Singularity Set Computation: A Hands-On Session with the CUIK Suite

Oriol Bohigas

Abstract This chapter provides an introduction to the analysis of the singularities of robot mechanisms using the CUIK suite software. The CUIK suite is an open-source toolbox for motion analysis of general closed-chain mechanisms, resulting from several years of research and development within the Kinematics and robot design group at the Institut de Robòtica i Informàtica Industrial. It is available under GPLv3 license from the CUIK Project home page. The intention is not to provide thorough definitions or developments, but to illustrate the basic concepts around singularity analysis through a short and simple presentation. The text assumes basic knowledge of the kinematics of robot mechanisms, such as the notions of configuration space, or the forward and inverse kinematics problems. An exhaustive description of the methods, algorithms, and the underlying mathematical concepts used by the CUIK suite can be found in Bohigas et al. (2016). For a better understanding, we do strongly recommend to install the CUIK suite and execute the examples provided while reading this chapter.

1 Introduction

The so-called singularities of a robot mechanism are special postures of the mechanism in which the kinetostatic behavior may degrade dramatically. Rigidity or dexterity losses arise, and there may appear unresolvable or uncontrollable end-effector forces, among other effects (Bohigas et al. 2016). Singularities generally pose problems for the normal operation of a robot, and thus, they should be analyzed before the actual construction of a prototype.

The CUIK suite provides a toolbox of methods for singularity computation and avoidance to this end. They are applicable to nonredundant mechanisms of large generality and allow a detailed study in the initial phases of the development of a mechanism. The computation of the singularity set provides comprehensive

O. Bohigas (✉)
Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
e-mail: bohigas@upc.edu; obohigas@iri.upc.edu

information on the local and global motion capabilities of a mechanism: Its projections onto the task and joint spaces determine the working regions in such spaces, may inform about the existence of different assembly configurations, and highlight areas where control or dexterity losses may arise. These projections also supply a fair view of the feasible movements of the mechanism, although they do not reveal all possible singularity-free motions. In order to also tackle this issue, the CUIK suite includes general path planning methods that can generate movements of the mechanism that avoid problematic singularities (Porta et al. 2014).

Overall, the CUIK suite can help eliminate barriers in design creativity, and it contributes to the general understanding on how the motions of complex multibody systems can be predicted, planned, and controlled in an efficient and reliable way.

This chapter concentrates on singularity computation, and the interested reader on singularity avoidance can refer to Bohigas et al. (2016).

1.1 Basic Concepts

Configuration space: Usually, all the feasible configurations of a mechanism can be characterized by a system of equations

$$\Phi(\mathbf{q}) = \mathbf{0}, \tag{1}$$

which expresses the assembly constraints between the different bodies of the robot, and $\mathbf{q}$ is the vector of configuration coordinates (see Sect. 2.2.1 of Bohigas et al. (2016)). The set of these possible configurations is known as the configuration space, or C-space, of the mechanism.

Velocity equation: For a given configuration $\mathbf{q}$ of the mechanism, it is possible to find a velocity equation

$$\mathbf{L}\,\mathbf{m} = \mathbf{0}, \tag{2}$$

which gives all possible motions of the manipulator at that specific configuration.

The velocity equation can be obtained in several ways, for example, by differentiating the assembly constraints, or using the tools of screw theory, or by other means (Zlatanov 1998). $\mathbf{L}$ is a matrix that depends on the configuration, and $\mathbf{m}$ is a velocity vector that globally characterizes the velocity state of the manipulator. In this vector, we can distinguish three different blocks. We have the output velocities that usually encode the velocity of a point, or the angular velocity or the twist of some body. The input velocities, which represent the velocities of the acutated joints, and the rest, which we call passive velocities:

$$\mathbf{m} = \begin{bmatrix} \mathbf{m}_u \\ \mathbf{m}_v \\ \mathbf{m}_p \end{bmatrix} \begin{matrix} \rightarrow output \\ \rightarrow input \\ \rightarrow passive \end{matrix}$$

Forward and inverse instantaneous kinematics problem: In general, the instantaneous kinematic analysis of a manipulator addresses two main problems:

- The *forward* instantaneous kinematics problem (FIKP): find $\mathbf{m}$ given the input velocity $\mathbf{m}_v$.
- The *inverse* instantaneous kinematics problem (IIKP): find $\mathbf{m}$ given the output velocity $\mathbf{m}_u$.

Note that in both cases, it is required to find *all* velocity components of $\mathbf{m}$, not just those referring to the output or input velocities. Following Zlatanov (1998), a configuration is said to be *nonsingular* when both the FIKP and the IIKP have unique solutions for any input or output velocity, and *singular* otherwise.

Forward and inverse singularities: Let $\mathbf{L}_y$, $\mathbf{L}_z$, and $\mathbf{L}_p$ be the submatrices of $\mathbf{L}$ obtained by removing the columns corresponding to the input, output, and both the input and output velocities, respectively. It is easy to see that the singular configurations are those in which either $\mathbf{L}_y$ or $\mathbf{L}_z$ is rank deficient (Bohigas et al. 2016).

Recall that if a matrix is rank deficient, its kernel has to be nonnull and, in particular, it must include a vector of unit norm. Thus, all singular configurations can be determined by solving the following two systems of equations:

$$\left.\begin{array}{c}\Phi(\mathbf{q})=\mathbf{0}\\ \mathbf{L}_y^{\mathsf{T}}\xi=\mathbf{0}\\ \|\xi\|^2=1\end{array}\right\},\quad \left.\begin{array}{c}\Phi(\mathbf{q})=\mathbf{0}\\ \mathbf{L}_z^{\mathsf{T}}\xi=\mathbf{0}\\ \|\xi\|^2=1\end{array}\right\}. \tag{3}$$

The first line of each system constrains $\mathbf{q}$ to be a feasible configuration of the mechanism, and the second and third lines enforce the existence of a nonzero vector in the kernel of the corresponding matrix. The solutions of the system on the left include all singularities where the FIKP is indeterminate, called *forward singularities*, while the solutions of the system on the right include all singularities where the IIKP is indeterminate, called *inverse singularities*.

All singularities of a mechanism can thus be obtained as the union of the solution sets of the systems of equations in Eq. (3).

1.2 An Illustrative Example

To illustrate the previous concepts on a very simple example, consider the three-slider mechanism in Fig. 1. Let (x_P, y_P) denote the coordinates of points $P = A$, B, or C relative to the reference frame OXY in the figure, and let L_1 and L_2 be the lengths of the connector links. Clearly, a configuration of the mechanism can be shortly described by the tuple $\mathbf{q} = (y_A, y_B, x_C)$, because $x_A = x_B = y_C = 0$ in any configuration. Since the distances from A to B and from B to C must be kept equal to L_1 and L_2, the assembly constraints for this mechanism can be written as

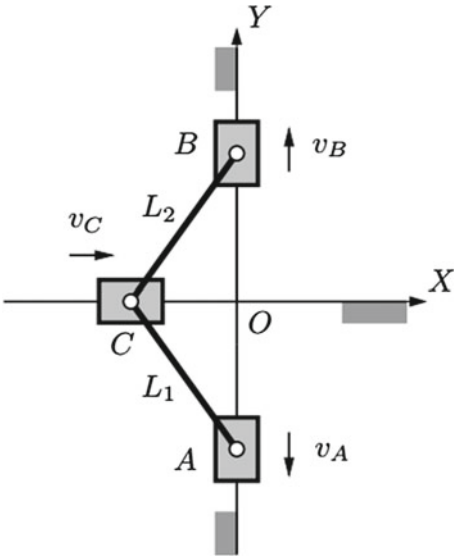

Fig. 1 A 1-DOF mechanism with three sliders. The prismatic joints at A and B are on a line perpendicular to the axis of the prismatic joint at C

$$\left.\begin{aligned} {y_A}^2 + {x_C}^2 &= {L_1}^2 \\ {y_B}^2 + {x_C}^2 &= {L_2}^2 \end{aligned}\right\}, \tag{4}$$

which corresponds to Eq. (1) for this mechanism. It is easy to realize that in this case the C-space corresponds to the intersection of two cylinders in the (y_A, y_B, x_C) space, as illustrated in Fig. 2.

