}
\]

(14)

Similarly, we let

\[
q_k^* := q(t_k^*), \quad \dot{q}_k^* := \dot{q}(t_k^*), \quad k \in \{0, 1, \ldots, n - 1\}
\]

(15)

where \(q_k^*\) and \(\dot{q}_k^*\) can be represented by \(q_k\) and \(q_{k+1}\)

\[
q_k^* \simeq \frac{q_k + q_{k+1}}{2}
\]

(16)

\[
\dot{q}_k^* \simeq \frac{q_{k+1} - q_k}{h}
\]

(17)

Apply the approximation schemes (Eqs.16,17) and the Midpoint Rule [15], the integral of \(L_1\) can be approximated as:

\[
\int_{t_0}^{t_n} L_1(q(t), \dot{q}(t)) \, dt \simeq \sum_{k=0}^{n-1} L_1(q_k + \frac{q_{k+1} + q_k}{2}, \frac{q_{k+1} - q_k}{h}) \cdot h
\]

(18)

Meanwhile, using the approximation schemes (Eq.13) and the Trapezoidal Rule [15], and assuming the initial and final poses \((q_0, q_n)\) satisfy the motion constraints, the integral of \((L_2 + \sum \mu_i C_i)\) can be approximated as:

\[
\int_{t_0}^{t_n} \left( L_2(q(t), \dot{q}(t)) + \sum_{i=1}^{l} \mu_i(t)C_i(q(t)) \right) \, dt \simeq L_2(q_0, \frac{2q_1 - 2q_0}{h^2} - \frac{2}{h} \xi_0) \cdot \frac{h}{2}
\]

\[
+ \sum_{k=1}^{n-1} \left( L_2(q_k, \frac{q_{k-1} - 2q_k + q_{k+1}}{h^2}) \cdot \frac{h}{2} \right)
\]

\[
+ L_2(q_n, \frac{2q_{n-1} - 2q_n + \frac{2}{h} \xi_n}{h^2}) \cdot \frac{h}{2}
\]

\[
+ \sum_{i=1}^{l} \sum_{k=1}^{n-1} \mu_{i,k}C_i(q_k) \cdot h
\]

(19)

From the integration schemes (Eqs.18,19), we can see the original optimization objective is finally approximated by a
function of a set of the intermediate discrete joint positions and Lagrange multipliers.

\[
\dot{J}(q, \mu) \approx \dot{J}^*(Q)
\]

where

\[
\dot{J}^* := \text{R.H.S. of Eq.18} + \text{R.H.S. of Eq.19}
\]

\[
Q := \left[ q_{1,1}, \ldots, q_{p,n-1}, \mu_{1,1}, \ldots, \mu_{l,n-1} \right]^T
\]

and \(q_{j,k}\) is the \(j\)th element of \(q_k\).

With Eq.20, we eventually convert an infinite-dimensional variational problem \(P4\) into a finite-dimensional optimization problem \(P5\) as described below, for which a number of numerical methods have already been developed.

**P5** Find a vector \(Q \in \mathbb{R}^{(p+1)(n-1)}\) (Def.22) which minimizes the function \(J^*\) (Def.21).

The result of the discretized system \((Q)\) such that \(\nabla \dot{J}^*(Q) = 0\) will converge to the result of the Euler-Lagrange boundary problem (Eq.10), as the time step size \(h\) goes to 0. Levin et al. [16] give the proof of the above proposition for "the simplest problem of calculus of variations". It can be shown that our approximation (Eqs.18,19) schemes are 'consistent' in the sense of numerical analysis [17], therefore the proof in [16] can be extended to the constrained variational problem containing second order derivatives (i.e., \(P3\)), please refer to [18] for the detailed proof. At the moment, we require \(q\) and \(\dot{q}\) to be discretized under two time partition schemes (Eqs.12,15) and such a 'discrimination' on \(q\) and \(\dot{q}\) results in decomposing the integrand of \(J\) into \(L_1\) and \(L_2\).

To solve \(P5\), one of the most common methods is to apply Newton’s iteration (A1).

