7. Velocity Analysis and Manipulator Jacobians

7.1 Joint twist

Consider the motion of link $i$ relative to link $i-1$. It is given by the homogeneous transformation matrix:

$$ i^{-1}A_i = \begin{bmatrix} i^{-1}R_i & p_i \\ 0 & 1 \end{bmatrix} $$

(1)

where the joint angle $\theta_i$ in $R_i(\theta_i)$ is variable if the $i$th joint is revolute, or the joint extension $d_i$ in $p_i(d_i)$ is variable if the $i$th joint is prismatic.

We define the $i$th joint twist as the twist of link $i$ due to the motion of joint $i$, assuming all the other joints (1, 2, ..., $i-1$, $i+1$, $i+2$, ..., $n$) are immobile. Since we have reference frames attached to link $i$ (the moving rigid body) and to link $i-1$ (which can be considered fixed since joints 1, 2, ..., $i-1$ are immobile), the joint twist can be obtained very easily by differentiating the homogeneous transformation matrix $i^{-1}A_i(t)$. The twist in frame $i-1$ is simply:

$$ \frac{d}{dt} \left( i^{-1}A_i \right) \left( i^{-1}A_i \right)^{-1} = \begin{bmatrix} i^{-1}\dot{R}_i & i^{-1}\dot{p}_i \\ 0 & 0 \end{bmatrix} \begin{bmatrix} (i^{-1}R_i)^T \\ 0 \end{bmatrix} - \begin{bmatrix} (i^{-1}R_i)^T \\ 0 \end{bmatrix} \begin{bmatrix} (i^{-1}R_i)^T & i^{-1}p_i \end{bmatrix} $$

(2)

Thus the joint twist matrix, $i^{-1}T_i$, is obtained as shown below:

$$ i^{-1}T_i = \begin{bmatrix} i^{-1}\dot{R}_i (i^{-1}R_i)^T & i^{-1}\dot{p}_i - i^{-1}\dot{R}_i (i^{-1}R_i)^T p_i \\ 0 & 0 \end{bmatrix} $$

Further,
\[ i^{-1} T_i = \begin{bmatrix} i^{-1} \dot{R}_i \left( i^{-1} R_i \right)^T & 0 \\ 0 & 0 \end{bmatrix}, \text{for revolute joints} \]

and

\[ i^{-1} T_i = \begin{bmatrix} 0 & \dot{p}_i \\ 0 & 0 \end{bmatrix}, \text{for prismatic joints.} \]

![Figure 1](image)

Figure 1 Two adjacent links, \( i \) and \( i-1 \), in a serial chain.

If the joint axis is aligned with the \( z \) axis, the twist vector is:

\[ i^{-1} t_i = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \dot{\theta}_i, \text{ for revolute joints} \]

\[ = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \dot{d}_i, \text{ for prismatic joints} \]

Note that the joint twist matrix or vector tells us the angular velocity of link \( i \) or for that matter, the end effector (since all the joints between link \( i \) and the end effector are immobile) and the linear velocity of a point on the end effector that is instantaneously coincident with the origin \( O_{i-1} \) if only joint \( i \) is displaced.
7.2 Joint twist in the base frame and end effector frame

Clearly the joint twist matrix can be transformed to the appropriate reference frame. In particular, we can transform it to the base frame (frame 0) by the similarity transformation:

\[
0^{(i-1)T_i} = 0^{A_{i-1}} i^{-1} T_i \left(0^{A_{i-1}}\right)^{-1}
\]

\[
= 0^{A_1} A_2 \cdots i^{-2} A_{i-1} i^{-1} T_i \left(i^{-2} A_{i-1}\right)^{-1} \cdots \left(1^{A_2}\right)^{-1} \left(0^{A_1}\right)^{-1}
\]

\[
= 0^{A_1} A_2 \cdots i^{-2} A_{i-1} i^{-1} T_i i^{-1} A_{i-2} \cdots 2^{A_1} 1^{A_0}
\]

(3)

Note that \(0^{T_i}\) tells us the angular velocity of the end effector and the linear velocity of a point on the end effector that is instantaneously coincident with the origin \(O_0\) in the reference frame 0, if only joint \(i\) is displaced.

