Derivation of kinematic relations for a robot using Maple

Johanna Wallén, Svante Gunnarsson, Mikael Norrlöf

Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
WWW: `http://www.control.isy.liu.se`
E-mail: `johanna@isy.liu.se`, `svante@isy.liu.se`,
`mino@isy.liu.se`

1st June 2006

Report no.: LiTH-ISY-R-2732
Submitted to Reglermöte 2006

Abstract

A first step towards making a toolbox in Maple for industrial robot modelling is taken. Position and orientation of the tool can be determined in terms of the Denavit-Hartenberg joint variables and also the Jacobian relating the linear and angular velocities to the joint velocities. Further on it will be possible to, *e.g.*, differentiate the Jacobian. Future work includes to evaluate different kinds of sensors and sensor locations and symbolically generate the kinematic models using Maple. It also means to incorporate the models in Matlab- or C-code for including the results, *e.g.*, in an Extended Kalman Filter algorithm for state estimation.

Keywords: Modelling, Industrial robots, Symbolic computations, Sensor fusion

DERIVATION OF KINEMATIC RELATIONS FOR A ROBOT USING MAPLE [1]

Johanna Wallén [*] **Svante Gunnarsson** [*]
Mikael Norrlöf [*]

[*] *Department of Electrical Engineering*
Linköpings universitet
SE-581 83 Linköping, Sweden
`{johanna, svante, mino}@isy.liu.se`

Abstract: A first step towards making a toolbox in Maple for industrial robot modelling is taken. Position and orientation of the tool can be determined in terms of the Denavit-Hartenberg joint variables and also the Jacobian relating the linear and angular velocities to the joint velocities. Further on it will be possible to, *e.g.*, differentiate the Jacobian. Future work includes to evaluate different kinds of sensors and sensor locations and symbolically generate the kinematic models using Maple. It also means to incorporate the models in Matlab- or C-code for including the results, *e.g.*, in an Extended Kalman Filter algorithm for state estimation.

Keywords: Modelling, Industrial robots, Symbolic computations, Sensor fusion

1. INTRODUCTION

Industrial robots are nowadays used in many different areas and in various applications, which increases the demands on performance and productivity. When manufacturing the robots, low-cost alternatives of motors, gears and other components are used to a greater extent. This implies higher degree of nonlinearity and mechanical flexibility, which makes it more difficult to achieve the demands on performance. Integration of new low-cost sensors such as accelerometers, gyros, cameras, *etc.*, is one way to maintain and perhaps improve the performance. Additional information from sensors demands signal processing and sensor fusion, which is hard to achieve without good models and modelling tools.

The purpose of this paper is to present a first step towards a modelling platform for efficient derivation of the necessary equations for doing, *e.g.*, sensor fusion. The idea is to easy evaluate different kinds of sensors and also various sensor locations using Maple (Maplesoft, 2006) as a tool to symbolically generate the kinematic models. Thereafter the models will be incorporated in Matlab- or C-code to implement, *e.g.*, state estimation using an Extended Kalman Filter (EKF) algorithm.

An example of previous work in this area is Bienkowski and Kozlowski (1998), where a Mathematica based approach to robot modelling is presented. The kinematic derivation in Bienkowski and Kozlowski (1998) does, however, not use the Denavit-Hartenberg (DH) representation (see Sec. 3.2) which is a standard in the robotics field. Another contribution is Corke (1996) but the focus there is on the dynamics and how dynamic models of robots can be simplified using Maple. A Mathematica package, *Robotica*, has also been developed by Nethery and Spong (1994). This package is based upon Spong and Vidyasagar (1989) and gives support for both kinematic as well as dynamic modelling. Although similar to what the

[1] Supported by the Swedish Research Council (VR).

authors of this paper is aiming at, the package Robotica is no longer supported or updated and it is not sure that it works in the last version of Mathematica without editing the source code (according to the homepage of one of the authors of Robotica, Mark W. Spong).

This paper is organised as follows: in Sec. 3 basic concepts and equations of the robot kinematics are described, based upon mainly Spong and Vidyasagar (1989) and Norrlöf (1999). The Maple implementation, which is described in much more detail in Wallén (2006), is then presented in Sec. 4. Sec. 5 handles some applications of the symbolic models derived, and in Sec. 6 conclusions are drawn.

