NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
«KYIV POLYTECHNIC UNIVERSITY NAMED IGOR SIKORSKY»
INSTITUTE OF MECHANICAL ENGINEERING

Department of Applied hydro-aeromechanics and mechatronics

«Approved for defense»

Head of department
__________ Luhovskyi O.F.
(signature) (name and initials)

"___"_____________2020 year

Bachelor thesis

Specialization: **131 Applied mechanics**
(code and name)

Topic: Modernization of 6R robot
Made by: 4th-course student, group MA-61-2
(code of the group)

Kutovyi Oleksandr Sergiyovich
(name, surname, patronymic) (signature)

Supervisor: docent, Ph.D. Seminska N.V.
(title, name and initials) (signature)

Advisor for: occupational Safety and Health senior lecturer Kovtun A.I.
(name) (title, name and initials) (signature)

Advisor for: Part manufacturing docent, Ph.D. Korenkov V.M.
(name) (title, name and initials) (signature)

Reviewer
(title, name and initials) (signature)

Hereby I confirm that this work does not contain any other author`s work or any external materials, illustrations etc., that were not mentioned in references accordingly.
Student _____________
(signature)

Kyiv - 2020 year

National Technical University of Ukraine
«Kyiv Polytechnic University named Igor Sikorsky»

Institute: Institute of mechanical engineering
(full name)

Department: Applied hydro-aeromechanics and mechatronics
(full name)

Higher education type - first (bachelor)

Specialty: 131 Applied mechanics
(code and name)

CONFIRMED

Head of department

___________ Luhovskyi O.F.
(signature) (Initials)

"_____"___________2020 year

Task
For student`s bachelor thesis

Kutovyi Oleksandr Sergiyovich
1. Topic: Modernization of 6R robot;
Supervisor: Docent, Ph.D. Seminska Nataliya Valeryivna
Stated in the university decree from "20th" of May 2020 year № 1120-c
2. Final deadline – 10.06.2020
3. Initial conditions for the task: Manipulator on mobile platform Kuka YouBot, hydraulic fluid – HLP 32, pressure in the system – 4.5 MPa, rod stroke – 500 mm.

4. Content of explanatory note:
Chapter 1: introduction, deriving the formulas for robot movement control .
Chapter 2: theory and calculation of stepper motor.
Chapter 3: hydraulic scheme and calculation of cylinders.
Chapter 4: part machining process in CATIA V5.
Chapter 5: safety and precautions.
Chapter 6: conclusion.
Chapter 7: references.

5. List of graphic materials (including necessary drawings, posters, presentations, etc.) 4 A1 sheets :
Sheet 1: Work area of Kuka YouBot. General view.
Sheet 2: Lift hydraulic drive. Hydraulic scheme.

Sheet 3: Hydraulic cylinder. Assembly.
Sheet 4: Detailing.
6. Advisors

Chapter	Title, name and initials of advisor	Signature, date	
		Task issued	Task accepted
1. occupational Safety and Health	senior lecturer Kovtun A.I.		
2. Part manufacturing	docent, Ph.D. Korenkov V.M.		
3. Introduction, deriving the formulas for robot movement control.	docent, Ph.D. Seminska N.V.		

7. Date of issuing the task: 10.11.2019

SCHEDULE

#	Milestone name	Deadline	Annotation
1	Overview of existing mobile manipulators	10.11.2019	
2	Manipulator selection	01.01.2020	
3	Kinematics of the manipulator	28.02.2020	
4	Sheet 1: Work area of Kuka YouBot. General view.	05.02.2020	
5	Complete code. Simulation.	05.04.2020	
6	Calculation of stepper motor	15.04.2020	
7	Hydraulic scheme description	20.04.2020	
8	Sheet 2: Lift hydraulic drive. Hydraulic scheme.	10.05.2020	
9	Hydraulic cylinder calculation	15.05.2020	
10	Sheet 3: Hydraulic cylinder. Assembly.	20.05.2020	
11	Sheet 4: Detailing.	01.06.2020	
12	Part manufacturing	05.06.2020	
13	occupational Safety and Health	08.06.2020	
14	Project wrap and documentation	09.06.2020	

