

The Kinematics–DSL manual

Marco Frigerio *

July 2017

Contents

1	Introduction	1
1.1	Publications	2
2	Example document	2
3	Common definitions	4
4	Coordinate frames	4
4.1	Placement of coordinate frames	5
4.2	Example	7
4.3	Additional remarks	7
4.4	Optional frames	8
5	Inertia properties	8
6	Parametrization	10
6.1	Kinematics parameters	11
6.2	Dynamics parameters	11

1 Introduction

This manual describes a Domain Specific Language designed to represent the rigid–body model of articulated mechanisms, such as humanoid robots and manipulators. The language is called “Kinematics–DSL”. At the current version, the language only supports mechanisms without kinematic loops.

This language is the foundation of *The Robotics Code Generator* (*RobCoGen* in short), a computer program that generates optimized code for various kinematics and dynamics routines.¹ The Kinematics–DSL is very simple (in fact simplicity is one of its features), and you can learn the syntax simply by looking at an example, such as the one of Section 2.

This document is under development. Please check for updates on the wiki of *RobCoGen*.

*Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT)

¹More information and downloads about the code generator can be found in this website: <https://robcoigenteam.bitbucket.io/>. The code generator takes a robot model (i.e. a document of the Kinematics–DSL described in this article) as its primary input.

1.1 Publications

For further insight on the rationale of the language and the code generation please have a look at the following publications:

- Marco Frigerio, Jonas Buchli, Darwin G. Caldwell, and Claudio Semini (2016). “RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on Domain Specific Languages”. In: *Journal of Software Engineering for Robotics (JOSE)* 7.1 (Special Issue on Domain-Specific Languages and Models for Robotic Systems), pp. 36–54
- Marco Frigerio, Jonas Buchli, and Darwin G. Caldwell (2012). “Code Generation of Algebraic Quantities for Robot Controllers”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*

2 Example document

Before going through the description of the language, here is a complete example of a document of the Kinematics-DSL, which represents a fictitious 5-dof robot.

```
Robot Fancy {

/*
 * Kinematic chain without branches.
 * All the links are hollow cylinders infinitely thin (internal
 * and external radii are the same) with radius equal to 0.05
 * meters, unit length, unit mass.
 * They are connected by five joints: revolute, prismatic,
 * revolute, prismatic, revolute.
 */

RobotBase FancyBase {
    inertia_properties {
        mass = 1.0
        CoM = (0.0, 0.0, 0.0)
        Iy=0.0 Ix=0.0 Ixy=0.0 Iz=0.0 Ixz=0.0 Iyz=0.0
    }
    children {
        link1 via jA
    }
}

link link1 {
    id = 1
    inertia_properties {
        mass = 1.0
        CoM = (0.5, .0, .0)
        Ix=0.0025 Iy=0.33458 Iz=0.33458
        Ixy=0.0 Ixz=0.0 Iyz=0.0
    }

    children {
        link2 via jB
    }
}

link link2 {
```

```

    id = 2
    inertia_properties {
        mass = 1.0
        CoM = (0.0, 0.0, 0.5)
        Ix=0.33458  Iy=0.33458  Iz=0.0025
        Ixy=0.0  Ixz=0.0  Iyz=0.0
    }

    children {
        link3 via jC
    }
}

link link3 {
    id = 3
    inertia_properties {
        mass = 1.0
        CoM = (0.5, .0, .0)
        Ix=0.0025  Iy=0.33458  Iz=0.33458
        Ixy=0.0  Ixz=0.0  Iyz=0.0
    }

    children {
        link4 via jD
    }
}

link link4 {
    id = 4
    inertia_properties {
        mass = 1.0
        CoM = (0.0, 0.0, 0.5)
        Ix=0.33458  Iy=0.33458  Iz=0.0025
        Ixy=0.0  Ixz=0.0  Iyz=0.0
    }

    children {
        link5 via jE
    }
}

link link5 {
    id = 5
    inertia_properties {
        mass = 1.0
        CoM = (0.5, .0, .0)
        Ix=0.0025  Iy=0.33458  Iz=0.33458
        Ixy=0.0  Ixz=0.0  Iyz=0.0
    }

    children {}
    frames {
        ee {
            translation = (ee_x, 0.0, 0.0)
            rotation    = (0.0, 0.0, 0.0)
        }
    }
}

r_joint jA {          // 'r_' stands for revolute
    ref_frame {