The velocity equation in Eq. (2) could now be obtained using the revolute- and prismatic joint screws (Zlatanov 1998), but a more compact expression can here be derived by differentiating Eq. (4) with respect to time. Taking v_A and v_B as the input and output velocities, the differentiation yields

$$\mathbf{L} \cdot \mathbf{m} = \begin{bmatrix} 0 & 2y_A & 2x_C \\ 2y_B & 0 & 2x_C \end{bmatrix} \cdot \begin{bmatrix} v_B \\ v_A \\ v_C \end{bmatrix} = \mathbf{0}.$$

From here, we rapidly obtain

$$\mathbf{L}_y = \begin{bmatrix} 0 & 2x_C \\ 2y_B & 2x_C \end{bmatrix},$$

$$\mathbf{L}_z = \begin{bmatrix} 2y_A & 2x_C \\ 0 & 2x_C \end{bmatrix},$$

$$\mathbf{L}_p = \begin{bmatrix} 2x_C \\ 2x_C \end{bmatrix},$$

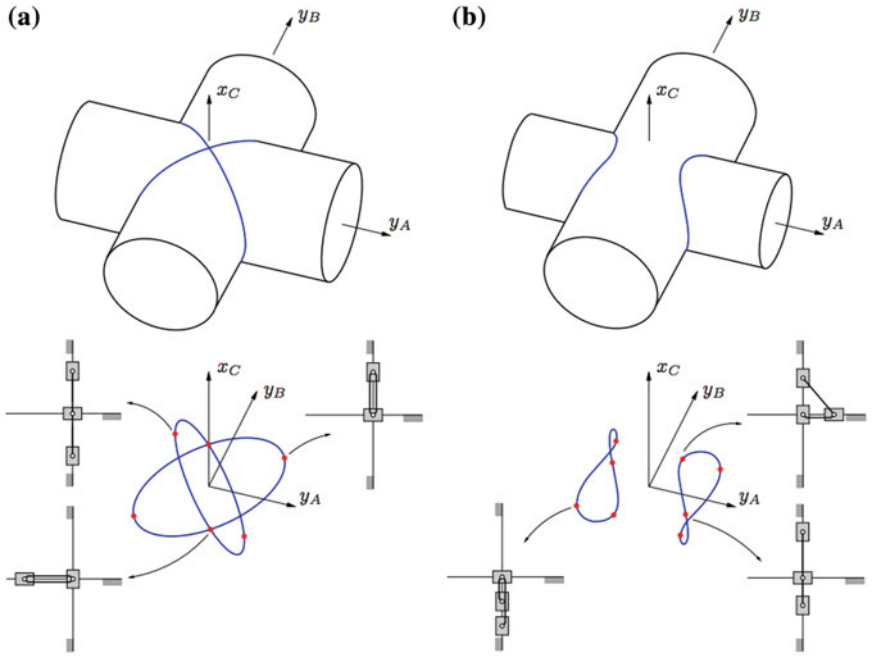

Fig. 2 C-space and singularities of the three-slider mechanism

which allow defining any of the systems in Eq. (3).

These systems can be solved analytically in this case. For example, if $L_1 = L_2 = 1$, the C-space has a single connected component composed of two ellipses intersecting on the x_C-axis (Fig. 2a), and the solutions of the systems reveal that the singularity set has six isolated configurations, marked in red in Fig. 2a, bottom, with the following values of $\mathbf{q}$:

$$\begin{gathered}(0,0,1),\ (0,0,-1),\\ (1,1,0),\ (1,-1,0),\\ (-1,-1,0),\ (-1,1,0).\end{gathered}$$

All of these configurations satisfy both systems in Eq. (3), so they are all forward and inverse singularities because both the FIKP and the IIKP are indeterminate on them. Therefore, the control of the input or the output velocity does not determine the overall motion of the mechanism. Note that the C-space self-intersects at two singularities and presents a bifurcation that allows to change the mode of operation from both sliders moving on the same side of the horizontal axis, $y_A y_B \geq 0$, to one slider moving on each side, $y_A y_B \leq 0$.

The topology of the C-space changes when $L_1 \neq L_2$, since it no longer presents any bifurcation. It is instead formed by two connected components (Fig. 2b). By

solving Eq. (3) for $L_1 = 1$ and $L_2 = 0.8$, eight singularities are obtained (Fig. 2b, bottom):

$$\begin{array}{rr} (1, 0.8, 0), & (-1, -0.8, 0), \\ (-1, 0.8, 0), & (0.6, 0, -0.8), \\ (1, -0.8, 0), & (-0.6, 0, 0.8), \\ (0.6, 0, 0.8), & (-0.6, 0, -0.8). \end{array}$$

In this case, to change the operation mode from $y_A \geq 0$ to $y_A \leq 0$ the mechanism has to be disassembled.

2 Singularity Set Computation with the CUIK Suite

The previous example was simple enough so that the systems of equations describing the singularities could be solved by hand, but this is usually not the case. This section shows how to use the CUIK suite to obtain the singularities of the Dextar, a five-bar parallel robot designed to support teaching and research, with a more interesting singularity set (Campos et al. 2010).

Despite its simple architecture, Dextar exhibits forward and inverse singularities, which requires careful analysis and planning to keep it under control and to exploit its full motion capabilities.

The section will guide you in the process of computing Dextar's singularities: from the formulation of the assembly constraints and singularity set equations, to their numerical solution, to the visualization and interpretation of the results. Along the way, we shall explain the syntax of the involved CUIK files and commands.

2.1 Preparation

For a better understanding, we suggest you to read Sects. 2.1 and 2.2 of Bohigas et al. (2016) and, ideally, also Sects. 3.1–3.4. You should have the CUIK suite installed in your computer. Refer to the instructions in the CUIK Project home page if you have not installed it yet.

During this example, we shall build few files incrementally. We recommend you to follow the instructions below, but you can download the finished files as well if you wish at Tutorial Solutions.

Throughout, we will call CUIK's commands from a UNIX/Linux terminal window. Occasionally, we shall give computation times. Take them as rough estimations. The actual time will depend on your machine. Ours was an iMac 2010 with a 2.93 GHz Intel Core i7 processor and 12 Gb of RAM, running Mac OSX "Yosemite". Unless where noted, we used just one of the eight CPUs of the machine.

