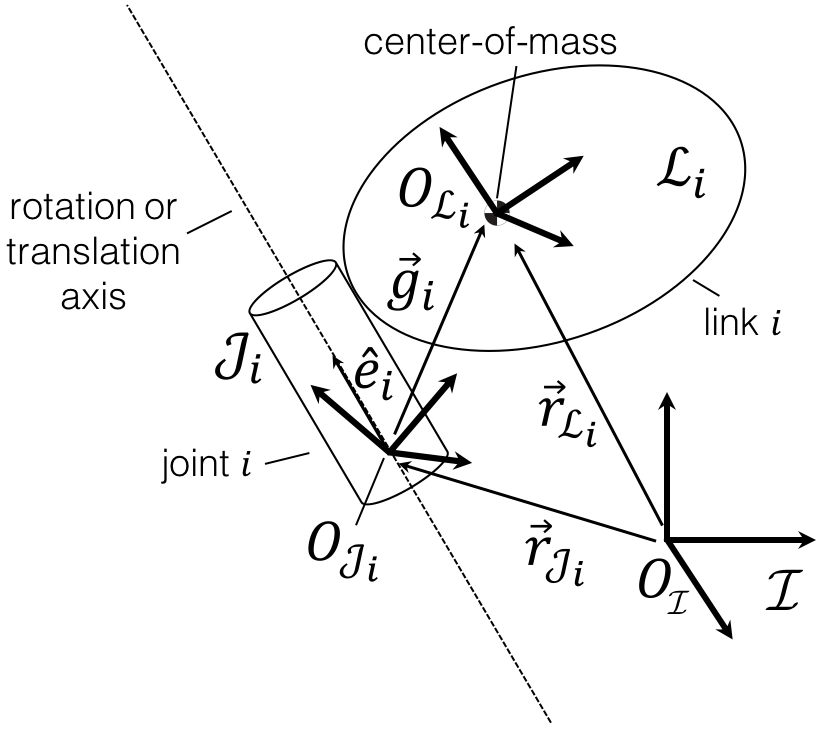
SPART Tutorial – Kinematics

Direct Kinematics

SPART can compute the position and orientation of all the links and joints. The definitions of the kinematic quantities of a generic link and joint are notionally shown in the following figure.



Schematic disposition of a generic link and its associated joint.

To obtain the kinematics of the system, the base-link position $\mathrm{r}_0 \in \mathrm{R}^3$ and orientation, as a rotation matrix $\mathrm{R}_0 \in \mathrm{SO}(3)$, with respect to the inertial CCS are first specified.

```
%Base-link position and orientation
R0=eye(3);  %Rotation from base-link with respect to the inertial CCS.
r0=[0;0;0]; %Position of the base-link with respect to the origin of the inertial frame, projected
in the inertial CCS.
```

In SPART, the vectors are represented by a 3-by-1 column matrix containing the components of the vector projection to the inertial CCS. Projections to other CCS are explicitly marked.

Note

The position of the base-link `r0` refers to the base-link center-of-mass, corresponding with the origin of the URDF `inertial` tag. The base-link orientation `R0` also corresponds with the orientation of the CCS specified in the URDF `inertial` tag.

The joint displacements, $\mathbf{q}_m \in \mathrm{R}^n$, also also defined as a n-by-1 column matrix.

$$\mathbf{q}_m = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}$$

```
%Joint displacements
qm=[0;0;0];
```

If the ith joint is revolute, q_i denotes an angular displacement around the rotation axis $\hat{e}_i$, whether if the ith joint is prismatic, q_i denotes a translational displacement along the sliding axis $\hat{e}_i$.