Student	(signature)	**Kutovyi O.S.** (name and initials)
Supervisor	(signature)	**Seminska N.V.** (name and initials)

BACHELOR THESIS CONTENT

#	Format	Denotation	Denomination	Number of sheets	Annotation
1	A4	-	Task for bachelor thesis	2	
2	A4	DP MA61206. 00.000 EN	Explanatory note	54	
3	A1	DP MA61206. 01.000 A	Hydraulic cylinder. Assembly	1	
4	A4	DP MA61206. 01.01.000	Sleeve	1	
5	A3	DP MA61206. 01.02.000	Rod cover	1	
6	A4	DP MA61206. 01.03.000	Cuff holder	1	
7	A4	DP MA61206. 01.04.000	Rod	1	
8	A4	DP MA61206. 01.000 A	Specification	2	
9	A1	DP MA61206. 02.000 HS	Lift hydraulic drive. Hydraulic scheme	1	
10	A1	DP MA61206. 03.000 GV	Work area of Kuka YouBot. General view	1	

	Initials	Sign.	Date			
Made by:	Kutovyi O.S.			Bachelor thesis content	Page	Pages
Supervisor	Seminska N.V.				1	63
Consultant					KPI, I. Sikorsky Dep. AGM, Gr.MA-61-2	
N/contr.						
Head of department						

Explanatory note to bachelor thesis

Modernization of 6R robot

Kyiv – 2020

Анотація

У цій роботі автор розглянув можливості маніпулятора на мобільній платформі Kuka YouBot для використанні його у лабораторіях чи майстернях. Завдяки використанню мобільної платформи площа роботи такого робота значно збільшується. Для такої платформи було розроблене програмне рішення, яке приймає на вхід поточне та бажане положення об’єкта у просторі, а на виході отримаємо необхідні кути повороту для кожного проміжку часу, які далі передаються у контролер типу Feedforward + PI Feedback, налаштування якого також були розглянуті. На підставі отриманих значень була проведена симуляція у середовищі CoppeliaSim EDU, яка демонструє усі вище надані переваги.

Окрім програми для керування роботом, також була запропонована додаткова гідросистема і розраховані гідроциліндри для фіксації та переміщення об’єкта, над яким проводяться маніпуляції роботом. Така гідросистема надає змогу надійно закріпити об’єкт, наприклад мотоцикл, а потім за допомогою великого гідроциліндра підняти його угору для збільшення можливостей щодо маніпуляцій, оскільки деякі операції, як от заміна мастила у двигуні, потребують доступу до нижньої частини апарату.

У цьому проекті були використані наступні програми:

- Python 3.7 + PyCharm IDE
- CoppeliaSim Edu
- DS Catia V5

Далі був розглянутий тех. процес для створення монтажної пластини робота – частина маніпулятора, до якої кріпиться корисна оснастка, наприклад хват, зварювальний апарат тощо. Була розроблена послідовність обробки, враховуючи параметри інструменту. Підведені підсумки щодо тривалості та вартості виробнитства такої деталі.

Наприкінці техніка безпеки і розрахунок необхідних умов у робочому приміщенні.

Abstract

In this paper, the author considered the possibilities of the manipulator on the mobile platform Kuka YouBot for use in laboratories or workshops. Due to the use of a mobile platform, the work area of such manipulator is significantly increased. For such a platform, a software solution was developed that takes the current and desired position of the object in space at the input, and at the output produces the necessary angles of rotation for each moment of time, which are then transmitted to the controller type Feedforward + PI Feedback, which settings were also considered . Based on the obtained values, a simulation was performed in the CoppeliaSim EDU environment, which demonstrates all the above advantages.

In addition to the robot control program, a supplementary hydraulic system was proposed, and hydraulic cylinders were designed to lock and move the object being manipulated by the robot. This hydraulic system allows you to properly secure an object, such as a motorcycle, and then use a large hydraulic cylinder to lift it up to increase handling, as some operations, such as changing the engine oil, which require access to the bottom part of the unit.