2. MOTIVATING EXAMPLES

In this section an example with large kinematic equations and then an example of sensor integration motivates the need of a good symbolic based modelling tool.

2.1 Kinematic equations

The need of a good modellig tool, and in particular a good symbolic based modelling tool such as Maple, is illustrated by the following example where the industrial robot IRB1400 (Norrlöf, 1999) from ABB Robotics is used. A more detailed derivation of the equations in this section is given in Sec. 3. The part of the kinematic relations giving the linear velocity of the robot tool as a function of joint position and joint angular velocity, $J(\theta)\dot{\theta}$, is referred to as the Jacobian of the robot. The Jacobian describing the linear velocity of the robot tool for the three first joints of IRB1400 is

$$J_{\mathbf{v}} = \begin{pmatrix} -0.15\sin\theta_1 & 0.6\cos\theta_1\cos\theta_2 & J_{\mathbf{v}13} \\ 0.15\cos\theta_1 & 0.6\sin\theta_1\cos\theta_2 & J_{\mathbf{v}23} \\ 0 & J_{\mathbf{v}32} & J_{\mathbf{v}33} \end{pmatrix} \quad (1)$$

where

$$\begin{aligned} J_{\mathbf{v}13} &= \cos\theta_1\big(0.12\cos\theta_2\cos(-\theta_2+\theta_3) \\ &\quad - 0.12\sin\theta_2\sin(-\theta_2+\theta_3)\big) \\ J_{\mathbf{v}23} &= \sin\theta_1\big(0.12\cos\theta_2\cos(-\theta_2+\theta_3) \\ &\quad - 0.12\sin\theta_2\sin(-\theta_2+\theta_3)\big) \\ J_{\mathbf{v}32} &= -0.6\sin^2\theta_1\sin\theta_2 - 0.6\cos^2\theta_1\sin\theta_2 \\ J_{\mathbf{v}33} &= -\sin\theta_1\big(0.12\sin\theta_1\sin\theta_2\cos(-\theta_2+\theta_3) \\ &\qquad + 0.12\sin\theta_1\cos\theta_2\sin(-\theta_2+\theta_3)\big) \\ &\quad - \cos\theta_1\big(0.12\cos\theta_1\sin\theta_1\sin\theta_2\cos(-\theta_2+\theta_3) \\ &\qquad + 0.12\cos\theta_1\cos\theta_2\sin(-\theta_2+\theta_3)\big). \end{aligned}$$

As can be seen, the expressions are large even though just the three first links are examined, and when considering all six links the Jacobian becomes much more complex (approximately three pages of Maple code). It is desirable to avoid deriving these expressions by hand, and instead use a computer algebra tool, *e.g.*, Maple.

2.2 Sensor integration

A second motivating example is an application of the results in this paper, namely sensor integration. This application is also further described in Sec. 5. Using an accelerometer at the robot tool gives the measurement equation

$$y_t = h(x_t) + e_t = \begin{pmatrix} \theta \\ \ddot{\chi} \end{pmatrix} + e_t,$$

see (25). The vector y_t contains all measured signals, θ is a vector containing the measured joint angles and $\ddot{\chi}$ describes the acceleration vector in the coordinate frame of the accelerometer. Comparison to equation (22) shows a need of computing the time derivatives of the Jacobian and considering the complexity of the Jacobian in the previous example it is clear that this derivation is impossible to do by hand. This expression is also used when computing the EKF, see Sec. 5.2.

3. ROBOT KINEMATICS

There are two ways to describe the position kinematics of the robot — *forward kinematics* and *inverse kinematics*. Here, forward kinematics, *i.e.*, the position and orientation of the robot tool are determined in terms of the *joint variables*, and also the DH representation is described.

Generally a robot has $n+1$ links and the base of the robot is defined as link 0. It also has n joints, where joint i is situated where links $i-1$ and i are connected. The ith joint variable, q_i, represents the relative displacement between adjacent links. For a *revolute* joint q_i is the angle of rotation and the joint displacement in the case of a *prismatic* joint. Fig. 1 shows the joint variables $q_i = \phi_i$ for the robot IRB1400, with only revolute joints. Each link also has a coordinate frame i attached rigidly to link i. (Spong and Vidyasagar, 1989)

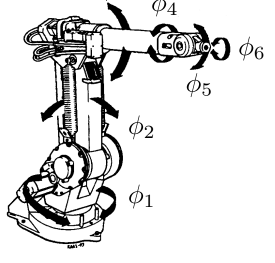

Fig. 1. The IRB1400 manipulator with joint variables ϕ_i.