**A1** (Newton’s iteration)

1) Pick a reasonable guess of \(Q\).
2) Update \(Q\) by the following law until it converges.

\[
Q_{i+1} = Q_i - \delta H_{ij}^{-1}(Q_i) \nabla \dot{J}^*(Q_i)
\]

where \(Q_i\) is the \(i\)th iterate of \(Q\), \(\nabla \dot{J}^*\) is the gradient of \(\dot{J}^*\) with respect to \(Q\), and \(H_{ij}\) is the square matrix (Hessian) of second partial derivatives of \(\dot{J}^*\) with respect to \(Q\).

Keep applying step 2 until the norm of \(\nabla \dot{J}^*\) gets small enough.

However, the Newton’s iteration usually only enjoys local convergence properties, so a good choice of \(Q\)’s initial value is necessary. To make such a ‘guess’ more efficient, we can gradually increase the number of time partitions \(n\), then the previously converged result can be used to form the initial value for the next round of Newton’s iteration.

**A2** (Newton’s iteration, modified)

1) Pick a small \(n\) which makes \(Q\) easy to converge, and pick an initial guess of \(Q\).
2) \((\text{Inner loop})\) Update \(Q\) by the following law until it converges.

\[
Q_{i+1} = Q_i - \delta H_{ij}^{-1}(Q_i) \nabla \dot{J}^*(Q_i)
\]

The step size \(\delta \in (0,1]\) has been introduced here to improve the numerical stability. \(\delta\) should satisfy the Armijo condition [19], and can be calculated by the backtracking line search [20].

3) \((\text{Outer loop})\) Double \(n\), the value of newly introduced elements is initially set as the mean of their calculated left and right neighbors. With the expanded vector \(Q\), go to step 2.

**Extension:** Besides equality constraints such as Eq.1, robots are usually subject to inequality constraints as well, e.g., joint position/velocity limit or obstacle avoidance, which can eventually be converted into a bunch of constraints on the discrete joint positions \(C_i'(Q) < 0, i = l+1 \cdots m\) by using approximation schemes (Eqs.13,16,17).

To deal with the inequalities, we propose to use the interior point (IP) method [20], which relaxes the constraints \(C_i'(Q) < 0\) by inserting a convex, smooth ‘barrier’ function \(\phi\) in the cost function \(J^*\). An example for \(\phi\) is given as below. In practical implementations, the initial value of \(Q\) should be picked to satisfy all of the inequality constraints.

\[
\phi(Q) = \begin{cases} 
- \sum_{i=1}^{m} \log \left( -C_i'(Q) \right) & \text{all } C_i'(Q) < 0 \\
0 & \text{otherwise}
\end{cases}
\]

Apart from applying the above IP method, an easier way to avoid violating the joint velocity constraint is to linearly scale the execution time. Similar time scaling methods can sometimes also be employed for drive torque limit (e.g., the dynamic scaling technique [1]).

**B. Interpolation**

After having computed the intermediate discretized points, the next step is to find a sufficiently smooth joint trajectory \(q(t)\) such that

\[
q(t_k) = q_k, \quad C_i'(q(t)) = 0, \quad \forall t \in [t_0, t_n]
\]

To ensure the resulting joint curve satisfies both two conditions (Eqs.26,27), we design a 3-step calculation scheme (A3) as follows:

**A3** (Interpolation)

1) Interpolate the previously computed discretized points \(\{q_0, q_1, \cdots, q_{n-1}, q_n\}\) by a cubic spline called \(q_{org}\).
2) Interpolate \(p_k = \kappa_p(q_k)\) and \(R_k = \kappa_R(q_k)\) by curves \(p\) and \(R\) with the motion constraints \(C_i\) satisfied, or directly use the pre-determined \(p_d, R_d\) in Eq.3 if available.
3) Adjust the joint trajectory \( q_{\text{org}} \) to fit the end-effector path \( p, R \) by repeatedly applying the following law. Set \( q_{\text{old}}(t) = q_{\text{org}}(t) \) initially.

\[
q_{\text{new}}(t) = q_{\text{old}}(t) + J'(q_{\text{old}}(t)) \left[ \left( \log \left( \kappa_R(q_{\text{old}}(t))^\top R(t) \right) \right) - \kappa_P(q_{\text{old}}(t)) \right] \]  

