OPKID: Optimal Kinematic Design of Robots
Modelling and simulation of a 6 DOF articulated industrial robot in Delmia

Date: Thu 23:59, 16.12.16

Prof. P. Wenger, Prof. S. Caro, Prof. D. Chablat

Rabbia Asghar, BEng, Ernest Skrzypczyk, BSc

Contents

List of Figures

Introduction

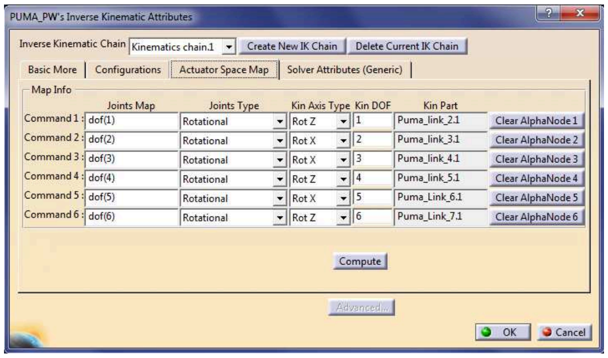
The objective of the practical work was the modelling and simulation of 6 degrees of freedom PUMA robot in Delmia and its setup in a robotic cell. Most of the steps involved in the process are already very well defined in the tutorial and are listed as follows:

- creation of kinematic joints and setting the joint limits,
- definition of direct and inverse kinematic models,
- creation of a gripper for the robot and definition of its open and close positions,
- creation of a robotic cell and attachment of gripper on robot,
- creation of a pick and place task,
- optimal placement of the robot for the task.

The more relevant steps and aspects of interest are explained in detail.

Modelling and simulation

Direct kinematic model of the robot is defined by creation of kinematic joints. Using the jog mechanism, the robot can then be manipulated in joint space or Cartesian space when IKM is present, where individual joint angles or coordinates respectively can be varied. Joint limits were assigned, such that the moving parts did not collide and did not go out of the base limits. In order to define the inverse kinematic model, frames of interest for end-effector and base were defined. Figure (1) shows the configuration for IKM in Delmia.

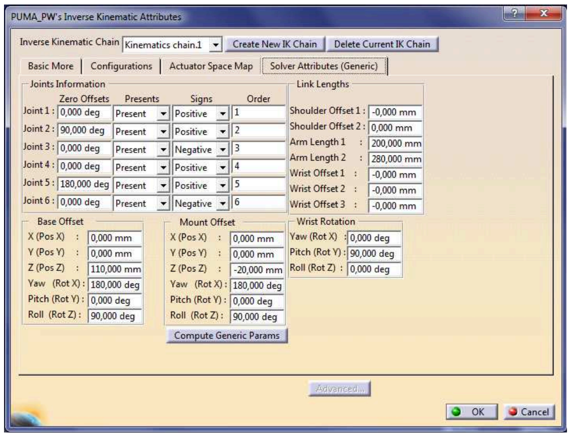

PUMA_PW's Inverse Kinematic Attributes

Inverse Kinematic Chain: Kinematics chain.1

Actuator Space Map — Map Info

	Joints Map	Joints Type	Kin Axis Type	Kin DOF	Kin Part
Command 1 :	dof(1)	Rotational	Rot Z	1	Puma_link_2.1
Command 2 :	dof(2)	Rotational	Rot X	2	Puma_link_3.1
Command 3 :	dof(3)	Rotational	Rot X	3	Puma_link_4.1
Command 4 :	dof(4)	Rotational	Rot Z	4	Puma_link_5.1
Command 5 :	dof(5)	Rotational	Rot X	5	Puma_Link_6.1
Command 6 :	dof(6)	Rotational	Rot Z	6	Puma_Link_7.1

(a) Actuator space map enlists all the joints in the mechanism. Kinematic axis types are defined for each joint in reference to global reference axis.


PUMA_PW's Inverse Kinematic Attributes

Inverse Kinematic Chain: Kinematics chain.1

Solver Attributes (Generic) — Joints Information

	Zero Offsets	Presents	Signs	Order
Joint 1 :	0,000 deg	Present	Positive	1
Joint 2 :	90,000 deg	Present	Positive	2
Joint 3 :	0,000 deg	Present	Negative	3
Joint 4 :	0,000 deg	Present	Positive	4
Joint 5 :	180,000 deg	Present	Positive	5
Joint 6 :	0,000 deg	Present	Negative	6

Link Lengths

Shoulder Offset 1 :	-0,000 mm
Shoulder Offset 2 :	0,000 mm
Arm Length 1 :	200,000 mm
Arm Length 2 :	280,000 mm
Wrist Offset 1 :	-0,000 mm
Wrist Offset 2 :	-0,000 mm
Wrist Offset 3 :	-0,000 mm