Instead, we can also obtain the joint twist matrix in reference frame \(n\) that is attached to the end effector:

\[
n^{(i-1)T_i} = (i^{-1} A_n)^{-1} i^{-1} T_i i^{-1} A_n
\]

\[
= (n^{-1} A_n)^{-1} \cdots (i^{-1} A_{i+1})^{-1} (i^{-1} A_i)^{-1} i^{-1} T_i i^{-1} A_i i^{-1} A_{i+1} \cdots n^{-1} A_n
\]

\[
= n^{A_{n-1}} \cdots i^{A_i} i^{-1} T_i i^{-1} A_i i^{-1} A_{i+1} \cdots n^{-1} A_n
\]

(4)

This matrix gives us the angular velocity of the end effector and a linear velocity of the point \(O_n\) on the end effector in frame \(n\).

Note that both these transformations could have been performed using vector representations of the joint twist. Recalling the usual definitions:

\[
0^{A_{i-1}} = \begin{bmatrix} 0^{R_{i-1}} & 0^{P_{i-1}} \\ 0 & 1 \end{bmatrix}, \quad n^{A_{i-1}} = \begin{bmatrix} n^{R_{i-1}} & n^{P_{i-1}} \\ 0 & 1 \end{bmatrix}
\]

the transformation law for twist vectors yields:
where \((\cdot)\hat{p} (\cdot)\) is the skew-symmetric matrix corresponding to \((\cdot)p(\cdot)\).

### 7.3 Velocity analysis of the manipulator in the base frame (frame 0)

We want to find the components of the angular velocity of the end effector, and the linear velocity of a point on the end effector that is instantaneously at the origin of frame 0, in reference frame 0. We do this by differentiating the position equations (without assuming any joint to be immobile).

\[
0 (i-1 t_i) = 0 \Gamma_{i-1} i-1 t_i = \left[ \begin{array}{cc} 0 R_{i-1} & 0 \\ 0 \hat{p}_{i-1} & 0 R_{i-1} \end{array} \right] i-1 t_i
\]

\[
6 (i-1 t_i) = 6 \Gamma_{i-1} i-1 t_i = \left[ \begin{array}{cc} 6 R_{i-1} & 0 \\ 6 \hat{p}_{i-1} & 6 R_{i-1} \end{array} \right] i-1 t_i
\]

(5)

where \((\cdot)\hat{p} (\cdot)\) is the skew-symmetric matrix corresponding to \((\cdot)p(\cdot)\).

\[
0 T_n = \frac{d}{dt} \left[ 0 A_n \right] \left[ 0 A_n \right]^{-1}
\]

\[
= \frac{d}{dt} \left[ 0 A_1 1 A_2 \ldots n-1 A_n \right] \left[ 0 A_1 1 A_2 \ldots n-1 A_n \right]^{-1}
\]

\[
= \left\{ \frac{\partial}{\partial q_1} \left[ 0 A_1 \right] \dot{q}_1 1 A_2 \ldots n-1 A_n + 0 A_1 \frac{\partial}{\partial q_2} \left[ 1 A_2 \right] \dot{q}_2 \ldots n-1 A_n
\]

\[
+ 0 A_1 1 A_2 \ldots \frac{\partial}{\partial q_n} \left( n-1 A_n \right) \dot{q}_n \right\} \left[ 0 A_1 1 A_2 \ldots n-1 A_n \right]^{-1}
\]

where \(q_i\) is the \(i\)th joint variable (\(\theta_i\) or \(d_i\) depending on the type of joint). Note that each term involves a partial derivative with respect to \(q_i\), which is analogous to keeping all joints except the \(i\)th joint immobile. Multiplying the matrices through we get:
Thus the twist of the end effector in the base reference frame is the sum of the joint twists in the base reference frame. In vector notation:

\[
0 \mathbf{T}_n = 0 \mathbf{T}_1 + 0[1 \mathbf{T}_2] + \ldots + 0[n-2 \mathbf{T}_{n-1}] + 0[n-1 \mathbf{T}_n]
\]

(7)