```

```

        translation = (0.0, 0.0, 0.0)
        rotation = (0.0, 0.0, 0.0)
    }
}

p_joint jB {          // 'p_' stands for prismatic
    ref_frame {
        translation = (1.0, 0.0, 0.0)
        rotation = (-PI/2.0, 0.0, 0.0)
    }
}

r_joint jC {
    ref_frame {
        translation = (0.0, 0.0, 1.0)
        rotation = (0.0, 0.0, 0.0)
    }
}

p_joint jD {
    ref_frame {
        translation = (1.0, 0.0, 0.0)
        rotation = (PI/2.0, 0.0, 0.0)
    }
}

r_joint jE {
    ref_frame {
        translation = (0.0, 0.0, 1.0)
        rotation = (0.0, 0.0, 0.0)
    }
}
}
}

```

3 Common definitions

This section establishes some common terminology for the description of a kinematic tree.

- the “supporting joint” of a link is the joint the moves the link (and thus the whole kinematic subtree rooted at the link – e.g. an elbow with respect to the lower arm); conversely, the “supporting link” of a joint is the link carrying that joint (the upper arm with respect to the elbow);
- for any link P - joint J - link S section of the tree, where P is the link supporting joint J and S is the link supported in turn by J, we call P and S respectively the *predecessor* and the *successor* of J (Featherstone 2008).

All the numerical values used in a document to describe the robot must use the standard SI units.

4 Coordinate frames

The first thing to know about the Kinematics-DSL is the convention about the placement of the reference coordinate frames on the mechanism. Any numerical

property of a mechanism model is expressed in a specific coordinate frame, so obviously one must know where the frames are physically located in order to measure such properties.

The convention is detailed in chapter 4 of (Featherstone 2008). What I describe here is a simplified, less general version for kinematic trees (no kinematic loops).

4.1 Placement of coordinate frames

Figure 1 shows a generic section of a kinematic tree, in the form predecessor-joint-successor, as mentioned before. In the rest of the text we will often refer

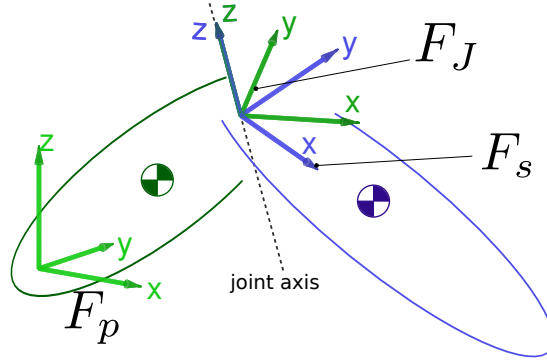


Figure 1: Placement of reference frames on a generic section of the kinematic tree, formed by two subsequent links and the connecting joint (a revolute joint, in this example). F_p is the frame of the predecessor link (green), F_J the frame of the joint, F_s the frame of the successor link (blue).

to the objects in this figure.

A systematic description of the convention follows:

1. The joint-frame F_J is located at the joint, such that the z axis is coincident with the joint axis (the rotation axis for a revolute joint or the translation axis for a prismatic joint). This constraint is the basis of the convention. The joint frame does *not* move during the motion of the joint itself.
2. The link-frame of the successor, F_s , has the same origin and same z axis of the supporting-joint-frame F_J . F_s is attached to the link and does move as the link moves.
3. F_s and F_J coincide for a single, specific configuration of the link. When this condition is verified, the joint status variable is zero.
4. The coordinate transform between F_J and F_s depends solely on the joint status variable.
5. The actual pose of the joint frame F_J is specified with respect to the frame F_p of the predecessor link. In a Kinematics-DSL document this specification consists of 6 values (see Figure 2):

- 3 for the translation vector

- 3 for the orientation: these numbers are basically euler angles, that tell the rotation about the x , y and z axis respectively. Each rotation is about the axis *after the rotation* specified by the previous parameter

We call this group of 6 values the *joint frame parameters*.

- The joint-frame parameters must respect the constraint described in point 1, that is, the z axis must coincide with the joint axis. As the latter is obviously determined by the actual mechanism, and it is not arbitrary, it follows that the joint-frame parameters specify the location in space of the joint axis, with F_p coordinates. Therefore all the **joint** paragraphs in a document (see Section 2 and Figure 2) provide all the geometric information about the mechanism, i.e. where the joints are and what is the orientation of the axes. These values do not depend on any joint status. The transform between F_p and F_J is constant; it is a geometric property of the robot.²
- Five parameters are enough to specify a line in space, which means that the six joint frame parameters are redundant. The additional degree of freedom allows to specify where one wants the link to be when the joint status variable is zero. That is, which is the robot configuration when the joint status vector is zero.
- For the robot base, the placement of the link frame is arbitrary.