Base Offset

X (Pos X) :	0,000 mm
Y (Pos Y) :	0,000 mm
Z (Pos Z) :	110,000 mm
Yaw (Rot X) :	180,000 deg
Pitch (Rot Y) :	0,000 deg
Roll (Rot Z) :	90,000 deg

Mount Offset

X (Pos X) :	0,000 mm
Y (Pos Y) :	0,000 mm
Z (Pos Z) :	-20,000 mm
Yaw (Rot X) :	180,000 deg
Pitch (Rot Y) :	0,000 deg
Roll (Rot Z) :	90,000 deg

Wrist Rotation

Yaw (Rot X) :	0,000 deg
Pitch (Rot Y) :	90,000 deg
Roll (Rot Z) :	0,000 deg

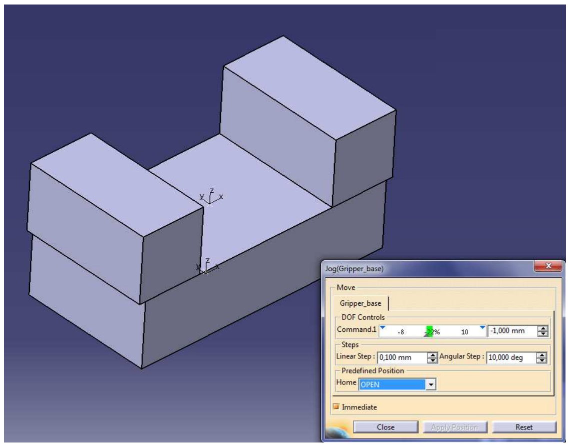
(b) Solver attributes tab shows joint information like offsets and link lengths for the defined inverse kinematic model.

Figure 1: Inverse kinematic model configuration.

Question 1: Inverse Kinematics: Now click ”Jog” mechanism and select ”Home position 1” as predefined position. Now click on “Cartesian” and try to change the TCP (Tool Center Point) coordinates. What can you conclude?

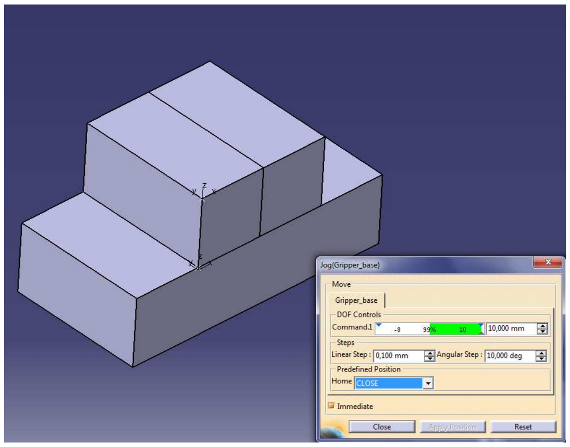
Once the inverse kinematic model (IKM) is defined, jogging also offers manipulation of robot in the Cartesian space as previously mentioned. TCP is assigned the end-effector/tool point of interest. The position and orientation of this point can be varied in Cartesian space and the software computes the respective joint angles. This 6 degree of freedom serial robot has a maximum 8 of IKM solutions. Thus, for a given TCP pose, the simulator offers 8 solutions which are referred to as postures in Delmia. The software also provides information if a certain posture solution is reachable or not.

A gripper was used as the end effector. In order to perform pick and place operations with the gripper, its *open* and *close* positions were defined as shown in figure (2).



Jog(Gripper_base)
Command.1: -8 | -22% | 10 | -1,000 mm
Linear Step : 0,100 mm
Angular Step : 10,000 deg
Home: OPEN

(a) Open position for gripper



Jog(Gripper_base)
Command.1: -8 | 99% | 10 | 10,000 mm
Linear Step : 0,100 mm
Angular Step : 10,000 deg
Home: CLOSE

(b) Close position for gripper

Figure 2: Definition of open and close position for gripper.

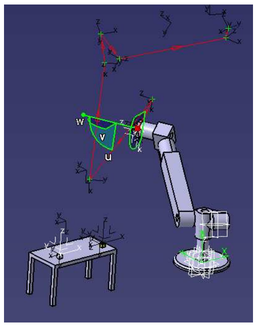
Question 2: Perform simple task definition. Why does the robot change posture? Can we define the posture? Can we have different kind of motion planning? Describe the motion, TCP Trace.