3.1 Position kinematics

A parallel translation of the vector $\mathbf{p}_1$ by the vector $\mathbf{d}_0^1$ in coordinate frame 0 is described by

$$\mathbf{p}_0 = \mathbf{p}_1 + \mathbf{d}_0^1.$$

The basic rotation matrix R_1^0 describes the transformation of $\mathbf{p}$ between coordinate frame 0 and frame 1 as $\mathbf{p}_1 = R_1^0\mathbf{p}_0$. The inverse transformation is according to Spong and Vidyasagar (1989)

$$R_1^0 = (R_0^1)^{-1} = (R_0^1)^T.$$

Rotation with angle θ around the z-axis can be represented by the basic rotation matrix

$$R_0^1 = R_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The transformation matrices describe the order of the rotations. Study for example

$$\mathbf{p}_0 = R_0^1 R_1^2 \mathbf{p}_2 = R_0^2 \mathbf{p}_2. \tag{2}$$

First a rotation of the frame 1 is performed relative to frame 0 described by R_0^1 followed by rotating the frame 2 relative to frame 1 according to R_1^2. The order of the transformation matrices cannot be changed. (Spong and Vidyasagar, 1989)

If the rotation is around the fixed frame 0 all the time the rotation matrix becomes

$$R_0^2 = R_1^2 R_0^1. \tag{3}$$

(2) is similar to (3), because both matrices represent the same transformation between frame 0 and frame 2.

The most general transformation between the coordinate frame n and frame 0 can be described by a rotation combined with a translation. This is called a *rigid motion* if R_0^n below is orthogonal

$$\mathbf{p}_0 = R_0^n \mathbf{p}_n + \mathbf{d}_0^n \tag{4}$$

The rigid motion can be represented by a matrix of the form (Spong and Vidyasagar, 1989)

$$H = \begin{pmatrix} R & \mathbf{d} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{5}$$

These transformations are called *homogeneous transformations.* The *homogeneous representation* $\mathbf{P}_i$ of the vector $\mathbf{p}_i$ is defined as

$$\mathbf{P}_i = \begin{pmatrix} \mathbf{p}_i \\ 1 \end{pmatrix}.$$

Now the transformation (4) can be written as the homogeneous matrix equation

$$\mathbf{P}_0 = H_0^1 \mathbf{P}_1.$$

Combining two homogeneous transformations, *e.g.*, $\mathbf{p}_0 = R_0^1\mathbf{p}_1 + \mathbf{d}_0^1$ and $\mathbf{p}_1 = R_1^2\mathbf{p}_2 + \mathbf{d}_1^2$, gives

$$H_0^1 H_1^2 = \begin{pmatrix} R_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} R_1^2 & \mathbf{d}_1^2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} R_0^1 R_1^2 & R_0^1\mathbf{d}_1^2 + \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{pmatrix}.$$

3.2 Denavit-Hartenberg representation

The DH representation describes a systematic way to parameterise the forward kinematics for rigid robots.

The homogeneous matrix A_i describes the transformation of the coordinates from frame i to frame $i-1$ under the assumption that the joints are either revolute or prismatic. The homogeneous transformation A_i is of the form

$$A_i = \begin{pmatrix} R_{i-1}^i & \mathbf{d}_{i-1}^i \\ 0 & 1 \end{pmatrix}.$$

The total homogeneous transformation T_i^j is then

$$T_i^j = A_{i+1}A_{i+2}\cdots A_{j-1}A_j = \begin{pmatrix} R_i^j & \mathbf{d}_i^j \\ 0 & 1 \end{pmatrix}. \tag{6}$$

Each homogeneous transformation A_i is regarded as a product of four basic transformations

$$\begin{aligned} A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\ &= \begin{pmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{7}$$

in the DH convention. See for example Spong and Vidyasagar (1989) for how the DH link parameters *angle* θ_i, *length* a_i, *offset* d_i and *twist* α_i are defined.