The set of $\mathbf{R}_0$, $\mathbf{r}_0$, $\mathbf{q}_m$ constitute a set of variables $\mathcal{Q}$, known as *generalized variables*, which full define the state of the multibody system,

$$\mathcal{Q} = \left\{ \mathbf{R}_0, \mathbf{r}_0, \mathbf{q}_m \right\}$$

With the generalized variables specified, SPART is ready to compute the kinematics of the system.

```
%Kinematics
[RJ,RL,rJ,rL,e,g]=Kinematics(R0,r0,qm,robot);
```

The output of the `Kinematics` function is as follows:

- RJ – Joint 3x3 rotation matrices – as a [3x3xn] matrix.
- RL – Links 3x3 rotation matrices – as a [3x3xn] matrix.
- rJ – Positions of the joints projected in the inertial CCS – as a [3xn] matrix.
- rL – Positions of the links projected in the inertial CCS – as a [3xn] matrix.
- e – Joint rotation/sliding axis projected in the inertial CCS – as a [3xn] matrix.

- g – Vector from the origin of the ith joint CCS to the origin of the ith link CCS, projected in the inertial CCS – as a [3xn] matrix.

The results for each link/joint are stacked into a single variable. For example, to get the position of the second link center-of-mass:

```
%Position of the center-of-mass of a link
i=2;
rL(1:3,i)
```

and the rotation matrix corresponding to the second link CCS:

```
%Position of the center-of-mass of a link
i=2;
RL(1:3,1:3,i)
```

If your Matlab installation includes the Symbolic Math Toolbox SPART is able to obtain the analytic expressions of these kinematic quantities. To do so, just define the generalized variables as *symbolic expressions*.

```
%Base-link position
r0=sym('r0',[3,1],'real');

%Base-link orientation
Euler_Ang=sym('Euler_Ang',[3,1],'real');
R0 = Angles321_DCM(Euler_Ang)';

%Joint displacements
qm=sym('qm',[robot.n_q,1],'real');

%Kinematics
[RJ,RL,rJ,rL,e,g]=Kinematics(R0,r0,qm,robot);
```

Warning

To obtain analytic expressions, all inputs must be symbolic. Otherwise, errors may occur.

Differential kinematics

The angular and linear velocities of the ith link with respect to the inertial frame, projected in the inertial CCS, are encapsulated into the **twist** variable $\mathbf{t}_i \in \mathrm{R}^6$.

$$\mathbf{t}_i = \begin{bmatrix} \omega_i \\ \dot{\mathbf{r}}_i \end{bmatrix}$$

The twist can be recursively propagated outward from one link to the next using the 6-by-6 $\mathbf{B}_{ij}$ twist–propagation matrix and the 6-by-1 $\mathbf{p}_i$ twist–propagation "vector":

$$\mathbf{t}_i = \mathbf{B}_{ij}\mathbf{t}_j + \mathbf{p}_i\dot{q}_i \quad \text{for} \quad j = i - 1$$

The twist–propagation matrices and "vectors", which form the basis of the differential kinematics, can be computed with the `DiffKinematics` function.

```
%Differential kinematics
[Bij,Bi0,P0,pm]=DiffKinematics(R0,r0,rL,e,g,robot);
```

The output of the differential kinematics is as follows:

- Bij – Twist–propagation [6x6xn] matrix (for manipulator i>0 and j>0).
- Bi0 – Twist–propagation [6x6xn] matrix (for i>0 and j=0).
- P0 – Base–link twist–propagation [6x6] matrix.
- pm – Manipulator twist–propagation [6xn] vector.

The set of generalized velocities $\mathbf{u} \in \mathrm{R}^{6+n}$ (joint-space velocities) contains the base-link velocities $\mathbf{u}_0 \in \mathrm{R}^6$ and the joint velocities $\mathbf{u}_m \in \mathrm{R}^n$.

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_m \end{bmatrix}$$

With the base-link and joint velocities defined as:

$$\mathbf{u}_0 = \begin{bmatrix} \omega_{0\{\mathcal{L}_0\}} \\ \dot{\mathbf{r}}_0 \end{bmatrix}$$

$$\mathbf{u}_m = \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

Note that $\omega_{0\{\mathcal{L}_0\}}$ denotes the angular velocity of the base-link, with respect to the inertial frame, projected in the base-link body-fixed CCS (this is the angular velocity that is obtained when using an onboard rate-gyro).