The following software were used in this project:

- Python 3.7 + PyCharm IDE
- CoppeliaSim Edu
- DS Catia V5

Next was considered machining process for creating a mounting plate of the robot - a part of the manipulator to which useful equipment is attached, such as a grip, a welding machine, etc. A machining sequence was developed,

considering the parameters of the tool. Summarized the duration and cost of production of such part.

Finally, safety, precautions and necessary conditions at the workplace.

Contents

					DP MA61206. 00.000 EN				
Ed.	Page	№ doc.	Sign	Date					
Made by		Kutovyi O.S..			6R robot modernization. Explanatory note	Lit.	Mass	Scale	
Checked by		Seminska N.V.							
T.contr.						Page. 1	Pages 54		
Review						MMI			
N. Contr.									
Approved by									

1. Introduction

Modern lifestyle and consumption market as a whole dictate the rules to the manufacturers of goods and services. When some goods or services are made in a small amount the requests can be handled by human workers. Because the scale is small it is easy to track the processes and maintain the schedule. Upon evolution companies tend to hire more staff in the beginning, but it shortly backfires in terms of human errors and work ethic. With scaling in some areas of business and in mostly every single one, where good manufacturing is a key component, automation is a good choice, that neglects all the flaws of human workers but comes with its very own pros and cons.

When someone considers installing couple of industrial manipulators instead of hiring new staff the following must be concerned.

Pros:

- Great precision
- High repeatability
- Lower cost of assembling/manufacturing of goods
- Can work 24/7 (considering proper technical maintenance)
- Can work in hazardous environment

Cons:

- Requires new trained staff to program and maintain
- Significant damage will be done in case if initial process was not properly engineered/ programmed
- Even minor adjustments require full stop of a production line for a significant period of time

Every unique problem can be solved in different ways, and that is why industrial robots come in variety of shapes and sizes. The smallest are used in a precision stitching or welding, when at the same time the biggest representatives can lift and move whole cars.

Besides load capacity robots are subdivided in 4 major form-factor:

- Cartesian
- SCARA
- 6-axis

- Delta-robots

Here is a brief explanation of each.

- Cartesian robot is typically a frame consisting of linear actuators, moving the end-effector in all 3 axes respectively, thus moving in a Cartesian space. Typical exemplars are 3-d printers, cutters, millers etc.
- SCARA or otherwise “Pick-and-Place” robots are intermediate link between 6R robots and Cartesian ones. They have typically 2 revolute joints and 1 linear allowing faster movement and slightly greater versatility than Cartesian. Because SCARAs have the easiest integration they seem like the best solution for the majority of applications, but Cartesians are more common because of their level of customization.
- 6R robots are the most versatile and customizable of them all. As it comes from the name, they have 6 degrees of freedom, allowing the end-effector movements that are similar to humans. They are most commonly used in heavy machinery manufacturing, welding jobs etc. But all this comes at a price of complex programming, and for redundant robots – those with more than 6 degrees of freedom, every configuration can be achieved in different ways, thus greatly increasing the complexity and versatility. It makes these robots desirable only for non-private use, however some manufacturers and individuals have recently begun production of smaller ones for “civilian” use.
- Delta robots are the fastest and most expensive. They have a unique, dome-shaped work envelope in which they can achieve very high speeds. Delta robots are best for fast pick-and-place or product transfer applications like moving parts from a conveyor belt and placing them in boxes or onto another conveyor belt. They also come as complete solutions for machine designers but are more complicated in use than the 6-Axis or SCARA robots. The main advantage of Delta robots is their speed and precision with which they operate.

3.1. Mobile platforms

In some cases, it is useful to mount a n-R manipulator, where n – number of revolute joints, on a mobile platform. Such modification greatly increases the feasible working space and available tasks to robot. As an example, consider a NASA rover “Curiosity”.

It is capable of vast range of movements due to the rotating wheels and also precise manipulations for soil examination etc.