(28)

where

a) \( \dagger \) denotes the pseudo-inverse operation;

b) \( \log \) denotes the matrix logarithm, uniquely defined in an open neighborhood of the identity matrix;

c) \( J'(t) : \mathbb{R}^p \rightarrow \mathbb{R}^6, \dot{q}(t) \rightarrow \begin{bmatrix} \omega(t) \\ \dot{p}(t) \end{bmatrix} \) can be regarded as \( J(t) \) (the Jacobian) transformed into another coordinate system, here \( I_3 \) is the \( 3 \times 3 \) identity matrix;

d) \( \sim : \mathbb{R}^3 \rightarrow SO_3, x \mapsto \tilde{x} \) is as

\[
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}
\]

with obvious inverse \( \wedge^{-1} : SO_3 \rightarrow \mathbb{R}^3, \tilde{x} \mapsto (\tilde{x})^{\wedge^{-1}} = x \).

e) the upper element in the square bracket is the rotation vector in canonical coordinates extracted from two rotation matrices, whose closed form has been revealed by Rodrigues’ Formula [11].

Sometimes only the tracking of \( p \) is feasible (or necessary) due to the robot’s kinematic constraints (or the demand of motion task). A pull back/push forward technique with rolling and wrapping for smooth interpolating curves on a manifold was proposed recently [21] and it can be used to compute \( p \). One remarkable feature of this technique is that it is a coordinate-free approach.

Eq.28 is a Newton-Raphson-like root-finding algorithm. Using the pseudo-inverse of the Jacobian is to fine-tune the robot pose so that the end-effector configuration will get closer to the desired configuration \( \{ p(t), R(t) \} \) with minimum change in joint position, i.e., \( \Delta q(t) = q_{\text{new}}(t) - q_{\text{old}}(t) \). Here we assume we’ve already have enough discretized points, in other words, \( \{ p(t), R(t) \} \) and \( \{ \kappa_P(q_{\text{org}}(t)), \kappa_R(q_{\text{org}}(t)) \} \) will be sufficiently close to each other.

IV. EXAMPLE

WAM is a 4-joint robot manipulator with human-like kinematics (see Fig.2). Viewed in the Cartesian space, WAM’s 4-DOF corresponds to its end-effector moving to an arbitrary 3D position within its reachable space and rotating about vector \( p(t) \) (1-DOF in orientation).

Now suppose the robot is required to move its end-effector on some working surface, for instance, a sphere (see Fig.3).

Given the initial and final joint positions, now the challenge is to find a joint trajectory yielding a comprehensive optimality with respect to the following four aspects:

1) WAM’s last link is as perpendicular to the spherical surface as possible (i.e., \( \theta \rightarrow \min \));

2) The end-effector traverses minimum distance;

3) The joints traverse minimum distance;

4) The joints yield minimum curvature.

In addition, the joint position limit must be satisfied throughout the movement.

Summarily, the overall path planning problem for WAM can be described as follows.

P6 Find a sufficiently smooth 4-dim joint curve \( q \) subject to

\[
q_{\text{min}} \leq q(t) \leq q_{\text{max}}, \forall t \in [t_0, t_n]
\]

(30)

\[
\| r + p(t) \| = R, \forall t \in [t_0, t_n]
\]

(31)

\[
q(t_0) = q_0, \quad q(t_n) = q_n
\]

(32)

such that the following cost function \( J \) has a minimal value.

\[
J = \int_{t_0}^{t_n} \left( \cos(\pi - \theta(t)) + \alpha \dot{p}^\top \dot{p} + \beta \dot{q}^\top \dot{q} + \gamma \ddot{q}^\top \ddot{q} \right) \, dt
\]

(33)

\[
\cos(\pi - \theta(t)) = \frac{o(t)^\top (r + p(t))}{\|o(t)\| \| (r + p(t)) \|}
\]

(34)

where \( o(t) \in \mathbb{R}^3 \) is the \( z \)-axis of link 4’s local coordinate viewed in the inertial frame \( \{ o_r, x_r, y_r, z_r \} \).

\[
o(t) = R(t) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \kappa_R(q(t)) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}
\]