10 different frames were defined using teach a device command to create a simple task of pick and place. Upon simulation, robot's end-effector follows a trajectory joining the frames. The postures are changed based on the set motion type. For example, the linear motion will use for the operation space the Cartesian space to find the IKM solutions. For the joint motion it will be the joint space. Delmia does not seem to provide the capability of showing the current operational space in the 3D environment or in a separate window. The actual choice of which configuration to choose is rather determined by the numerical methods involved in estimating the next possible operation point. In other words there is no way of determining that unless the source code can be seen. As discussed before, the simulation offers 8 different postures for a certain TCP pose and also provides their reachability status. The postures can be changed for specific motion types using in the table format using the teach command, where individual operations/tag points are accessible. Figure (3) shows other options offered in table format like the interpolation mode setting different methods for movement in between tag points or motion profile selecting a defined configuration of velocities.

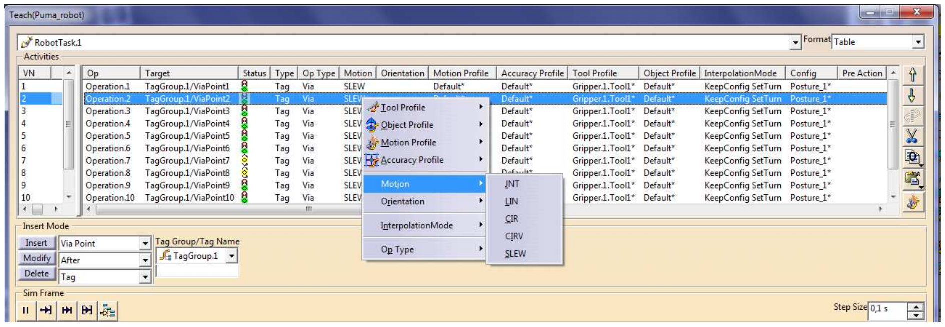
The teach mode also offers 5 different motion types:

- jointmove (JNT),
- linearmove (LIN),
- circluarmove (CIR),
- circularvia (CIRV)
- and slew (SLEW).

These motion types define the trajectory between two frames and how the two points are interpolated. TCP trace can be activated to record the path followed by TCP during simulation run. If motion type LIN is selected, TCP will follow a linear path between the frames. On the other hand, SLEW offers a smooth, but difficult to intuitively predetermine trajectory through tag points.



(a)

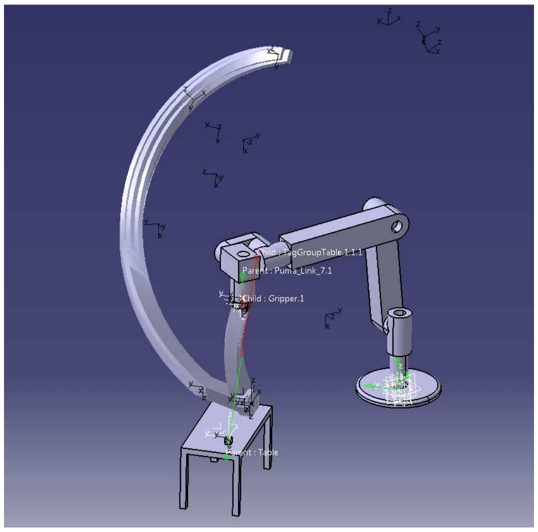


VN	Op	Target	Status	Type	Op Type	Motion	Orientation	Motion Profile	Accuracy Profile	Tool Profile	Object Profile	InterpolationMode	Config	Pre Action
1	Operation.1	TagGroup.1/ViaPoint1		Tag	Via	SLEW		Default*	Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
2	Operation.2	TagGroup.1/ViaPoint2		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
3	Operation.3	TagGroup.1/ViaPoint3		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
4	Operation.4	TagGroup.1/ViaPoint4		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
5	Operation.5	TagGroup.1/ViaPoint5		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
6	Operation.6	TagGroup.1/ViaPoint6		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
7	Operation.7	TagGroup.1/ViaPoint7		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
8	Operation.8	TagGroup.1/ViaPoint8		Tag	Via	SLEV			Default*	Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
9	Operation.9	TagGroup.1/ViaPoint9		Tag	Via	SLEV				Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	
10	Operation.10	TagGroup.1/ViaPoint10		Tag	Via	SLEV				Gripper.1.Tool1*	Default*	KeepConfig SetTurn	Posture_1*	