The homogeneous transformation $A_i = A(q_i)$ is a function of just one variable q_i, so three of the four parameters above are constant for a link and the fourth is the present joint variable. The joint variable for a revolute joint is θ_i whereas it is d_i for a prismatic joint. (Spong and Vidyasagar, 1989)

It is important to notice that choices of the coordinate frames are not unique. (Spong and Vidyasagar, 1989)

Example IRB1400 For the robot IRB1400 from ABB Robotics, depicted in Fig. 1, all joints are revolute. In the DH representation, the robot is put in a certain position when the joint variables are zero. This position is however different from the position of IRB1400 when ϕ_i are zero. In order to get the robot variables equal to the DH joint variables, the ϕ_i-variables need to be translated from its origin with the vector θ_0 and the transformation matrix θ_{trans} (Norrlöf, 1999)

$$\theta = \theta_{trans}\phi + \theta_0. \tag{8}$$

The DH joint variables are $q_i = \theta_i$, because there are only rotations and no translations of the joints. θ_{trans} is introduced in order to cope with the fact that the IRB1400 has a parallelogram linkage structure to link 3 (Spong and Vidyasagar, 1989). In practice this structure means that changing ϕ_2

will change the respective DH-parameter q_2 but it will also affect q_3 so that link 3 keeps the same orientation with respect to the base frame.

The DH link parameters α_i, a_i, d_i and joint variables θ_i for IRB1400 are given in Table 1 (Norrlöf, 1999).

Table 1. The DH link parameters and joint variables for ABB Robotics' industrial robot IRB1400 (Norrlöf, 1999).

Joint/link i	α_i [rad]	a_i [m]	θ_i [rad]	d_i [m]
1	$-\pi/2$	0.15	θ_1	0.475
2	0	0.6	θ_2	0
3	$-\pi$	0.12	θ_3	0
4	$\pi/2$	0	θ_4	0.72
5	$-\pi/2$	0	θ_5	0
6	0	0	θ_6	0.085

Below the transformation matrix for the robot tool relative to the base frame 0, when the variables ϕ_i are zero, is given as

$$T_0^6 = A_1A_2A_3A_4A_5A_6 \approx \begin{pmatrix} 0 & 0 & 1 & 0.955 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1.195 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

according to (6) and (7). The last column represents the position of the tool relative to the base frame, *i.e.*, $\mathbf{d}_0^6$ as described in (6), which means that the position of the tool relative to the base frame of the robot is 0.995 m in the x_0-direction, 0 m in the y_0-direction and 1.195 m in the z_0-direction.

3.3 Velocity kinematics

The forward kinematic equations determine the position and orientation of the robot tool given the joint variables. The Jacobian J_0^n (from the nth coordinate frame expressed in the frame 0) of this function determines the linear and angular velocities of a point on the robot to the joint velocities $\dot{\mathbf{q}}$ as

$$\begin{pmatrix} \mathbf{v}_0^n \\ \omega_0^n \end{pmatrix} = \begin{pmatrix} J_\mathbf{v} \\ J_\omega \end{pmatrix} \dot{\mathbf{q}} = J_0^n \dot{\mathbf{q}}, \tag{10}$$

where the Jacobian is a function of $\mathbf{q}$.

Linear velocity The linear velocity of the robot tool is

$$\mathbf{v}_0^n = \dot{\mathbf{d}}_0^n = \sum_{i=1}^{n} \frac{\partial \mathbf{d}_0^n}{\partial q_i} \dot{q}_i. \tag{11}$$

The prismatic joint i with the joint variable $q_i = d_i$ gives the relation

$$\mathbf{d}_{i-1}^i = d_i \mathbf{k} + R_{i-1}^i a_i \mathbf{i},$$

where $\mathbf{i} = (1\ 0\ 0)^T$ is the unit vector in coordinate frame i. If all other joints but the ith are fixed, differentiation gives

$$\dot{\mathbf{d}}_0^n = R_0^{i-1} \dot{\mathbf{d}}_{i-1}^i = \dot{d}_i R_0^{i-1} \mathbf{k} = \dot{d}_i \mathbf{z}_{i-1}.$$

Compared to (11) this gives

$$\frac{\partial \mathbf{d}_0^n}{\partial q_i} = \mathbf{z}_{i-1} = R_0^{i-1} \mathbf{k} = R_0^{i-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \tag{12}$$

where $\mathbf{z}_{i-1}$ is the resulting vector in the base frame when the unit vector $\mathbf{k}$ in coordinate frame $i-1$ is expressed in the orientation of the base frame with the transformation matrix R_0^{i-1}.