One can perform the velocity analysis of the manipulator in the end effector frame (frame \( n \)) in exactly the same fashion:

\[
0 \mathbf{t}_n = 0 \mathbf{t}_1 + 0[1 \mathbf{t}_2] + \ldots + 0[n-2 \mathbf{t}_{n-1}] + 0[n-1 \mathbf{t}_n]
\]

(7)

7.4 The manipulator Jacobian matrix

We can write the velocity equations in any reference frame, say \( k \), (so far, \( k=0 \) or \( n \)). In other words, we can write the end effector twist as the sum of the individual joint
twists in any convenient reference frame. If we examine the joint twist matrices (or joint twist vectors) we notice that we can write each joint twist as a product of a scalar joint velocity, \( \dot{q}_i \), and a unit joint twist. Let us denote the unit joint twist at the \( i \)th joint as follows:

\[
\begin{bmatrix}
\ldots
& t_{i-1}
& t_i
\end{bmatrix}
= s_i \dot{q}_i
\]  

Clearly, we can think of \( \dot{q}_i \) as the magnitude of the joint twist while \( s_i \) is the unit joint twist describing the motion of frame \( i \) relative to frame \( i-1 \) in frame \( i \).

Let \( k s_i \) denote the unit joint twist in frame \( k \). Now we can write the velocity equations from Equations (6), in matrix form in frame \( k \):

\[
\begin{bmatrix}
\dot{q}_1 \\
\vdots \\
\dot{q}_{n-1} \\
\dot{q}_n
\end{bmatrix}
= k J \dot{q}
\]

Here \( kJ \) is called the \( 6 \times n \) manipulator Jacobian matrix in frame \( k \). It relates the \( 6 \times 1 \) end effector twist vector to the \( n \times 1 \) vector of joint rates. Given the joint velocities or the rate at which we displace the \( n \) joints, we can obtain the end effector twist. If \( n=6 \), the matrix can be inverted to find the joint rates required to obtain any desired end effector twist.

### 7.5 Examples

#### 7.5.1 Example 1

The forward kinematics for the Stanford Arm like \( R-R-P-R-R-R \) structure shown in Figure 1 is given by the chain of transforms:

\[
\text{Trans}(z, a_1) \text{ Rot}(z, \theta_1) \text{ Trans}(x, a_2) \text{ Rot}(x, \theta_2) \text{ Trans}(z, d_3) \text{ Rot}(x, \theta_4) \text{ Trans}(z, a_4) \\
\text{Rot}(y, \theta_5) \text{ Rot}(z, \theta_6)
\]
Figure 2  The Stanford Arm like $R-R-P-R-R-R$ structure.

Figure 3  The Stanford Arm
We consider 7 reference frames, numbered \( \{F_0\} \) through \( \{F_6\} \) or simply 0 through 6. The zeroth frame is \( x-y-z \), shown in the figure. The transforms below show the intermediate frames:

\[
\begin{align*}
0A_1 &= \text{Trans}(z, a_1) \text{ Rot}(z, \theta_1) \\
1A_2 &= \text{Trans}(x, a_2) \text{ Rot}(x, \theta_2) \\
2A_3 &= \text{Trans}(z, d_3) \\
3A_4 &= \text{Rot}(x, \theta_4) \\
4A_5 &= \text{Trans}(z, a_4) \text{ Rot}(y, \theta_5) \\
5A_6 &= \text{Rot}(z, \theta_6)
\end{align*}
\]

Note the key rules governing the assignment of intermediate frames are:

1. The homogeneous transformation matrix relating adjacent frames must have only one joint variable; and
2. The \( ith \) axis of rotation or translation must be easily identifiable in the \( ith \) frame, making sure that the axis for a rotational joint passes through the origin of the \( ith \) frame.