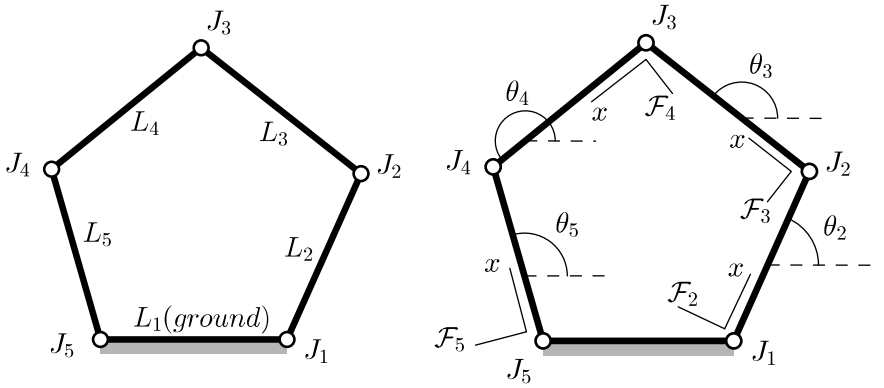

Fig. 3 Geometry of the Dextar robot. A reference frame F_i is attached to each link of the Dextar robot. Also, the orientation of each link is represented by angle θ_i

2.2 Dextar's Geometry

Dextar is a planar five-bar robot. It can be thought of as two-serial, three-revolute arms that share their distal joint. We start labeling its joints and links as $J_1, \ldots, J_5$ and $L_1, \ldots, L_5$, respectively (Fig. 3). We assume that l_i is the length of link L_i. The distance between J_1 and J_5 is d. We shall adopt the parameters of the Mecademic version of Dextar:

$$l_2 = l_3 = l_4 = l_5 = 90\,\text{mm}$$
$$d = 118\,\text{mm}$$

The actuated joints are J_1 and J_5. The end-effector is located in joint J_3, and only its x and y coordinates can be controlled.

For convenience, we assign a reference frame to each link L_i, indicated with F_i. We mark the x-axis of each frame with an "x". Frame F_1 is centered in J_5 and acts as the absolute frame. Frames $F_2, \ldots, F_5$ are centered and oriented as indicated in their respective links (Fig. 3).

We next see how to formulate Dextar's assembly constraints and the conditions characterizing its singularity locus. For simplicity, we shall derive the latter by taking the time derivative of Eq. (1). This will require us to include both the input and output coordinates in $\mathbf{q}$:

- The input coordinates will be θ_2 and θ_5.
- The output coordinates will be J_{3x} and J_{3y}.

2.3 Assembly Constraints

Equation (1) has to be formulated in a form that allows it to be converted into quadratic equation. There are basically two ways of achieving so: using joint-assembly constraints or using loop-closure constraints. We next explain both.

Using joint-assembly constraints: Under this method, we formulate Eq. (1) by gathering Eq. 3.1 in Bohigas et al. (2016):

$$\mathbf{r}_j + \mathbf{R}_j \mathbf{p}_i^{F_j} = \mathbf{r}_k + \mathbf{R}_k \mathbf{p}_i^{F_k}$$

for all joints J_i of the mechanism, which forces the assembly of links L_j and L_k at J_i (see Sect. 3.2.1 of Bohigas et al. (2016)), and:

- $\mathbf{r}_j$ is the absolute position vector of the origin frame F_j.
- $\mathbf{R}_j$ is the rotation matrix giving the orientation of F_j.
- $\mathbf{p}_i^{F_j}$ is the position vector of J_i in frame F_j, written in the vector basis of F_j.

Since this is a planar linkage, Eq. 3.2 in Bohigas et al. (2016) is not needed in this case. If we write the equation for each joint J_i, we obtain

$$\begin{bmatrix} d \\ 0 \end{bmatrix} = \begin{bmatrix} J_{1x} \\ J_{1y} \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} J_{1x} \\ J_{1y} \end{bmatrix} + \mathbf{R}_2 \begin{bmatrix} l_2 \\ 0 \end{bmatrix} = \begin{bmatrix} J_{2x} \\ J_{2y} \end{bmatrix} + \mathbf{R}_3 \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} J_{2x} \\ J_{2y} \end{bmatrix} + \mathbf{R}_3 \begin{bmatrix} l_3 \\ 0 \end{bmatrix} = \begin{bmatrix} J_{3x} \\ J_{3y} \end{bmatrix} + \mathbf{R}_4 \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} J_{3x} \\ J_{3y} \end{bmatrix} + \mathbf{R}_4 \begin{bmatrix} l_4 \\ 0 \end{bmatrix} = \begin{bmatrix} J_{5x} \\ J_{5y} \end{bmatrix} + \mathbf{R}_5 \begin{bmatrix} l_5 \\ 0 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} J_{5x} \\ J_{5y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9}$$

which altogether form the system of Eq. (1). Note that, using these equations,

$$\mathbf{q} = (J_{3x}, J_{3y}, J_{1x}, J_{1y}, J_{2x}, J_{2y}, J_{5x}, J_{5y}, \theta_3, \theta_4, \theta_2, \theta_5),$$

where $\mathbf{v} = (\theta_2, \theta_5)$ and $\mathbf{u} = (J_{3x}, J_{3y})$, encompasses the input and output coordinates of the mechanism, respectively.

Since

$$\mathbf{R}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & cos\theta_i \end{bmatrix},$$

Equations (5)–(9) are not quadratic and, thus, not directly solvable using CUIK. However, we can apply the change of variables

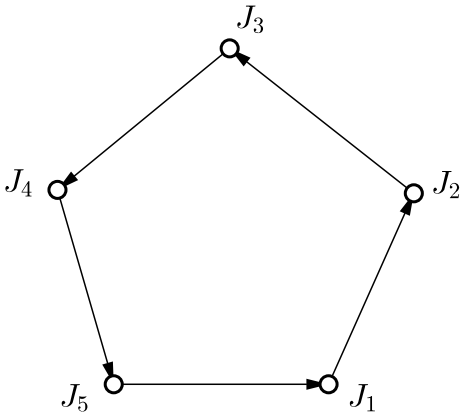

Fig. 4 Dextar robot has only one kinematic loop

$$c_i = \cos\theta_i$$
$$s_i = \sin\theta_i c$$

and include the circle equations

$$s_i^2 + c_i^2 = 1 \tag{10}$$

to readily obtain a quadratic system solvable by the CUIK suite.

This is a common technique in any system to be solved by CUIK. We first formulate the systems without worrying about the appearance of sines and cosines, and finally algebrize the systems with the mentioned change of variables.

In spatial mechanisms, we can directly formulate the systems in algebraic form following Sect. 3.2 in Bohigas et al. (2016).

Using loop-closure constraints: The previous method is simple, but sometimes it leads to a system $\Phi(\mathbf{q}) = \mathbf{0}$ with too many variables and equations. More compact systems of equations can often be obtained using loop-closure constraints and graph theory tools.

Dextar has only one kinematic loop. Forcing the assembly of this loop is equivalent to constraining the inter-joint vectors to form a closed polygon (Fig. 4). This condition can be expressed with the equations

$$d + l_2\cos\theta_2 + l_3\cos\theta_3 + l_4\cos\theta_4 - l_5\cos\theta_5 = 0 \tag{11}$$
$$l_2\sin\theta_2 + l_3\sin\theta_3 + l_4\sin\theta_4 - l_5\sin\theta_5 = 0 \tag{12}$$

which directly characterize the C-space of the robot.

Note that Eqs. (11) and (12) contain the input coordinates θ_2 and θ_5 as variables, but not the output coordinates J_{3x} and J_{3y}. To introduce the latter, we can simply add the equations

$$J_{3x} = d + l_2 \cos\theta_2 + l_3 \cos\theta_3 \tag{13}$$
$$J_{3y} = l_2 \sin\theta_2 + l_3 \sin\theta_3. \tag{14}$$

Then, Eq. (1) is the system formed by Eqs. (11)–(14), and the configuration vector of the system is

$$\mathbf{q} = (J_{3x}, J_{3y}, \theta_3, \theta_4, \theta_2, \theta_5),$$

again with $\mathbf{v} = (\theta_2, \theta_5)$ and $\mathbf{u} = (J_{3x}, J_{3y})$ encompassing the input and output coordinates of the mechanism, respectively.

Incidentally, note that Eqs. (11) and (12) can be obtained by adding Eqs. (5)–(9). This is true in a general mechanism. The loop-closure constraint of any kinematic loop can be obtained by adding the joint-assembly constraints of the loop joints. If our robot has multiple loops, Eq. (1) will include the loop-closure equations of a set of fundamental cycles of its connectivity graph. For details, see Porta et al. (2009), which also shows that the two formulations (via joint-assembly or loop-closure constraints) are equivalent.