The revolute joint i is described by the relation

$$\mathbf{d}_0^n = \mathbf{d}_0^{i-1} + R_0^{i-1} \mathbf{d}_{i-1}^n.$$

If only the ith joint is actuated, all joint angles except θ_i are fixed and $\mathbf{d}_0^{i-1}$ and R_0^{i-1} are constant (Norrlöf, 1999). Differentiation gives

$$\dot{\mathbf{d}}_0^n = R_0^{i-1} \dot{\mathbf{d}}_{i-1}^n, \tag{13}$$

which can be written

$$\dot{\mathbf{d}}_{i-1}^n = \dot{q}_i \mathbf{k} \times \mathbf{d}_{i-1}^n \tag{14}$$

for revolute joints (Spong and Vidyasagar, 1989). Combining (13) and (14) gives

$$\frac{\partial \mathbf{d}_0^n}{\partial q_i} = \mathbf{z}_{i-1} \times (\mathbf{d}_0^n - \mathbf{d}_0^{i-1}). \tag{15}$$

(12) and (15) together with (10) and (11) gives the part of the Jacobian for the linear velocity, $J_\mathbf{v}$, as

$$J_\mathbf{v} = \begin{pmatrix} J_{\mathbf{v}1} & \ldots & J_{\mathbf{v}n} \end{pmatrix} \tag{16}$$

where

$$J_{\mathbf{v}i} = \frac{\partial \mathbf{d}_0^n}{\partial q_i} = \begin{cases} \mathbf{z}_{i-1}, & \text{prismatic joint,} \\ \mathbf{z}_{i-1} \times (\mathbf{d}_0^n - \mathbf{d}_0^{i-1}), & \text{revolute joint.} \end{cases} \tag{17}$$

Angular velocity Angular velocities can be added vectorially in a common coordinate frame (Spong and Vidyasagar, 1989). It is then possible to express the angular velocity of each link in the base frame and sum them to the total angular velocity of the robot tool.

If the ith joint is revolute, the joint variable is $q_i = \theta_i$ and the angular velocity expressed in coordinate frame $i-1$ can be written

$$\omega_{i-1}^i = \dot{q}_i \mathbf{k}, \tag{18}$$

where $\mathbf{k}$ is the unit vector $(0\ 0\ 1)^T$ in frame $i-1$.

For a prismatic joint instead, there is a pure translation and

$$\omega_{i-1}^i = 0. \tag{19}$$

Summing the angular velocities from each link gives the following angular velocity (Spong and Vidyasagar, 1989) of the tool

$$\begin{aligned}\omega_0^n &= \rho_1\dot{q}_1\mathbf{k} + \rho_2\dot{q}_2R_0^1\mathbf{k} + \ldots + \rho_n\dot{q}_nR_0^{n-1}\mathbf{k}\\ &= \sum_{i-1}^{n}\rho_i\dot{q}_i\mathbf{z}_{i-1}.\end{aligned} \tag{20}$$

(18)–(20) together with (10) give the part of the Jacobian for the angular velocity, J_ω, as

$$\begin{aligned}J_\omega &= \begin{pmatrix}\rho_1\mathbf{z}_0 & \ldots & \rho_n\mathbf{z}_{n-1}\end{pmatrix},\\ \rho_i &= \begin{cases}0, & \text{prismatic joint,}\\ 1, & \text{revolute joint.}\end{cases}\end{aligned} \tag{21}$$

Acceleration equations Deriving the Jacobian (10) gives the linear and angular acceleration (Spong and Vidyasagar, 1989) according to

$$\begin{aligned}\begin{pmatrix}\ddot{x}_0^n\\ \dot{\omega}_0^n\end{pmatrix} &= \frac{d}{dt}\big(J_0^n(\mathbf{q})\dot{\mathbf{q}}\big) = J_0^n(\mathbf{q})\ddot{\mathbf{q}} + \frac{d}{dt}\big(J_0^n(\mathbf{q})\big)\dot{\mathbf{q}}\\ &= J_0^n(\mathbf{q})\ddot{\mathbf{q}} + \left(\sum_{i=1}^{n}\frac{\partial}{\partial q_i}\big(J_0^n(\mathbf{q})\big)\dot{q}_i\right)\dot{\mathbf{q}}.\end{aligned} \tag{22}$$

4. MAPLE IMPLEMENTATION

The Maple implementation and the functions described in this section is based upon Wallén (2006), where the implementation is explained in more detail.