Table 1 The six unit joint twists for the Stanford Arm like \( R-R-P-R-R-R \) structure.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Frame</th>
<th>Description</th>
<th>6x1 unit joint twist vector, ( s_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>rotation about z axis</td>
<td>([0, 0, 1; 0, 0, 0]^T)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>rotation about x axis</td>
<td>([1, 0, 0; 0, 0, 0]^T)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>translation along z axis</td>
<td>([0, 0, 0; 0, 0, 1]^T)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>rotation along x axis</td>
<td>([1, 0, 0; 0, 0, 0]^T)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>rotation about y axis</td>
<td>([0, 1, 0; 0, 0, 0]^T)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>rotation about z axis</td>
<td>([0, 0, 1; 0, 0, 0]^T)</td>
</tr>
</tbody>
</table>

7.5.2 Example 2

The forward kinematics for the PUMA manipulator shown in Figure 4 is given by the chain of transforms:

\[
\begin{align*}
\text{Trans}(z, a_1) \text{ Rot}(z, \theta_1) \text{ Trans}(x, a_2) \text{ Rot}(x, \theta_2) \text{ Trans}(z, a_3) \text{ Rot}(x, \theta_3) \text{ Trans}(z, a_4) \\
&\quad \text{Trans}(y, -a_5) \text{ Rot}(z, \theta_4) \text{ Rot}(y, \theta_5) \text{ Rot}(z, \theta_6)
\end{align*}
\]
We consider 7 reference frames, numbered \( \{F_0\} \) through \( \{F_6\} \) or simply 0 through 6. The zeroth frame is \( x\text{-}y\text{-}z \), shown in the figure. The transforms below show the intermediate frames:

\[
\begin{align*}
0A_1 &= \text{Trans}(z, a_1) \text{ Rot}(z, \theta_1) \\
1A_2 &= \text{Trans}(x, a_2) \text{ Rot}(x, \theta_2) \\
2A_3 &= \text{Trans}(z, a_3) \text{ Rot}(x, \theta_3) \\
3A_4 &= \text{Trans}(z, a_4) \text{ Trans}(y, -a_5) \text{ Rot}(z, \theta_4) \\
4A_5 &= \text{Rot}(x, \theta_5) \\
5A_6 &= \text{Rot}(z, \theta_6)
\end{align*}
\]

Figure 4 The PUMA manipulator.
Figure 5 The six degree-of-freedom PUMA 560 robot manipulator.

Table 2 The six unit joint twists for the Puma Manipulator.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Frame</th>
<th>Description</th>
<th>6×1 joint twist vector, $s_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>rotation about $z$ axis</td>
<td>$[0, 0, 1; 0, 0, 0]^T$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>rotation about $x$ axis</td>
<td>$[1, 0, 0; 0, 0, 0]^T$</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>rotation along $x$ axis</td>
<td>$[1, 0, 0; 0, 0, 0]^T$</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>rotation about $z$ axis</td>
<td>$[0, 0, 1; 0, 0, 0]^T$</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>rotation along $x$ axis</td>
<td>$[1, 0, 0; 0, 0, 0]^T$</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>rotation about $z$ axis</td>
<td>$[0, 0, 1; 0, 0, 0]^T$</td>
</tr>
</tbody>
</table>

7.6 Geometric method to assemble the Jacobian matrix

We outline a simple procedure for constructing the Jacobian matrix without differentiating any homogeneous transformation matrix.
1. Choose a convenient reference frame, \( k \), in which we want to define the Jacobian matrix. Usually, a reference frame midway between the 0th and the \( nth \) reference frame will be best.

2. Obtain the unit joint twists by inspection as discussed in the examples above.

3. Find the \( n \) homogeneous transformation matrices, \( \mathbf{A}_i \) (\( i=1, 2, \ldots, n \)), that allow transformation from the \( ith \) frame to the \( kth \) frame.

4. For each of the homogeneous transformation matrices, find the corresponding \( 6 \times 6 \) transformation matrix for twists, \( \mathbf{T}_i \).

5. Transform the unit joint twists to the \( kth \) frame:

\[
k \mathbf{s}_i = k \mathbf{T}_i s_i
\]

6. Assemble the Jacobian matrix according to Equation (11).

After going through this procedure several times, it is easy to see that steps 3-5 can be eliminated by directly computing \( k \mathbf{s}_i \). Define the following symbols:

- \( \mathbf{p}_i \) the position vector of a point on axis \( i \)
- \( \mathbf{u}_i \) a unit vector along axis \( i \).