Another great example is sapper robot. It is built on track-platform for enhanced grip and can defuse bombs with an utmost precision and carefulness thanks to the installed manipulator.

These two examples are certainly not the unique in terms of applicability of mobile robotic platforms, but clearly states the point in dealing with most challenging tasks either exploring the far horizons or providing a security in dense environments.

The author of this work proposes the concept as well as math and programming for Kuka YouBot mobile platform for use in dense environment such as workshop. Its main purpose is to aid mechanics in monotonous tasks and working side by side with humans. This concept requires active obstacle avoidance algorithm as well as custom gripper for performing most common tasks as screwing and moving objects.

3.2. Kinematics of the YouBot

Before deriving complex formulas for robot kinematics and control it is necessary to understand some basic concepts. It includes but not limited to: transformation matrix, screw axis and twist, Jacobian matrix.

3.3. Transformation matrices

Any point can be represented in the Cartesian space with three numbers, ultimately describing its position in three corresponding coordinates. But when it comes to describing positions of the body, consisting of multiple rigidly connected points, it is most common practice to set a coordinate system to some definitive position on this body and then represent this coordinate system in the given and fixed space frame. So, rather then just three coordinates for bodies transformation matrices are used. It consists of rotation matrix and some vector p. Rotation matrix is used to represent a “rotation” of body frame relative to the fixed space frame, and at the same time vector p represents position of the body frame in the space frame. The special Euclidean group SE(3), also known as the group of rigid-body motions or homogeneous transformation matrices in R^3, is the set of all 4x4 real matrices T of the form

$$T_{sb} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The row of zeros and one is introduced for ease of computations.

So, for example, if body frame does not rotate and only moves along the vector p the transformation matrix would look like this:

$$T_{sb} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Or if it only rotates, so new Y-axis pointing to space X-axis, X-axis in the direction of minus Y-axis and Z-Axis remains we get:

$$T_{sb} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is fairly easy to represent any arbitrary angle of rotation and translation along some vector with transformation matrix, and with the help of last row of zeros and one perform computations on these matrices.

Screw axis and twist

It is rather common task when body moves in the space, as such T_{sb} is no longer stationary and becomes time dependent. To get $T_{sb}(t)$ we need to pre- or post-multiply derivative of T by the inverse of T as described in [1] (p. 95 - p. 98).

As a result of we get what is called a twist:

$$V = \begin{bmatrix} w \\ v \end{bmatrix} \in \mathbb{R}^6$$

w – represents angular velocity;

v – represents linear velocity.

Since a screw axis S is just a normalized twist, the 4x4 matrix representation [S] of S = (w, v) is:

$$[S] = \begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix} \in se(3)$$

And

$$[w] = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \in so(3)$$

Is a skew-symmetric representation of w.

3.4. Jacobian matrix

The Jacobian matrix represents the linear sensitivity of the end-effector velocity x\` to the joint velocity θ\` and it is a function of the joint variables θ. In a simple manner Jacobian matrix defines how much joint rotation affects the total displacement, speed and end-effector force of the robot. As can be seen in [1] (p. 170 – p. 175) the derivation is somewhat similar to closed vector form equations, used in Theory of machines. On the other hand, writing correct equations for simple 2R robot is long and takes time. For industry-standard 6R robot this form is deemed to be far too sophisticated. Instead, using properties of transformation matrices and exponential representation of a robot, precise Jacobian matrix can be generated fast and effective. The space Jacobian $J_s(\theta) \epsilon R^{6xn}$ relates the joint rate vector θ\` ϵR^n to the spatial twist V_s via