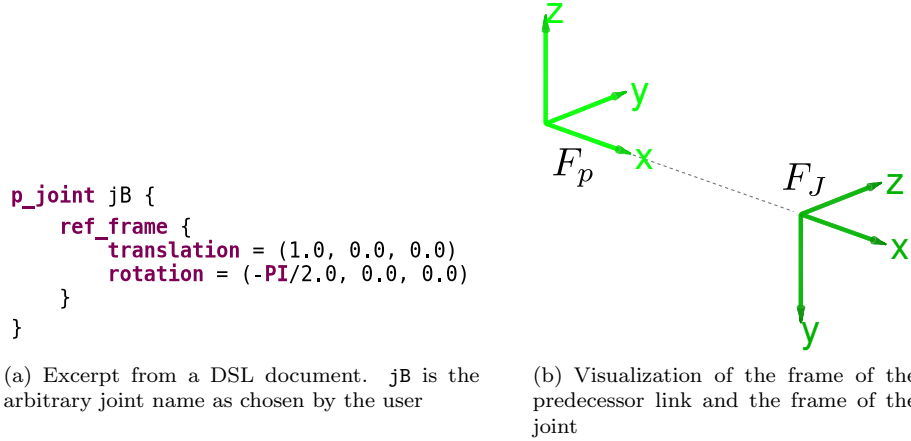


Figure 2: Specification of the location of the joint frame F_J with respect to the supporting link frame F_p . In this example F_J has its origin at a distance of one meter along the x , and it is also rotated of $-\pi/2$ about the same x axis. Note that both F_p and F_J lay on the same rigid body, and their relative pose is a geometric constant of the mechanism model.

²If the robot model is parametric, these values can be possibly changed. See section 6.

4.2 Example

As an example, consider a generic, linear kinematic chain made by N moving links, plus the fixed base. Here are the steps you would need to take in order to work out the joint-frame-parameters in the document describing your robot:

1. Choose where to put the base-frame; this is completely arbitrary and it is up to you.
2. Choose where to put the frame of link 1, with respect to the actual geometry of the link itself. We call this frame F_s . The z axis of F_s must be on the joint axis, wherever the axis would be when the link is assembled with the base. Choose a pose that would feel intuitive or convenient for you. Remember F_s is attached to the link, and the two will always move together.
Note that, so far, we have not determined any number to be inserted in the robot model.
3. Establish the position of the joint-frame F_J for joint 1 (the one between the base and link 1), with respect to the base-frame chosen initially. Work out the parameters expressing the pose of F_J with base-frame coordinates. Perform this step considering the joint status to be zero. Also consider that:
 - The z axis of F_J must be on the joint axis.
 - The final pose of F_J must be consistent with the pose of link 1 in the assembled mechanism, in the sense that F_J and F_s must coincide. At this point, you might want to go back to step 2 to revise your choice of F_s .
 - Effectively, only one degree of freedom for the positioning of F_J is left. Choose it to have the desired mechanism configuration when the joint status is zero.
4. Now you already know the pose of the frame of the first moving link, (link 1) as it coincides with F_J . Therefore there is no need for any further parameter for the link-frame, which is fully determined.
5. Having established the pose of the link-frame for link 1, proceed to determine the pose of the joint-frame for joint 2 (which is supported by link 1). Apply again steps 2 and 3 to the new targets joint 2 and link 2.
6. Iterate this procedure for the subsequent links/joints.

4.3 Additional remarks

In the case of a revolute joint, the origin of F_J is constrained, and the remaining degree of freedom corresponds to the orientation of the x and y axes. In this case, the degree of freedom directly maps to one of the joint-frame parameters, the rotation about the z axis.

For a prismatic joint, instead, it is the location of the origin along the joint axis to be arbitrary. It is still a single degree of freedom, which however does not map in general to a single parameter.

Once again, in both cases the choice of the last degree of freedom in the positioning of F_J determines the status of the mechanism when the joint status vector is conventionally said to be zero.