The unit joint twist is given by one of the two following expressions. For revolute joints,

\[
k \mathbf{s}_i = \begin{bmatrix} k \mathbf{u}_i \\
\mathbf{p}_i \times k \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} k \mathbf{R}_i \mathbf{u}_i \\
\mathbf{p}_i \times k \mathbf{R}_i \mathbf{u}_i \end{bmatrix}
\]

while for prismatic joints,

\[
k \mathbf{s}_i = \begin{bmatrix} 0 \\
k \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} 0 \\
k \mathbf{R}_i \mathbf{u}_i \end{bmatrix}
\]

Because \( \mathbf{u}_i \) is easily found from inspection (see Table 1, for example), and the rotation matrices are easy to find from the description of the kinematics, the main difficulty lies in finding \( \mathbf{p}_i \). This is also not difficult if we see that \( ' \mathbf{p}_i \) is the zero vector. Clearly, \( k \mathbf{p}_i \), the position vector in the \( kth \) frame is given by:
Thus, the relevant position vector is immediately available from the homogeneous transformation matrices.

\[ k_\mathbf{q} = \begin{bmatrix} k_\mathbf{R} & k_\mathbf{p} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = k_\mathbf{p}_f \]

(12)

7.7 Computational issues

Once the Jacobian matrix is constructed in frame \( k \), one knows the relationship between joint velocities and end effector velocities in frame \( k \). For a six degree-of-freedom manipulator, the inverse of the Jacobian yields the joint rates required to move the end effector with a desired velocity:

\[
\begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2 \\
\vdots \\
\dot{q}_{n-1} \\
\dot{q}_n
\end{bmatrix} = [k_\mathbf{J}]^{-1} k_\mathbf{J} [0_t_n]
\]

(13)

From a practical viewpoint, one generally specifies the end effector velocities either in the end effector frame (frame \( n \) is favored by Paul) or in the base frame. Thus \( k \) is taken to be either 0 or \( n \). However, since the Jacobian matrix is a complicated function of the joint variables, analytical inversion is extremely difficult. Numerical inversion is possible, but computationally expensive. Remember that the joint rates have to update in real-time (several 100 times per second).

One way of simplifying the expression of the Jacobian and making it possible to perform an analytical inversion is by calculating it on an intermediate frame (somewhere between frame 0 and frame 6). For example, for the Stanford arm or for the PUMA manipulator, it is convenient to compute the Jacobian matrix in frame 3 (i.e., \( k=3 \)). Not
only is analytical expression possible, but one gets a better physical feel for the columns of the Jacobian. This is particularly important for identifying the singularities of the Jacobian matrix. If the Jacobian matrix is singular, the inversion in (13) is not possible. This has important practical implications for the control of the manipulator and it is essential to identify the configurations at which the matrix becomes singular. This task becomes particularly easy in the intermediate reference frame.

### 7.8 Sample Maple code

The following is sample Maple code for computing the Jacobian matrix in Frame 3 for a Stanford Arm like manipulator.

```maple
> restart;
> with(linalg):

Library of procedures for direct and inverse kinematics.

Elemental translations and rotations
> TransX:=x-> vector([x, 0, 0]):
> TransY:=y-> vector([0, y, 0]):
> TransZ:=z-> vector([0, 0, z]):
> RotX:= t -> array(1..3,1..3,[1, 0, 0], [0, cos(t), -sin(t)], [0, sin(t), cos(t)]):
> RotY:=t -> array(1..3,1..3,[cos(t), 0, sin(t)], [0, 1, 0],[-sin(t), 0, cos(t)]):
> RotZ:=t -> array(1..3,1..3,[cos(t), -sin(t), 0], [sin(t), cos(t), 0],[0, 0, 1]):

Homogeneous transformation matrix from rotation matrix and translation vector
> HomTrans:=(R, d) -> array(1..4,1..4,[R[1,1], R[1,2], R[1,3], d[1]], [R[2,1], R[2,2], R[2,3], d[2]], [R[3,1], R[3,2], R[3,3], d[3]], [0, 0, 0, 1]):

Rotation matrix and translation vector from the homogeneous transformation matrix from

Skew symmetric matrix operator corresponding to a 3x1vector.
> SkewMatrixOp := a -> array(1..3,1..3,[0, -a[3], a[2]], [a[3], 0, -a[1]], [-a[2], a[1], 0]):

6x6 twist transformation matrix (Gamma) corresponding to a 4x4 homogeneous transformation matrix.
> AdjOp:= proc(A) local X, R; R:=RotHomTrans(A); X:=evalm(SkewMatrixOp(TransHomTrans(A)) &* R); array(1..6,1..6, [[R[1,1], R[1,2], R[1,3], 0, 0, 0], [R[2,1], R[2,2], R[2,3], 0, 0, 0], [R[3,1], R[3,2], R[3,3], 0, 0, 0], [X[1, 1], X[1, 2], X[1, 3], R[1, 1], R[1, 2], R[1, 3]], [X[2, 1], X[2, 2], X[2, 3], R[2, 1], R[2, 2], R[2, 3]], [X[3, 1], X[3, 2], X[3, 3], R[3, 1], R[3, 2], R[3, 3]]]):

3x1 vector corresponding to a skew symmetric matrix operator, 6x1 twist vector corresponding to a twist matrix.
```