$$V_s = J_s(\theta)\dot{\theta}$$

The ith column of $J_s(\theta)$ is

$$J_{si}(\theta) = Ad_{e^{[Si]\theta i} \ldots e^{[Si-1]\theta i-1}}(S_i)$$

To understand the physical meaning behind the columns of $J_s(\theta)$, observe that the ith column is of the form $Ad_{T_{i-1}}(S_i)$, where $T_{i-1} = e^{[Si]\theta i}$ ... $e^{[Si-1]\theta i-1}$; recall that S_i is the screw axis describing the ith joint axis in terms of the fixed frame with the robot in its zero position. $Ad_{T_{i-1}}(S_i)$is therefore the screw axis describing the ith joint axis after it undergoes the rigid body displacement T_{i-1}. Physically this is the same as moving the first i1 joints from their zero position to the current values $\theta_1 \ldots \theta_{i-1}$. Therefore, the ith column $J_{si}(\theta)$ of $J_s(\theta)$ is simply the screw vector describing joint axis i, expressed in fixed-frame coordinates, as a function of the joint variables $\theta_1 \ldots \theta_{i-1}$.

In summary, the procedure for determining the columns J_{si} of $J_s(\theta)$ is similar to the procedure for deriving the joint screws S_i in the product of exponentials formula $e^{[S1]\theta 1} \ldots e^{[Sn]\theta n}M$: each column $J_{si}(\theta)$ is the screw vector describing joint axis i, expressed in fixed-frame coordinates, but for arbitrary θ rather than $\theta = 0$.

3.5. Omnidirectional wheels

Robots come in different varieties and shapes, but almost all of them have some features in common. That implies some mobile platform and an end-manipulator. The purpose is to move in wide range within an allowable space and to perform high-precision manipulations in the desired position for mobile platform and manipulator, respectively. Bearing in mind, that different terrains and conditions are suitable for different types of robotic platforms, as for example differential drive robot that has two independently controlled wheels and one or more to keep it planar; a regular car-like platform; a platform with omnidirectional or mecanum wheels that allow skidding in other direction than forward or backward; and tank-like robot with tracks.

Specifically, one group of car-like robots are the hardest to control in term of planning the trajectory. This is due to nonholonomic constraints to the configuration of the chassis – the car cannot move sideways directly, it can achieve net sideway displacement by "parallel parking" maneuvers, however.

The author for his project of mobile manipulator has chosen a platform with four mecanum wheels. This setup allows to move in any arbitrarily direction only by spinning the wheels in different directions and different speeds. In order to get angular speed of each particular wheel from chassis velocity $\dot{q} = (\dot{\varphi}, \dot{x}, \dot{y})$ we set the chassis frame {b} that is rigidly attached to the moving base of the robot, its initial configuration $q = (\varphi, x, y)$, which is expressed in the fixed space frame {s}. Each wheel configuration is set by three numbers (β_i, x_i, y_i), which are expressed, however, in the base frame {b}. Each wheel has radius r_i, and since this an omnidirectional wheel, there is an angle γ_i, describing the free-sliding direction. For all the chosen wheels it is set to 45 deg. or -45 deg. The configuration of the frame {b} of the mobile base, relative to the frame {s} on the floor, is described by the 3-vector $q = (\phi, x, y)$ or the $SE(3)$ matrix:

$$T_{sb} = \begin{bmatrix} cos\varphi & -sin\varphi & 0 & x \\ sin\varphi & cos\varphi & 0 & y \\ 0 & 0 & 1 & 0.0963 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $z = 0.0963$ meters is the height of the {b} frame above the floor. The forward-backward distance between the wheels is $2l = 0.47$ meters and the side-to-side distance between wheels is $2w = 0.3$ meters. The radius of each wheel is $r = 0.0475$ meters. The wheel numbering and forward driving and "free sliding" direction γ of each wheel is indicated in the figures.