Which formulation is better? Since Eqs. (11)–(14) involve less equations and variables than Eqs. (5)–(9), we shall prefer them to formulate the systems of the various singularity sets of Dextar.

Note however that, in other cases, there might be no gain in using loop-closure equations. For example, the general formulation of Bohigas et al. (2016) uses screw theory to formulate the velocity equation (Sect. 3.2.2) which requires the joints P_i to be available (see Table 3.1 of Bohigas et al. (2016) and the corresponding text). In such a case, it might be simpler to use joint-assembly constraints, which make all P_i points explicit and ready-to-use in the system.

2.4 Inverse Singularity Conditions

Inverse singularities occur when the mechanism is found to be shaky after locking its outputs. In Dextar, this happens when the left or right arms happen to be aligned. For example, if the left arm is aligned, we can feel the shakiness shown in Fig. 5.

Recall that such singularities are the configurations $\mathbf{q}$ that satisfy the right system in Eq. (3), where $\mathbf{L_z} = \Phi_\mathbf{z}(\mathbf{q})$ as shown in Sect. 2.2.1 of Bohigas et al. (2016). Since in our case $\Phi(\mathbf{q}) = \mathbf{0}$ is defined by Eqs. (11)–(14), and

$$\mathbf{z} = (\theta_3, \theta_4, \theta_2, \theta_5),$$

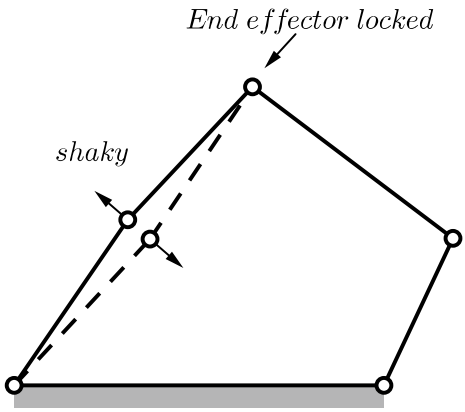

Fig. 5 An inverse singularity happens when the left (or right) arm of the Dextar is aligned

we have

$$\Phi_{\mathbf{z}}(\mathbf{q}) = \begin{bmatrix} -l_3 \sin\theta_3 & -l_4 \sin\theta_4 & -l_2 \sin\theta_2 & l_5 \sin\theta_5 \\ l_3 \cos\theta_3 & l_4 \cos\theta_4 & l_2 \cos\theta_2 & -l_5 \cos\theta_5 \\ -l_3 \sin\theta_3 & 0 & -l_2 \sin\theta_2 & 0 \\ l_3 \cos\theta_3 & 0 & l_2 \cos\theta_2 & 0 \end{bmatrix}. \tag{15}$$

The rows of $\Phi_{\mathbf{z}}(\mathbf{q})$ correspond to the partial derivatives of Eqs. (11)–(14) with respect to the $\mathbf{z}$ variables.

Again, the resulting system of equations is not quadratic, but it can be converted into a quadratic one by replacing the sines and cosines of θ_i by s_i and c_i, and adding the corresponding circle equations.

2.5 Forward Singularity Conditions

The forward singularities are the configurations in which the mechanism is shaky after locking the actuators. In Dextar, this happens when J_2, J_3, and J_4 happen to be aligned, as shown in Fig. 6.

These are the configurations that satisfy the left system in Eq. (3), with $\mathbf{L}_{\mathbf{y}} = \Phi_{\mathbf{y}}(\mathbf{q})$. In our case

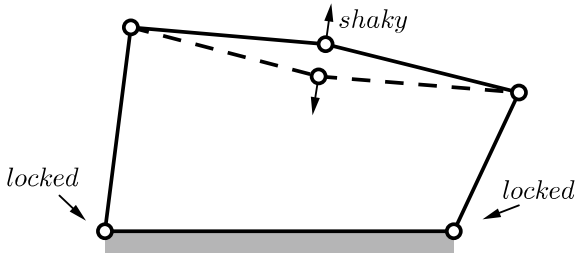

Fig. 6 A forward singularity happens when J_2, J_3, and J_4 are aligned

$$\Phi_{\mathbf{y}}(\mathbf{q}) = \begin{bmatrix} 0 & 0 & -l_3 \sin\theta_3 & -l_4 \sin\theta_4 \\ 0 & 0 & l_3 \cos\theta_3 & l_4 \cos\theta_4 \\ -1 & 0 & -l_3 \sin\theta_3 & 0 \\ 0 & -1 & l_3 \cos\theta_3 & 0 \end{bmatrix}, \tag{16}$$

where the rows correspond to the partial derivatives of Eqs. (11)–(14) with respect to the $\mathbf{y}$ variables

$$\mathbf{y} = (J_{3x}, J_{3y}, \theta_3, \theta_4).$$

2.6 Computation and Output Plots

We next see how to prepare the CUIK input files associated with the previous systems of equations and how to solve them using CUIK. We shall compute, in sequence, the following sets:

- The full C-space;
- The inverse singularity locus;
- The forward singularity locus.

These correspond to the sets indicated on the top of Fig. 7. Strictly speaking, the silhouettes of the C-space relative to the input and output spaces are called the input and output singularity loci. This is because these silhouettes are only defined for regular points of the C-space. However, Dextar does not have C-space singularities, also known as increased instantaneous mobility singularities (Zlatanov 1998). Thus, its inverse and forward singularity loci coincide with its output and input singularity sets, respectively.

Once computed, the three sets will be projected to the input and output spaces, producing what we call the input and output portraits of the mechanism (see Sect. 3.4 of Bohigas et al. (2016)). These portraits are also indicated in the bottom of Fig. 7.

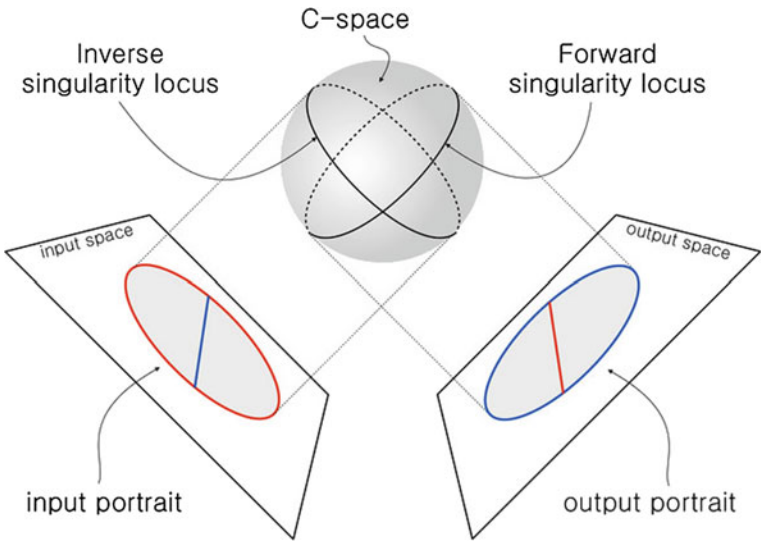

Fig. 7 Singularities seen as C-space silhouettes (in an abstract C-space)

Computing the configuration space: Open a new file with a text editor, and save it with the name `cspace.cuik`. Any cuik file has three sections which encompass, respectively, definitions of constants used in the equations, the list of equation variables, and the equations themselves:

```
[CONSTANTS]
...
[SYSTEM VARS]
...
[SYSTEM EQS]
...
```

In `cspace.cuik` type

```
[CONSTANTS]
l := 0.90
d := 1.18
l2 := l
l3 := l
l4 := l
l5 := l
```

These constants define the five link lengths. Although all moving links are of the same length in Dextar, we prefer to define the link lengths independently using "$l_i = value$" because this will allow us to test different geometric parameters.