---

Inverse of a homogeneous transformation matrix
> InvHomTrans:=proc(A) local R; R:=transpose(RotHomTrans(A)); HomTrans(R,
scalarmul(multiply(R, TransHomTrans(A)), -1)) end;

Abbreviate cos(ti) by ci, sin(ti) by si.
> alias(seq(c.i=cos(t.i), i=1..6), seq(s.i=sin(t.i), i=1..6));

Direct Kinematics
> A1:=HomTrans(RotZ(t1), TransZ(a1));
> A2:=HomTrans(RotX(t2), TransX(a2));
> A3:=HomTrans(RotZ(0), TransZ(d3));
> A4:=HomTrans(RotX(t4), TransX(0));
> A5:=HomTrans(RotY(t5), TransZ(a4));
> A6:=HomTrans(RotZ(t6), TransZ(0));
> Tool:=HomTrans(RotZ(0), TransZ(a5));
> A:=multiply(A1, A2, A3, A4, A5, A6);

Jacobian
Compute Jacobian in an intermediate kth frame, say k= 3. First compute all the joint twists in a local frame. Twist 1 in frame 1, Twist 2 in frame 2, etc..
> for i from 1 to 6 do T.i:=ExtractTwist(simplify(evalm(InvHomTrans(A.i)&*map(diff, A.i, t.i)))) od;
T1 := [0, 0, 1, 0, 0, 0]
T2 := [1, 0, 0, 0, 0, 0]
T3 := [0, 0, 0, 0, 0, 0]
T4 := [1, 0, 0, 0, 0, 0]
T5 := [0, 1, 0, 0, 0, 0]
T6 := [0, 0, 1, 0, 0, 0]
The expression for the third joint twist is not correct. We need to differentiate with respect to d3. This is corrected below.
> T3:=ExtractTwist(simplify(evalm(InvHomTrans(A3)&*map(diff, A3, d3))));
T3 := [0, 0, 0, 0, 0, 1]
Now we obtain the columns of the Jacobian matrix.
> J1:=simplify(evalm(AdjOp(InvHomTrans(multiply(A2, A3)))&* T1));
> J2:=simplify(evalm(AdjOp(InvHomTrans(A3))&* T2));
> J3:=evalm(T3);
> J4:=simplify(evalm(AdjOp(A4)&* T4));
> J5:=simplify(evalm(AdjOp(multiply(A4,A5))&* T5));
> J6:=simplify(evalm(AdjOp(multiply(multiply(A4,A5,A6))&* T6)));
We concatenate the columns to obtain the Jacobian matrix.
> J:=concat(seq(J.i, i=1..6));
[ 0 1 0 1 0 s5 ]
[ s2 0 0 0 c4 -s4 c5 ]
[ c2 0 0 0 s4 c4 c5 ]
[ s2 d3 0 0 0 -a4 0 ]
[ a2 c2 -d3 0 0 0 c4 a4 s5 ]
[ -a2 s2 0 1 0 0 s4 a4 s5 ]
> simplify(det(J));
2 s2 d3 c5