Finally, note that the base-frame can possibly be chosen so as to coincide with the first link frame, since its location is entirely up to the user. In this case, all the 6 joint-frame parameters of the first joint would be zero.

4.4 Optional frames

The previous paragraphs discuss the placement of the main reference frames on the robot. Optionally, other additional reference frames on the links can be defined (e.g. to model the pose of a certain sensor), and the corresponding coordinate transforms be generated. Any optional frame belongs to a specific link, and it is defined in turn with respect to the default link-frame, using the exact same syntax as the one used for the joint frames. See Figure 3 for an example.

```
link link1 {
  id = 1
  inertia_params {
    mass = 1.0
    CoM = (0.5, .0, .0)
    Ix=0.0025 Iy=0.33458 Iz=0.33458 Ixy=0.0 Ixz=0.0 Iyz=0.0
  }
  children { link2 via jB }

  frames {
    customFrame1 {
      translation = (0.33, 0.0, 0.0)
      rotation = (0.0, 0.0, 0.0)
    }
    customFrame2 {
      translation = (0.0, 0.0, 0.0)
      rotation = (0.15, 0.0, -1.33)
    }
  }
}
```

Figure 3: Definition of additional, custom frames on a link of the mechanism. Translation and rotation parameters are expressed with respect to the default link-frame, in the same way used for the joint frame. Note though, that an additional name must be provided in this case. Additional reference frames are optional.

5 Inertia properties

Any link definition must contain the value of its inertia properties: total mass (Kilograms), position of the center of mass (coordinates in meters), moments of inertia (Kilograms * meter squared). The moments of inertia are basically the elements of the 3×3 inertia tensor, *but some signs are different*; here are the definitions of all the six moments and the definition of the inertia tensor (just for the discrete case; anyway what matters are the signs of the off-diagonal elements):

$$I_x = \sum_i m_i (y_i^2 + z_i^2) \quad (1)$$

$$I_y = \sum_i m_i (x_i^2 + z_i^2) \quad (2)$$

$$I_z = \sum_i m_i (x_i^2 + y_i^2) \quad (3)$$

$$I_{xy} = \sum_i m_i x_i y_i \quad (4)$$

$$I_{xz} = \sum_i m_i x_i z_i \quad (5)$$

$$I_{yz} = \sum_i m_i y_i z_i \quad (6)$$

Where m_i is the mass of the i – *th* point of the body, while x_i , y_i , z_i are its coordinates.

The inertia tensor is defined as follows:

$$I = \begin{pmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{pmatrix} \quad (7)$$

A Kinematics–DSL document requires I_x , I_y , I_z , I_{xy} , I_{xz} , I_{yz} , in any order, *not* the elements of the tensor. Thus, if you happen to have the inertia tensor of your link (e.g. from the documentation of the robot), then you will have to switch the signs of the off-diagonal elements before filling the values in the document. See Figure 4.

```
link link1 {
  id = 1
  inertia_params {
    mass = 1.0
    CoM = (0.5, .0, .0)
    Ix=0.0025 Iy=0.33458 Iz=0.33458 Ixy=0.0 Ixz=0.0 Iyz=0.0
  }
  children { link2 via jB }

  frames {
    customFrame1 {
      translation = (0.33, 0.0, 0.0)
      rotation = (0.0, 0.0, 0.0)
    }
    customFrame2 {
      translation = (0.0, 0.0, 0.0)
      rotation = (0.15, 0.0, -1.33)
    }
  }
}
```

Figure 4: Inertia properties of a generic link. The numbers in this example refer to a hollow cylinder infinitely thin (internal and external radii are the same) with radius equal to 0.05 meters, unit length, unit mass. The link-frame has its origin in the center of one of the bases of the cylinder, with the x axis aligned with the height of the cylinder. See the full robot example in Section 2.

The position of the center of mass (COM) and the inertia tensor elements are expressed by default with respect to the link-frame (which is *not* located at the COM, in general).