The DH joint variables θ_i, possibly translated by the vector θ_0 and transformed by the matrix θ_{trans}, see (8), can be calculated by means of the function `DHparameter` with the 6×1-matrices defining ϕ (`DHphi`) and θ_0 (`DHtheta0`) and the 6×6-matrix θ_{trans} (`Ttheta`) according to

```
DHthetatotal :=
  DHparameter(DHphi, DHtheta0, Ttheta);
```

With θ (`DHthetatotal`), d (`DHd`), a (`DHa`) and α (`DHalpha`) as input to `DHtransformation` the resulting DH transformation is computed as

```
DHtransformation(DHthetatotal, DHd, DHa,
  DHalpha):
```

For example the output `T2` gives the homogeneous transformation T_0^2 according to (6). When calculating (9) the robot stands still, so just change to $\phi = 0$, *i.e.*, $\theta = \theta_0$ in (8) and

```
DHtransformation(DHtheta0, DHd, DHa,
  DHalpha);
```

R_i^j and $\mathbf{d}_i^j$ in (6) and $\mathbf{p}_i$ as described in (4) can be computed with `DHposition`. For example the vector $\mathbf{p}_6$ and the matrix T_0^6 (from the calculations when the robot stands still) as inputs gives

```
p[6] := Vector([1,1,1]);
DHposition(p[6], T6):
```

The outputs are then $\mathbf{d}_0^6$ (`d_i`), R_0^6 (`Rij`) and $\mathbf{p}_0$ (`p_i`) according to (6). From the rotation matrices R_i^j it is also possible to be able to calculate the corresponding Euler angles and quaternions, see Spong and Vidyasagar (1989) and Sciavicco and Siciliano (2000).

The linear part of the Jacobian, see (16)–(17), can be calculated by `DHlinearjacobian`. The input `jointtype` describes if the different joints are prismatic (0) or revolute (1). The function calculates the linear part of the Jacobian for the number of joints prescribed in the input parameter `joints`. The transformation matrices `T1` to `T6` are also needed as inputs. Calculating the expression in (1) gives the following in Maple

```
jointtype := <<1>, <1>, <1>, <1>, <1>,
  <1>>:
joints := 3:
DHlinearjacobian(T1, T2, T3, T4, T5, T6,
  jointtype, joints):
Jv;
```

With `DHangularjacobian` the angular part of the Jacobian is computed, see (21). This function has the same inputs as `DHlinearjacobian` and the output is called `Jomega` as can be seen below.

```
DHangularjacobian(T1, T2, T3, T4, T5, T6,
  jointtype, joints):
Jomega;
```

The resulting Jacobian (10) including both the linear and the angular part, is computed by `DHjacobian` with the same input arguments as above and `J` as output, *i.e.*,

```
DHjacobian(T1, T2, T3, T4, T5, T6,
  jointtype, joints):
J;
```

5. APPLICATION

In this section, applications of the symbolic models derived in the previous sections will be presented. The first application is sensor integration, briefly covered in Sec. 2.2. A second application is state estimation using the EKF (Anderson and Moore, 1979).

5.1 Sensor integration

Sensor integration in robotics implies handling many different kinds of sensors, and here accelerometers and gyros will be discussed. First consider a rigid body model of a robot,

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = u, \tag{23}$$

with M the inertia matrix, C the Coriolis and centripetal term, and g gravity. u is the input torque. (23) can rewritten in state-space form

$$\dot{x} = f(x, u) \tag{24}$$

where $x = (\mathbf{q},\ \dot{\mathbf{q}})^T$. Sensor integration can now be interpreted as modelling the measurements produced by the sensor as a function of the states,

$$y = h(x).$$

A gyro measures the angular velocity $\omega_n^n = R_n^0 \omega_0^n$ in the gyro coordinate system, which is described by the Jacobian (10). Using an accelerometer instead gives the linear acceleration $\ddot{\mathbf{x}}_n^n = R_n^0 \ddot{\mathbf{x}}_0^n$ in the accelerometer frame, so the Jacobian has to be differentiated according to (22). Notice that in (22) $\ddot{\mathbf{q}}$ appears explicitly, but it is not part of the state vector. To compute $\ddot{\mathbf{q}}$ the model in (23) must be utilized and it will make the function h very complex. Again this is a motivation to use a symbolic modelling tool.