Now let us define the variables in the system. Type in

```
[SYSTEM VARS]
J3x:[-l4-l4,l4+l5]
J3y:[-l4-l4,l4+l5]
```

The syntax is clear. A variable is created by specifying its name, followed by ":" and its feasibility interval. Since the position of joint J_3 is constrained to lie inside a circle of radius $l_4 + l_5$ centered in the origin of the absolute frame, J_{3x} and J_{3y} can be given the previous feasibility intervals.

We also define the sines and cosines of all angles $\theta_2, \ldots, \theta_5$, which take values in the range $[-1, 1]$:

```
ct2:[-1,1]
st2:[-1,1]
ct3:[-1,1]
st3:[-1,1]
ct4:[-1,1]
st4:[-1,1]
ct5:[-1,1]
st5:[-1,1]
```

Finally, we add Eqs. (11)–(14) and the circle equations relating the cti and sti variables:

```
[SYSTEM EQS]

% Loop equations
d + l2*ct2 + l3*ct3 + l4*ct4 - l5*ct5 = 0;
l2*st2 + l3*st3 + l4*st4 - l5*st5 = 0;

% Position of J3
J3x = d + l2*ct2 + l3*ct3;
J3y = l2*st2 + l3*st3;

% Circle equations due to angle algebraizations
ct2^2 + st2^2 = 1;
```

```
ct3^2 + st3^2 = 1;
ct4^2 + st4^2 = 1;
ct5^2 + st5^2 = 1;
```

Pay attention at the syntax employed to define sums, products, and the square of a variable, and note that all equations must be finished with a semicolon. Also observe that a percentage sign "%" can be used to insert a comment. It can be placed anywhere, and the result is that any text to the right of "%" will be neglected.

Now copy the following parameters into a new file `cspace.param`:

```
SIGMA = 0.2
EPSILON = 1e-6
RHO = 0.95
ERROR_SPLIT = TRUE
DUMMIFY = 1
LR2TM_Q =0.5
LR2TM_S = 0.1
MAX_NEWTON_ITERATIONS = 0
```

From the point of view of a user, the most important parameter is SIGMA. It defines the resolution at which the solution set will be computed. The smaller the sigma, the higher the resolution. SIGMA, in fact, is the maximum edge length allowed to any of the solution boxes returned.

Now we are ready to launch the first computation. Execute

```
cuik cspace
```

This will throw a bunch of numbers showing CUIK's progress on solving the equations. Their meaning is not much relevant to an end user.

Wait until the computation finishes (about 45 s in our case). Once it does, you will see that CUIK has created the file `cspace.sol`, which contains, in order:

- The numerical parameters used (SIGMA, RHO, and so on);
- The solved system of equations;
- A list of the solution boxes returned;
- Execution statistics.

To see the latter statistics, type

```
tail -30 cspace.cuik
```

We can inspect the `*.sol` file with a text editor if we wish, but a more useful choice is to obtain a plot of the solution boxes. Since there are ten variables defined in `cpsace.cuik`, the boxes returned will be ten dimensions, and we can only observe their projections onto three of the ten coordinates.

To obtain such projections, use cuikplot3d to convert `cspace.sol` to a file `cspace.gcl`:

```
cuikplot3d cspace 1 2 3 0 cpsace.gcl
```

The extension gcl stands for "Geomview Command Language", and the created `*.gcl` file contains all graphic commands needed to visualize the boxes using Geomview. In the previous command:

- `cspace` is the name of the input `*.sol` file.
- 1, 2, 3 are the dimensions onto which we project the solution boxes: 1 means the first variable that appears in `cspace.cuik`, 2 means the second variable, and so on (in our case, these three variables are J3x, J3y, and ct2).
- 0 indicates that no artificial enlarging of the box is to be applied. If you use a number different from zero here, then the output boxes will have their edge lengths equal to this number. This is useful to visualize boxes that are too tiny in comparison to other boxes.
- `cspace.gcl` is the desired output file.

Now we can readily view the boxes with Geomview. Type

```
geomview cspace.gcl
```

We can now obtain the views of the configuration space in Fig. 8, top. Nicer plots can be obtained by using a smaller SIGMA value. For example, using SIGMA $=$ 0.05 (and waiting about 10 min), we get the refined plots in Fig. 8, bottom.

Observe that the first plot provides Dextar's workspace because it is the projection onto the J_{3x} and J_{3y} coordinates. The computation of the whole C-space, therefore, provides a simple way of obtaining workspace maps of any of the mechanism coordinates. In many cases, however, we shall prefer to obtain the workspace by finding a generalized singularity set, as explained in Chap. 4 of Bohigas et al. (2016). Such a procedure is advantageous as it yields the boundary and interior barriers of the workspace, and it also takes joint limits into account. It is often faster too because the boundaries and barriers are, generically, of lower dimension.

Computing the inverse singularity locus: Let us prepare the `*.cuik` file of the inverse singularity locus. To do so, we shall extend `cspace.cuik` with new equations forcing the rank deficiency of $\Phi_{\mathbf{z}}(\mathbf{q})$.

First copy `cspace.cuik` to `inv.cuik` and `cspace.param` to `inv.param`:

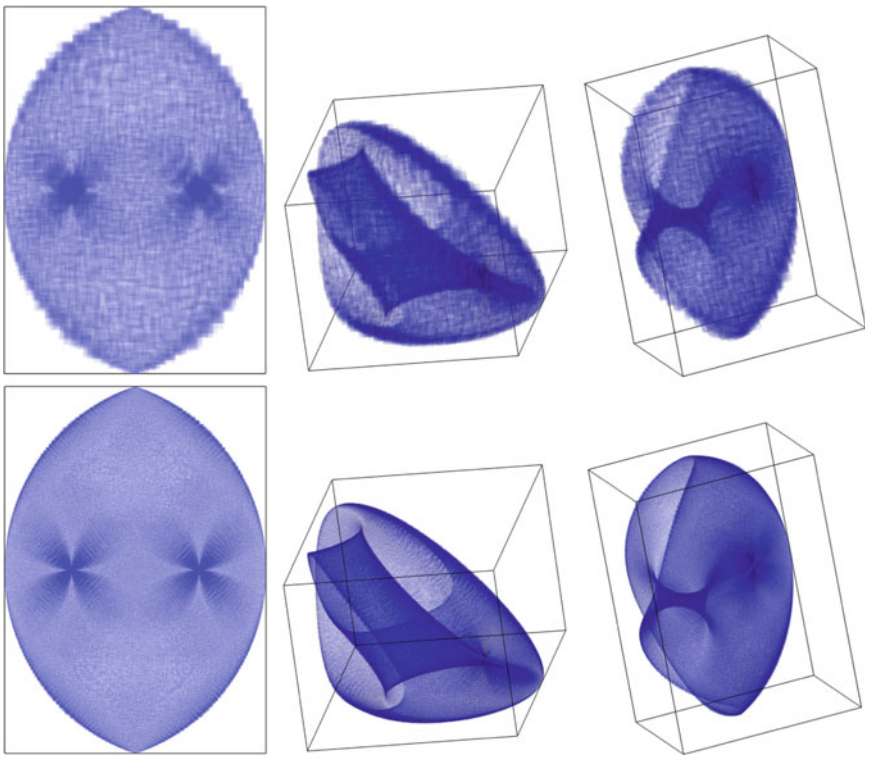

Fig. 8 Three different views of the C-space of the Dextar robot, using a different value for SIGMA (left and right)

```
cp cspace.cuik inv.cuik
cp cspace.param inv.param
```

and open `inv.cuik` with a text editor. Add the following variables to its `[SYSTEM VARS]` section:

```
[SYSTEM VARS]
...
xi1:[-1,1]
xi2:[-1,1]
xi3:[-1,1]
xi4:[-1,1]
```