(35)

\( r \in \mathbb{R}^3 \) is the vector from the center of sphere to the base of WAM, and \( o(t), p(t), r \) are all as Fig.3 illustrates. \( \alpha, \beta \), and \( \gamma \) are the weights of their corresponding terms, their values can be chosen according to customer’s preference. \( R \) is the radius of the sphere, and \( \| x \| = \sqrt{x^\top x} \).
Figs. 4, 5 show the result of implementing our path planning method for P6, the second step of interpolation is only carried out for \( p \) by using the pull back/push forward technique with rolling and wrapping. Here \( \alpha = 0.5, \beta = 2.5 \times 10^{-4}, \gamma = 5 \times 10^{-7}, R = 0.3, r = [-0.6, 0, 0] \), all data about WAM’s kinematic model including \( q_{\text{min}}, q_{\text{max}} \) are as in [22]. WAM’s initial and final poses are arbitrarily chosen, but they both satisfy the motion constraints. Fig. 6 illustrates the resulting end-effector path, from which we can see the equality constraint (Eq. 31) has been fulfilled throughout the entire movement. The black and red straight lines in Fig. 6 are respectively the sphere normal and direction of WAM’s last link, and the lower curve are the only places that WAM’s last link can be perpendicular to the spherical surface, so the path of end-effector looks like to be ‘attracted’ towards that region.

A Java 3D visualization demo has been submitted with this paper. Also, we have completed a motion control experiment implementing the our path planning algorithm on the real robotic system WAM, the experiment’s setup is shown in Fig. 2.

V. Future Work

The current time partition schemes (Eqs. 11, 14) for \( \ddot{q} \) and \( \dot{q} \) result in the integrand of \( J \) under the form of \( L_1(q(t), \dot{q}(t)) + L_2(q(t), \ddot{q}(t)) \). The limitation of such a decomposition is its inability of representing any optimization objective simultaneously relevant to \( q \), \( \dot{q} \) and \( \ddot{q} \). Therefore, it will be worthwhile to find ‘consistent’ [17] approximation schemes for \( \ddot{q}(t_k) \) and \( \dot{q}(t_k) \) discretized under a unique time partition scheme. Then the scope of our path optimization can be extended to some dynamic properties such as joint drive torque or end-effector acceleration.

Moreover, at the moment we are working on the smooth interpolation incorporating the end-effector’s position and orientation as well. Such an interpolation approach will be useful for some industrial processes with requirements on both two properties, e.g., spray coating.

Finally, some proper hybrid motion/force control law need to be designed so as to make a robot successfully execute a given compliant motion task. The majority of researchers in this area have formulated their hybrid control schemes (including the desired path, error extraction, and motion/force filter matrices) in the operational space and further employ robot’s dynamic model to transform the control signal (force or velocity) into the joint space [23],[24],[25],[26]. However, as we discovered, the operational-space controller is not so capable to overcome disturbances originated at the joint level and the tracking will become poor, especially when the robot’s inertia matrix is ill-conditioned [27]. Instead, we propose to reformulate the hybrid motion/force control law in the joint space, which allows us to apply different control gains on each of the joints and lets the error compensation be directly carried out at the joint level. Also, the path planning algorithm presented in this paper can be used to calculate the proposed controller’s reference trajectory.

VI. Conclusion

In this paper, we present a two step path planning approach for numerically calculating a joint trajectory for a manipulator subject to motion constraints yielding some synthetical geometrical optimality. The joint curve calculated from our algorithm will converge to the resulting curve of the Euler-Lagrange equation as we let the time step in the discretization scheme go to zero. An example of motion planning for a 4-DOF robot WAM performing a compliant motion task has been given at the end of this paper, it shows our method can generate quite satisfactory results for an actual robotic system with a fairly complicated optimization objective.

REFERENCES

Fig. 4. Discretized points and interpolating joint trajectory, a) joint 1, b) joint 2

Fig. 5. Discretized points and interpolating joint trajectory, a) joint 3, b) joint 4

Fig. 6. End-effector path calculated from joint trajectory in Figs.4,5. Black: sphere normal, red: WAM’s last link