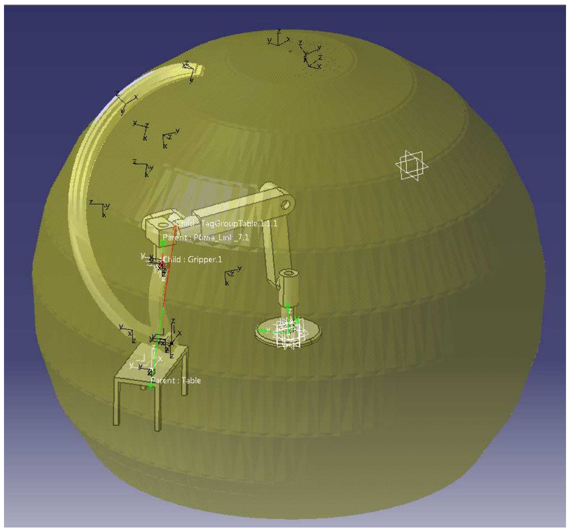
(b)

Figure 3: Teach mode for a simple task definition. In table format (3b), many options are available involving motion type and postures between different tag points.

Additionally for analysis, Delmia also offers visualization of the workspace of the robot and the swept volume for the robot or its end-effector. Figure (4) shows example of swept volume for the specified task and workspace generated for the PUMA robot, which is defined by its joint limits.



(a) Swept volume generated by the end-effector for a small task.

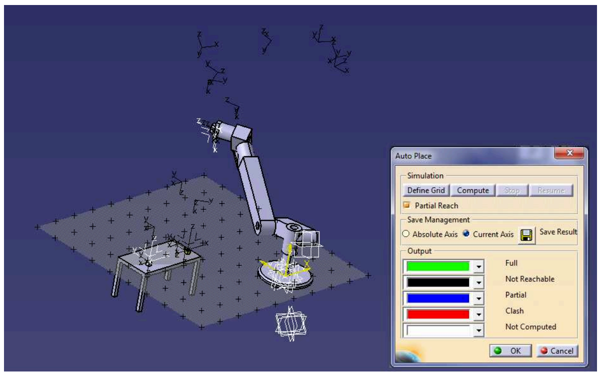


(b) Workspace of the robot enclosing the swept volume generated by the end-effector.

Figure 4: Workspace of PUMA robot and the swept volume of its end-effector.

Question 3: Optimal path placement: Change the location and simulate. What is changed?

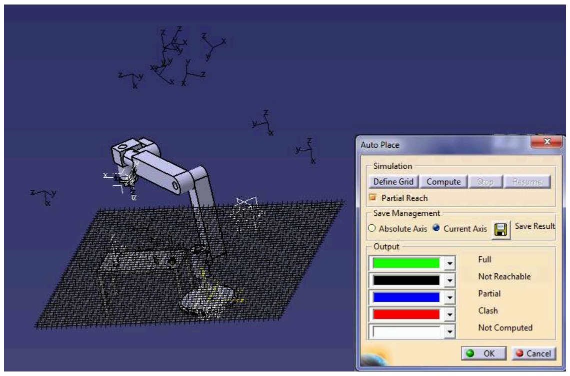
Delmia offers optimal path placement command to place a robot for a given task. First a task is selected, then an certain area is provided using the grid definition call. The algorithm then evaluates at each point of the grid using brute-force method full, or partial if selected, reachability and execution of the task. However, from experience this feature is currently not reliable. On many occasions, the algorithm offers misleading results, such as none of the points are able to execute the task even though many can when simulated manually. Two such examples of described scenarios are shown in figure (5).



[✓ Partial Reach]
Absolute Axis [✓ Current Axis]

🟩	Full
⬛	Not Reachable
🟦	Partial
🟥	Clash
⬜	Not Computed

(a) Optimal placement with 100 points grid.



[✓ Partial Reach]
Absolute Axis [✓ Current Axis]

🟩	Full
⬛	Not Reachable
🟦	Partial
🟥	Clash
⬜	Not Computed

(b) Optimal placement with heavily defined grid.

Figure 5: Optimal placement iterations. The algorithm was unable to found a solution even though many feasible placements existed.

Conclusions

This report briefly discusses and evaluates different features in Delmia concerning kinematics of a 6 degrees of freedom Puma robot and a simple pick and place task definition.

The direct and inverse kinematics of the robot were defined. A gripper was designed and modelled as its end-effector. Various features of the Delmia software were explored for the robotic cell and task definitions. In the latter the command teach a device was extensively used for its execution and changing of different parameters. During the practical session the pick and place operation was carried out with the PUMA robot while analysing different configuration options like the motion types or profiles. Weaknesses of the software were also noted, in particular the inability to find optimal placement for a given task.

Overall the Delmia software framework provides a wide variety of features relevant to modelling of robots and simulation of task in robotics.