These xi are the components of the ξ vector of Eq. (3). Now add the rank deficiency equations to the `[SYSTEM EQS]` section:

```
[SYSTEM EQS]
...
% Inverse singularity conditions
-l3*st3*xi1 - l4*st4*xi2 - l2*st2*xi3 + l5*st5*xi4 = 0;
 l3*ct3*xi1 + l4*ct4*xi2 + l2*ct2*xi3 - l5*ct5*xi4 = 0;
-l3*st3*xi1 - l2*st2*xi3 = 0;
 l3*ct3*xi1 + l2*ct2*xi3 = 0;

xi1^2 + xi2^2 + xi3^2 + xi4^2 = 1;
```

Note that when $\Phi_{\mathbf{z}}(\mathbf{q})$ is rank deficient, there will be at least two vectors ξ satisfying these equations. To avoid computing each singularity point twice, we can add the equation of a random half-plane through the origin, which will discard one of the points almost always. For example, let us add

```
0.5795*xi1 + 0.5683*xi2 + 0.3000*xi3 + 0.5013*xi4 >=0;
```

to the `[SYSTEM EQS]` section. Although adding this inequality is optional, it drastically reduces the computation time.

Now solve the equations

```
cuik inv
```

and plot the solutions in the (J3x, J3y, ct2) space:

```
cuikplot3d inv 1 2 3 0 inv.gcl
geomview inv.gcl
```

Using Geomview's controls, it is not difficult to obtain the plots of the inverse locus in Fig. 9, obtained for different values of SIGMA. Observe how, as expected, the plots delimit the workspace of Dextar in the (J3x, J3y) space.

The inverse locus is formed by two circular arcs and two isolated points. The arcs correspond to configurations in which the left or right arms are fully extended (Fig. 10, left), and the isolated points correspond to those in which the arms are folded back, with J_3 coinciding with J_2 or J_5 (Fig. 10, middle).

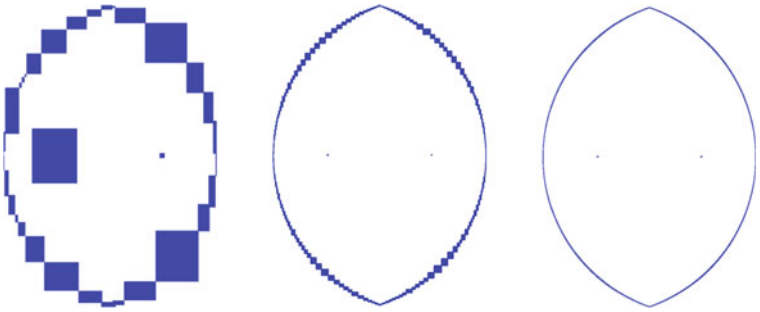

Fig. 9 Inverse singularity set of the Dextar robot computed at different SIGMA values. From left to right (in parentheses, the computation times needed in seconds): 1 (31), 0.1 (49), 0.01 (158)

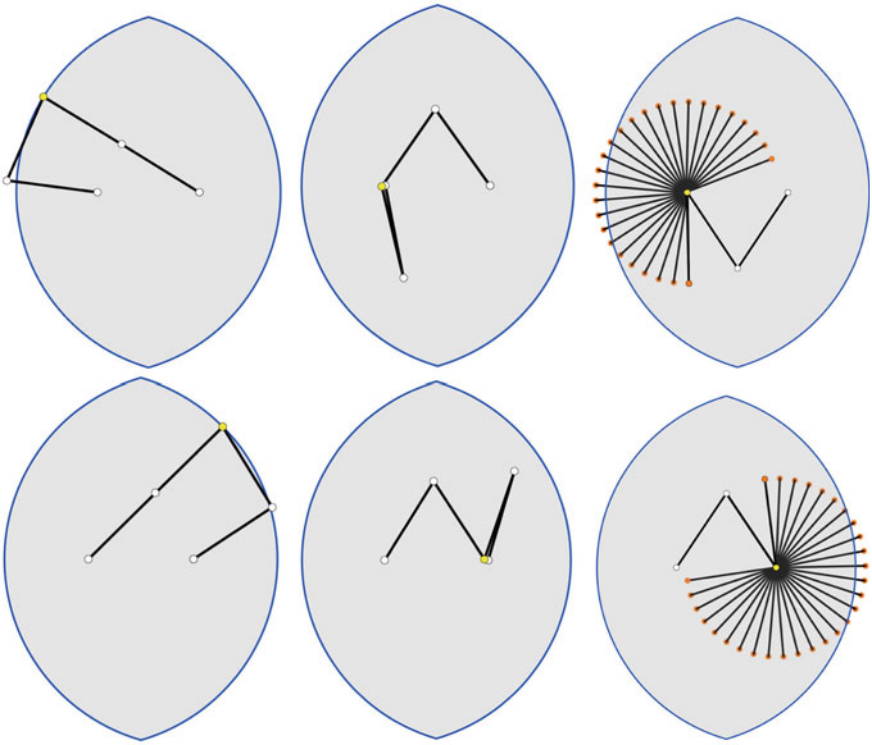

Fig. 10 Left: two configurations with the end-effector at the workspace boundary (right or left arm fully extended). Middle: two configurations with the end-effector at an interior barrier (left or right arm folded back). Right: there is a continuum of configurations for which the end-effector coincides with J_2 or J_5

In fact, the isolated points are curves in configuration space, as there is a one-dimensional continuum of configurations in which J_3 coincides with J_2 or J_5 (Fig. 10, right).

Computing the forward singularity locus: Let us now prepare the `*.cuik` file of the forward singularity locus. We shall extend `cspace.cuik` with new equations forcing the rank deficiency of $\Phi_{\mathbf{y}}(\mathbf{q})$.

First copy `cspace.cuik` to `fwd.cuik` and `cspace.param` to `fwd.param`:

```
cp cspace.cuik fwd.cuik
cp cspace.param fwd.param
```

Then, add the following variables to the `[SYSTEM VARS]` section of `fwd.cuik`:

```
[SYSTEM VARS]
...
xi1:[-1,1]
xi2:[-1,1]
xi3:[-1,1]
xi4:[-1,1]
```

Finally, add the rank deficiency equations to the `[SYSTEM EQS]` section:

```
[SYSTEM EQS]
...
% Forward singularity conditions
-l3*st3*xi3 - l4*st4*xi4 = 0;
 l3*ct3*xi3 + l4*ct4*xi4 = 0;
-xi1 - l3*st3*xi3 = 0;
-xi2 + l3*ct3*xi3 = 0;
 xi1^2 + xi2^2 + xi3^2 + xi4^2 = 1;

% Equation to select just one xi vector for each
singularity
0.0249*xi1 + 0.5923*xi2 + 0.6515*xi3 + 0.4734*xi4 >= 0;
```

Now, save `fwd.cuik` and start the computation of the forward singularity locus:

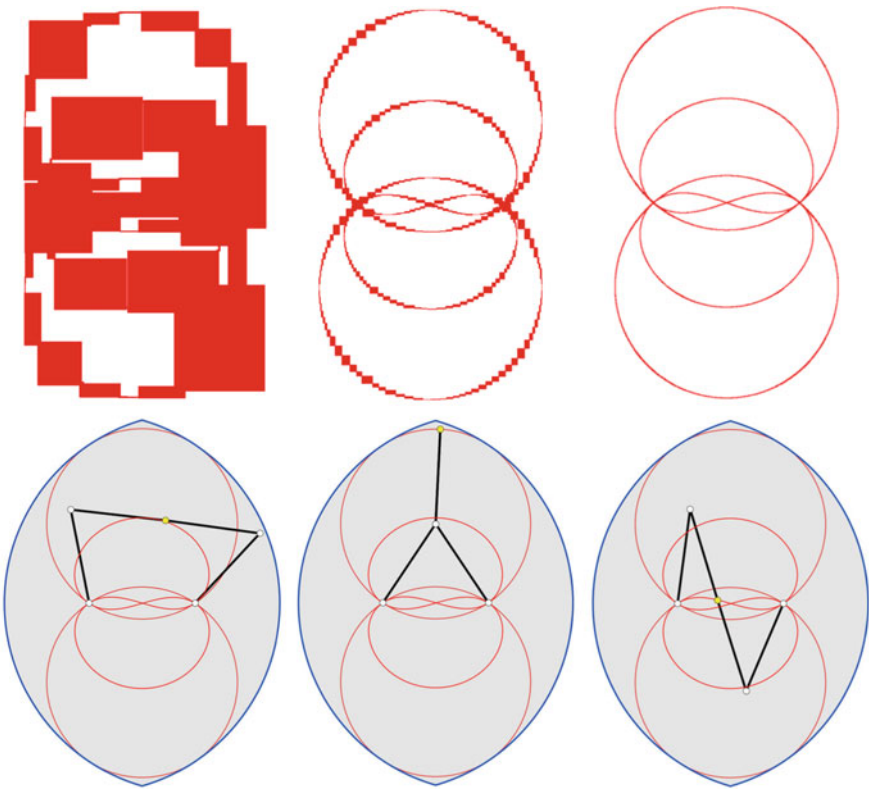

Fig. 11 Top: forward singularity set of the Dextar robot computed at different SIGMA values. From left to right (in parentheses, the computation times needed in seconds): 1 (3), 0.1 (11), 0.01 (69). Bottom: three examples of forward singularities of the Dextar robot

```
cuik fwd
```

When the computation has finished, execute

```
cuikplot3d fwd 1 2 3 0 fwd.gcl
geomview fwd.gcl
```

and use Geomview to obtain the plots in Fig. 11, top.

Each point of the red curve corresponds to a configuration of the mechanism in which J_2, J_3, and J_4 are aligned. The configurations in Fig. 11, bottom, for example, are forward singularities.

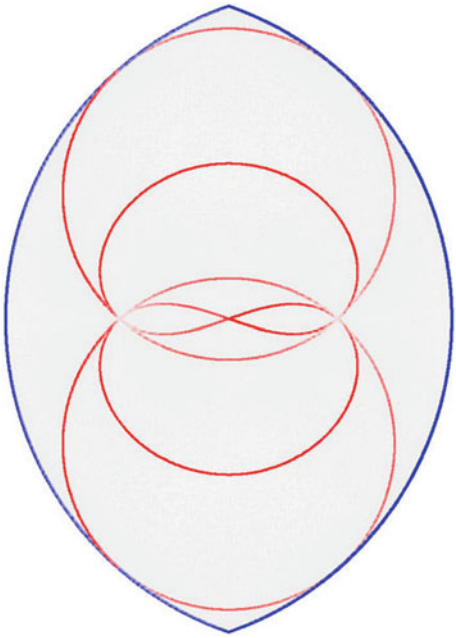

Fig. 12 Output portrait of the Dextar robot

2.7 *Visualizing the Output Portrait*

The previous examples show Dextar overlaid onto its output portrait (the projection of the C-space and the forward and inverse loci onto the output coordinates). To generate this portrait, run

```
geomview
```

and then load the C-space, inverse and forward singularity loci:

- File → Open, select `cspace.gcl` + click OK;
- File → Open, select `inv.gcl` + click OK;
- File → Open, select `fwd.gcl` + click OK.

By tuning a bit Geomview's output, you will obtain the image in Fig. 12.
To obtain this image, follow these hints after loading the three gcl files:

- Use ortographic projection:
  - Inspect → camera → ortographic
- Center the objects in your window:
  - Select World in "Targets"

  - Use "Zoom" and "Translate"
- Choose a white background:
  - Inspect → camera → background color
- Color the C-space boxes in gray and make them translucent:
  - Select the cspace in "targets"
  - Inspect → appearance and set Shading to "Constant". Also click on [Cf] to select a light gray color for the boxes
  - Inspect → material and click Transparent. Move the Alpha slider to a small value.
- Color the edges of the forward singularity boxes in red:
  - Inspect → appearance → Ce → select red. Also make the edges thicker by setting Line Width to 3 or 4 in the same panel.
- Color the edges of the inverse singularity boxes in blue:
  - Inspect → appearance → Ce → select blue. Also make edges thicker

2.8 Visualizing the Input Portrait

A great advantage of computing the singularity sets on the C-space is that they can be projected to any coordinate space. We next show how to project them to the active joint space (θ_2, θ_5).

The boxes in our `*.sol` files only contain the sine and cosine intervals of θ_2 and θ_5, but we can use

```
cuikatan2
```

to replace them by the corresponding angular intervals. Let us do that for `cspace.sol`. We run

```
cuikatan2 cspace 4 3 10 9 cspace_angles
```

since in cspace.cuik st2 and ct2 are declared as the fourth and third variables, and st5 and ct5 are declared as the tenth and ninth variables.

For each box in `cspace.sol`, this instruction substitutes the intervals appearing in the mentioned positions (4, 3, 10, and 9) by the corresponding angle intervals. The new angle intervals are added at the end of the box. The original file `cspace.cuik` is left unaltered, and the results are saved in `cspace_angles.sol`.

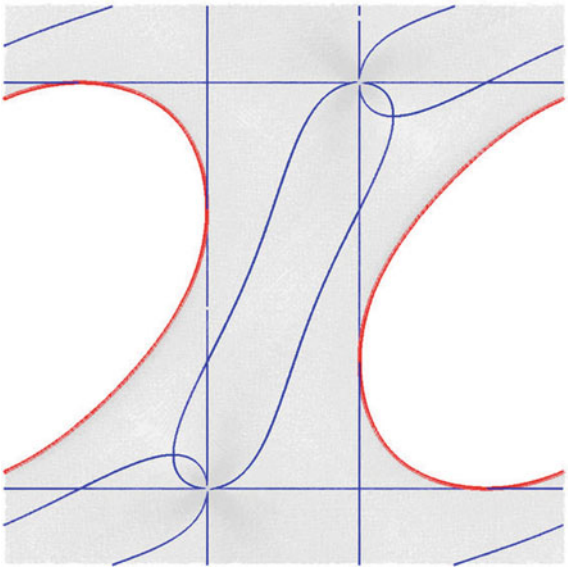

Fig. 13 Input portrait of the Dextar robot

Since each box in `cspace.sol` has 10 intervals, we will have 8 intervals in each box of `cspace_angles.sol` (cuikatan2 will have replaced 4 sine–cosine intervals by two angular intervals). The intervals relative to θ_2 and θ_5 will be in the last two positions in `cspace_angles.sol`. Thus, we can run

```
cuikplot3d cspace_angles 8 7 1 0 cspace_angles.gcl
```

to generate the `*.gcl` file corresponding to `cspace_angles.sol`.