However, the language supports the use of a custom reference frame associated with the inertia properties. This feature relieves the user from computing a roto-translation of the inertia properties, which can be confusing, in case values in a different coordinate system are available (e.g. robot documentation). The reference frame used for the inertia properties must in turn be defined as described in Section 4.4. Figure 5 illustrates the feature with an example. Please

```

link link3 {
  id = 3
  inertia_params {
    mass = 1.0
    CoM = (0.0, 0.0, 0.0)
    Ix = 0.0025  Iy = 0.08458  Iz = 0.08458
    Ixy = 0.0    Ixz = 0.0    Iyz = 0.0
    ref_frame = fr_com
  }
  [...]
}

frames {
  // A reference frame with origin in the COM:
  fr_com {
    translation = (0.5, 0.0, 0.0)
    rotation = (0.0, 0.0, 0.0)
  }
}

```

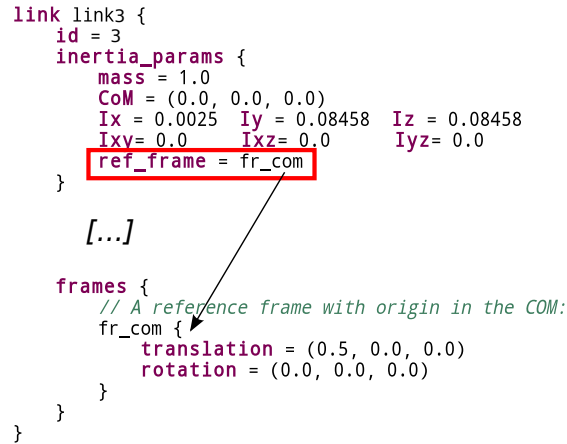


Figure 5: Inertia properties expressed with respect to a user-defined reference frame. The rigid body (link) of this example is equivalent to the one of Figure 4, but some values are different because they are expressed in a different frame. In particular, such a frame has the origin in the COM of the link, and it is oriented as the default link frame (all the rotation parameters are zero) – note that in such a frame the coordinates of the COM itself are all zeros).

note that, due to the difficulty of testing this feature, it has to be considered experimental.

6 Parametrization

Since March 2014, the Kinematics-DSL allows to specify *parametric* robot models, that is, robot descriptions where certain properties are not defined by a numerical constants. Arbitrary identifiers can be input instead of number literals.

Geometric and inertial parameters form two separate classes. Any parameter in the specification of a reference frame is a geometric parameter, whereas those appearing in the inertia properties of links are inertial parameters. The two classes are also called kinematics and dynamics parameters, respectively. Kinematics parameters are further partitioned into lengths and angles.

These classes do not appear explicitly in the documents, as the parameters are classified automatically depending on the context. The classification enables code generators to deal with different parameters with different strategies. See Figure 6 for a couple of examples about the parametrization of the robot model.

The parametrization effectively enables to model a *class* of equivalent mechanisms rather than a specific one. Code generators of the Kinematics-DSL can preserve this condition and generate code where the parameters can be changed at runtime. For more information please refer to the wiki of *RobCoGen* (see the introduction of this manual).

<pre> p_joint jB { ref_frame { translation = (1.0, 0.0, 0.0) rotation = (jB_rx, 0.0, 0.0) } </pre>	<pre> inertia_params { mass = ll_mass CoM = (0.5, .0, .0) Ix=0.0025 Iy=0.33458 Iz=0.33458 Ixy=0.0 Ixz=0.0 Iyz=0.0 } </pre>
(a) A kinematics parameter	(b) A dynamics parameter

Figure 6: Two examples of the usage of parameters in a robot model. In (a) the joint frame is rotated by jB_rx radians about the x axis of the link-frame (see Figure 2). In (b) the mass of the link is unspecified; note, though, that the inertia moments are constant and thus correspond to a specific value of the mass (1kg in this case, see Figure 4). Care must be taken by the user to avoid inconsistencies in parametric inertia properties.

6.1 Kinematics parameters

Kinematics parameters allow to parametrize the position of the joints of the mechanism, e.g. to model a link of varying length. The orientation of the joint frame can also be parametric, effectively modeling a variable joint axis (which is the z axis of the frame, according to the convention), as well as a variable default configuration of the mechanism.

6.2 Dynamics parameters

Dynamics parameters affect the inertia properties of the rigid links. Any property can be parametric, but care must be taken to avoid inconsistencies.

For example, as shown in figure 6(b), one might parametrize only the mass of the link. However the inertia moments also depend on the mass, and the constant values of the example have been computed considering a specific value for it (see equations from 1 to 6). Some mechanism to ensure the consistency should be adopted. For instance, the initial, default value of the mass parameter could be set to the value used to compute the inertia moments, and when the user decides to change the mass also the other properties are modified accordingly, even though they are not parametric. In fact, one might want to change the mass without changing the inertia moments, which basically corresponds to a change in the shape of the rigid body. The specific policy depends on the user needs.