5.2 State estimation

As a starting point for the state estimation, the discrete state-space model below is used

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t) \\ y_t &= h(x_t) + e_t, \end{aligned} \tag{25}$$

where the vector y_t contains all measured signals. To use (24) it is necessary to discretise the continuous system, but this operation is not discussed further in this paper. The probability density functions for the process noise and measurement noise are assumed to be known.

A standard method for state estimation in signal processing and control is the EKF (Anderson and Moore, 1979). Below the time and measurement update are presented for the EKF

$$\begin{cases} \hat{x}_{t+1|t} = f(\hat{x}_{t|t}, u_t, 0), \\ P_{t+1|t} = F_t P_{t|t} F_t^T + G_t Q_t G_t^T, \end{cases}$$

$$\begin{cases} \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\big(y_t - h(\hat{x}_{t|t-1})\big), \\ P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1}, \\ K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}, \end{cases}$$

where the linearized matrices are given as

$$\begin{aligned} F_t &= \frac{\partial}{\partial x} f(x_t, u_t, 0)|_{x_t = \hat{x}_{t|t}}, \\ G_t &= \frac{\partial}{\partial w} f(x_t, u_t, w_t)|_{x_t = \hat{x}_{t|t}}, \\ H_t &= \frac{\partial}{\partial x} h(x_t)|_{x_t = \hat{x}_{t|t-1}}. \end{aligned}$$

As can be seen, the functions f and h need to be differentiated with respect to the states (and the disturbance w_t). If a gyro is used as a sensor the necessary linearisation shown above in the EKF implies to differentiate the Jacobian. With an accelerometer an even more involved equation needs to be differentiated due to the linearisation. Again it is motivated to use a tool such as Maple to be able to manipulate the equations in an efficient way.

6. CONCLUSION

A first step towards making a toolbox in Maple for industrial robot kinematic modelling is taken in this paper. The work is motivated by applications, such as state estimation, where the kinematic model is used to relate the states of the dynamic model with the measurements. To avoid numeric derivatives it is important to be able to manipulate the model analytically, for example in Maple as is done in this paper.

Future work includes to further develop the Maple functions and make a package with a user friendly interface. It also means to develop an export function where the result can be exported in a format that can be read from Matlab and/or C for including the results, *e.g.*, in an EKF algorithm for state estimation.

REFERENCES

Anderson, B. D. O. and J. B. Moore (1979). *Optimal Filtering.* Prentice-Hall. Englewood Cliffs, New Jersey.

Bienkowski, A. and K. Kozlowski (1998). A toolbox for teaching and researching robotics. In: *Proc. 1998 Worldwide Mathematica Conf.* Chicago, USA.

Corke, P. I. (1996). An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. In: *Proc. 1996 IEEE Internat. Conf. Robotics and Automation.* Vol. 2. pp. 1018–1023.

Maplesoft (2006). www.maplesoft.com.

Nethery, J. F. and M. W. Spong (1994). Robotica: A Mathematica package for robot analysis. *IEEE Robot. Automat. Mag.* **1**(1), 13–20.

Norrlöf, M. (1999). Modeling of industrial robots. Technical Report LiTH-ISY-R-2208. Dept. Electr. Eng., Linköpings universitet, Sweden.

Sciavicco, L. and B. Siciliano (2000). *Modelling and Control of Robot Manipulators.* 2nd ed. Springer-Verlag. 4th printing 2003.

Spong, M. W. and M. Vidyasagar (1989). *Robot Dynamics and Control.* John Wiley & Sons. New York.

Wallén, J. (2006). On robot modelling using Maple. Technical Report LiTH-ISY-R-2723. Dept. Electr. Eng., Linköpings universitet, Sweden.