To get analogous files for the forward and inverse singularity loci, we run

```
cuikatan inv 4 3 10 9 inv_angles
cuikatan fwd 4 3 10 9 fwd_angles
```

Finally, using Geomview on the created `*.gcl` files, we obtain the input portrait in the active joints space shown in Fig. 13:

```
cuikplot3d inv_angles 12 11 1 0 inv_angles.gcl
cuikplot3d fwd_angles 12 11 1 0 fwd_angles.gcl
```

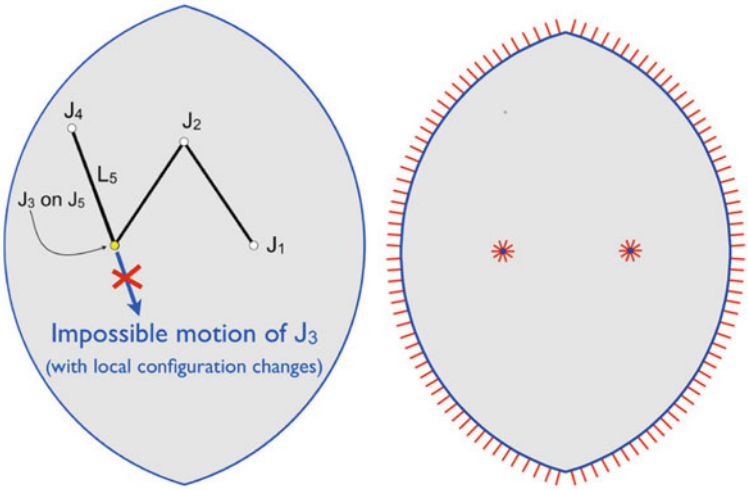

Fig. 14 Left: the shown configuration is an inverse singularity. The end-effector (J_3), coinciding with J_5, cannot move in the direction of link L_5. Right: workspace of Dextar relative to the output coordinates. In the plot, the inverse singularities in blue delimit the workspace boundaries, and small red vectors are shown on the forbidden direction of motion

The horizontal and vertical axes correspond to the θ_5 and θ_2 angles, respectively. The gray area spans the $[-\pi, \pi]$ range in both directions.

The input portrait provides the workspace of the (θ_2, θ_5) angles, which is also useful to a designer. For example, from the previous plot we see that, despite the existence of inner voids, the two angles can take any value in $[-\pi, \pi]$. This implies that Dextar should be equipped with multiturn actuators to exploit its full motion capabilities.

Note also how the roles of the forward and inverse singularity loci are reversed here. While the inverse locus delimited the attainable region in the task space, such region is bounded by the forward locus in the active joint space. However, it is shown in Bohigas et al. (2016) that not all points of the inverse (resp. forward) locus will necessarily project to the boundary of the workspace (resp. active joint) space. The isolated points of the inverse locus of Dextar, explained next, provide one example.

2.9 Workspace Boundaries and Interior Barriers

The inverse singularities help to delimit the workspace boundaries relative to the output coordinates, in this case (J_{3x}, J_{3y}). But they can also reveal the existence of interior barriers in the workspace. The isolated blue points that coincide with J_1 and J_5 are an example of such barriers: e.g., if we place J_3 on J_5, J_3 cannot move in the direction of link L_5 (using small variations of the input angles). In other words, there

is a motion barrier orthogonal to such direction (Fig. 14, left). Using the techniques of Chap. 4 in Bohigas et al. (2016), we would mark the workspace boundaries and barriers as shown in Fig. 14, right.

Note that our initial workspace map that resulted from projecting the C-space (Fig. 8, left) does not reveal the motion barriers interior to the workspace. The same can be said of any map obtained by the usual method of discretization (sweeping the task space points and checking whether the mechanism can be assembled in them, using inverse kinematics). Such a method will provide a good approximation of the workspace, but will fail to accurately detect the singularities and barriers present in its interior.

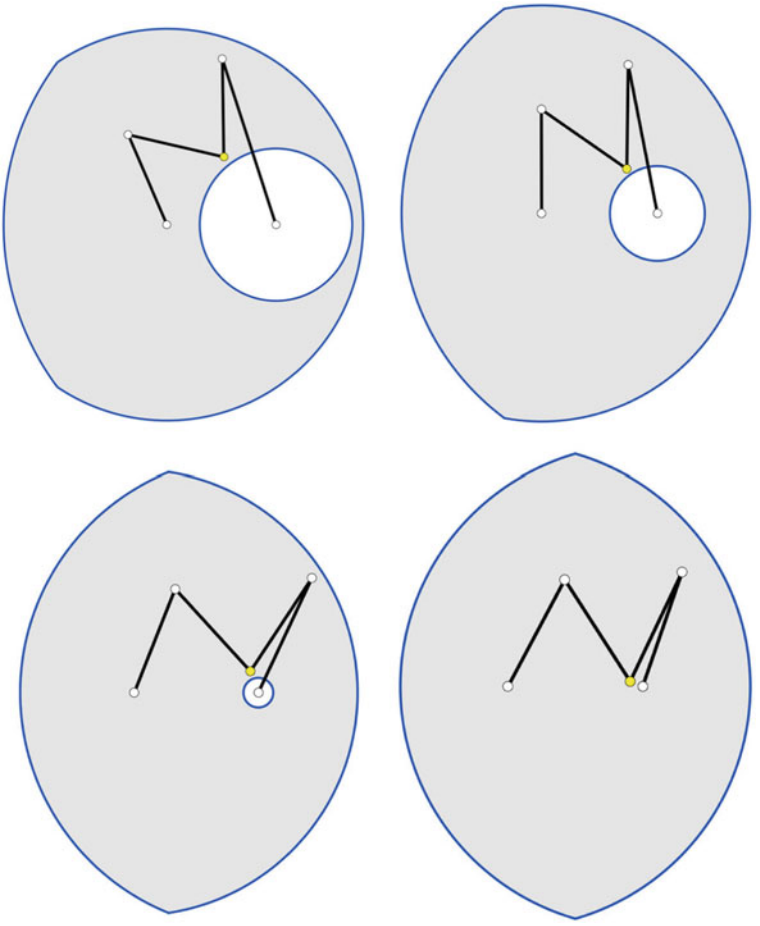

Fig. 15 From left to right and top to bottom, the value of l decreases from 0.7 to zero

The previous punctual barriers can be better understood when seen as the limit case of a family of workspaces. For example, if we consider the family of single-loop five-bar mechanisms with these parameters:

$$l_3 = l_4 = l_5 = 0.9\ ,\ \ l_2 = l + 0.9$$

and make l evolve from 0.7 to zero, we obtain the workspaces in Fig. 15. It is clear that the circular boundary of the workspace degenerates into an interior barrier with just one point when $l = 0$.

References

O. Bohigas, M. Manubens, L. Ros. *Singularities of Robot Mechanisms: Numerical Computation and Avoidance Path Planning* (Springer, Cham, 2016)

L. Campos, F. Bourbonnais, I. Bonev, P. Bigras, Development of a five-bar parallel robot with large workspace, in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE* (Montreal, Quebec, 2010)

Geomview. http://www.geomview.org/

Kinematics and robot design group at the Institut de Robòtica i Informàtica Industrial. http://www.iri.upc.edu/research/kinematics

Mecademic version of Dextar. http://www.mecademic.com

J.M. Porta, L. Ros, F. Thomas, A linear relaxation technique for the position analysis of multi-loop linkages. IEEE Trans. Robot. **25**(2), 225–239 (2009)

J.M. Porta, L. Ros, O. Bohigas, M. Manubens, C. Rosales, L. Jaillet, The CUIK suite: motion analysis of closed-chain multibody systems. IEEE Robot. Autom. Mag. **21**(3), 105–114 (2014)

The CUIK Project Home Page. http://www.iri.upc.edu/cuik

Tutorial Solutions. www.iri.upc.edu/people/ros/srm/Dextar-tutorial.zip

D. Zlatanov, Generalized Singularity Analysis of Mechanisms. Ph.D. thesis, University of Toronto, 1998