For the base-link, the twist is computed only using a modified 6-by-6 $\mathbf{P}_0$ twist-propagation matrix.

$$\mathbf{t}_0 = \mathbf{P}_0 u_0$$

With the twist-propagation quantities and the generalized velocities, the twists of all the links (operational-space velocities) can be determined.

```
%Generalized velocities (joint-space velocities)
u0=zeros(6,1); %Base-link angular (projected in the base-link body-fixed CCS) and linear
velocities.
um=[4;-1;5]*pi/180; %Joint velocities

%Twist (operational-space velocities)
[t0,tL]=Velocities(Bij,Bi0,P0,pm,u0,um,robot);
```

The output of the operational space velocities are as follows:

- t0 – Base–link twist projected in the inertial CCS – as a [6x1] matrix.
- tL – Manipulator links twist projected in the inertial CCS – as a [6xn] matrix.

Jacobians

The geometric Jacobian of a point p maps the joint-space velocities $\mathbf{u}$ into operational-space velocities of that point $\mathbf{t}_p$.

$$\mathbf{t}_p = \mathbf{J}_p \mathbf{u}$$

The contribution from the base-link and from the joints can be written more explicitly as:

$$\mathbf{J}_p = \begin{bmatrix} \mathbf{J}_{0p} & \mathbf{J}_{mp} \end{bmatrix}$$
$$\mathbf{t}_p = \mathbf{J}_{0p}\mathbf{u}_0 + \mathbf{J}_{mp}\mathbf{u}_m$$

The Jacobian of a point p, fixed to the ith link, can be obtained as follows:

```
%Jacobian of a point p in the ith link
%rp is the position of the point p, projected in the inertial CCS -- as a [3x1] matrix.
[J0p, Jmp]=Jacob(rp,r0,rL,P0,pm,i,robot);
```

The Jacobians corresponding to the center-of-mass of the the ith link of the multibody system are then computed as follows:

```
%Jacobian of the ith Link
[J0i, Jmi]=Jacob(rL(1:3,i),r0,rL,P0,pm,i,robot);
```

Accelerations

The accelerations of a link can be encapsulated in a twist-rate $\dot{\mathbf{t}}_i \in \mathrm{R}^6$:

$$\dot{\mathbf{t}}_i = \begin{bmatrix} \dot{\omega}_i \\ \ddot{\mathbf{r}}_i \end{bmatrix}$$

The generalized accelerations $\dot{\mathbf{u}} \in \mathrm{R}^{6+n}$ of the system are defined as:

$$\dot{\mathbf{u}} = \begin{bmatrix} \dot{\mathbf{u}}_0 \\ \dot{\mathbf{u}}_m \end{bmatrix}$$

The twist-rate can then be computed as follows:

```
%Define generalized accelerations
u0=zeros(6,1); %Base-link angular (projected in the base-link body-fixed CCS) and linear
accelerations
um=[-0.1;0.2;0.1]*pi/180; %Joint accelerations

%Accelerations, twist-rate
[t0dot,tLdot]=Accelerations(t0,tL,P0,pm,Bi0,Bij,u0,um,u0dot,umdot,robot)
```

Jacobian time derivative

The time derivatives of the Jacobians can also be obtained:

```
%Jacobain time derivative
%rp is the position of the point p, projected in the inertial CCS -- as a [3x1] matrix.
%tp is the twist of the point p -- as a [6x1] matrix.
[J0pdot, Jmpdot]=Jacobdot(rp,tp,r0,t0,rL,tL,P0,pm,i,robot)
```

The Jacobian time derivative can be used to obtain the twist-rate of a point on the multibody system.

$$\dot{\mathbf{t}}_p = \mathbf{J}_p \dot{\mathbf{u}} + \dot{\mathbf{J}}_p \mathbf{u}$$