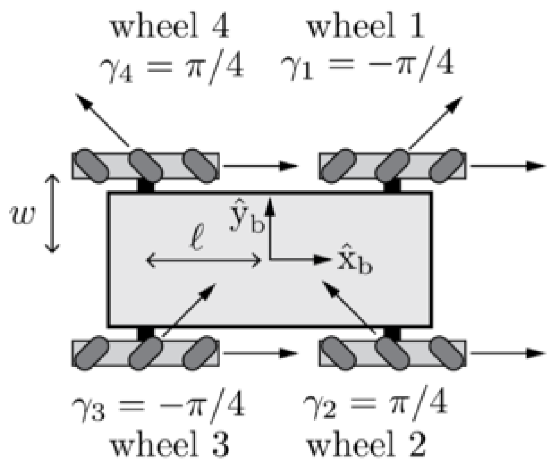


Figure 1. http://hades.mech.northwestern.edu/

Mobile platform with four mecanum wheels

With these conditions it is possible now to get individual angular speed of each wheel.

$$u_i = h_i(\varphi)\dot{q} =$$

$$= \begin{bmatrix} \frac{1}{r_i} & \frac{tan\,\gamma_i}{r_i} \end{bmatrix} \begin{bmatrix} cos\,\beta_i & sin\,\beta_i \\ -sin\,\beta_i & cos\,\beta_i \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\varphi & sin\,\varphi \\ 0 & -sin\,\varphi & cos\,\varphi \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

Solving for $h_i(\varphi)$ we get:

$$h_i(\varphi) = \frac{1}{r_i \cos\gamma_i} \begin{bmatrix} x_i \sin(\beta_i + \gamma_i) - y_i \cos(\beta_i + \gamma_i) \\ \cos(\beta_i + \gamma_i + \varphi) \\ \sin(\beta_i + \gamma_i + \varphi) \end{bmatrix}^T$$

When considering the whole robot platform vector of driving wheel speeds $u \in \mathbb{R}^m$ is obtained by stacking the m rows of $h_i(\varphi)$:

$$u = H(\varphi)\dot{q} = \begin{bmatrix} h_1(\varphi) \\ h_2(\varphi) \\ \vdots \\ h_m(\varphi) \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

Furthermore, it is useful to express the relationship between body twist V_b and u. This translation does not depend on the chassis orientation, however.

$$u = H(0)V_b = \begin{bmatrix} h_1(0) \\ h_2(0) \\ \vdots \\ h_m(0) \end{bmatrix} \begin{bmatrix} w_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}$$

It is worth noting, that orientation of free-sliding direction γ_i must be chosen specifically, so rank of $H(0)$ is 3. If this condition is not met, robot could not result at some arbitrary motions, ultimately leading to failure and inability to perform tasks. That is why author has chosen wheel configuration as shown in figure 2.

For some small amount of time u can be expressed as an angular rotation or change in angle of the wheels $\Delta\theta = H(0)V_b$. Solving for V_b:

$$V_b = H(0)^\dagger \Delta\theta = F\Delta\theta$$

Where $F = H(0)^\dagger$ is a pseudoinverse of $H(0)$.

For four-mecanum-wheel robot it simplifies to:

$$V_b = F\Delta\theta = \frac{r}{4}\begin{bmatrix} -1/(l+w) & 1/(l+w) & 1/(l+w) & -1/(l+w) \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}\Delta\theta$$

After short period of time new frame T$_{bb`}$ expresses new chassis frame {b`} in the initial frame{b}. From it is possible to get the change in coordinated in initial frame {b} $\Delta q_b = (\Delta\varphi_b, \Delta x_b, \Delta y_b)$ in terms of (w_{bz}, v_{bx}, v_{by}) .

If $w_{bz} = 0$, then

$$\Delta q_b = \begin{bmatrix} \Delta\varphi_b \\ \Delta x_b \\ \Delta y_b \end{bmatrix} = \begin{bmatrix} 0 \\ v_{bx} \\ v_{by} \end{bmatrix}$$

If $w_{bz} \neq 0$, then

$$\Delta q_b = \begin{bmatrix} \Delta\varphi_b \\ \Delta x_b \\ \Delta y_b \end{bmatrix} = \begin{bmatrix} w_{bz} \\ (\ v_{bx} \, sin \, w_{bz} + v_{by}(cos \, w_{bz} - 1))/w_{bz} \\ (v_{by} \, sin \, w_{bx} + v_{by}(1 - cos \, w_{bz}))/w_{bz} \end{bmatrix}$$

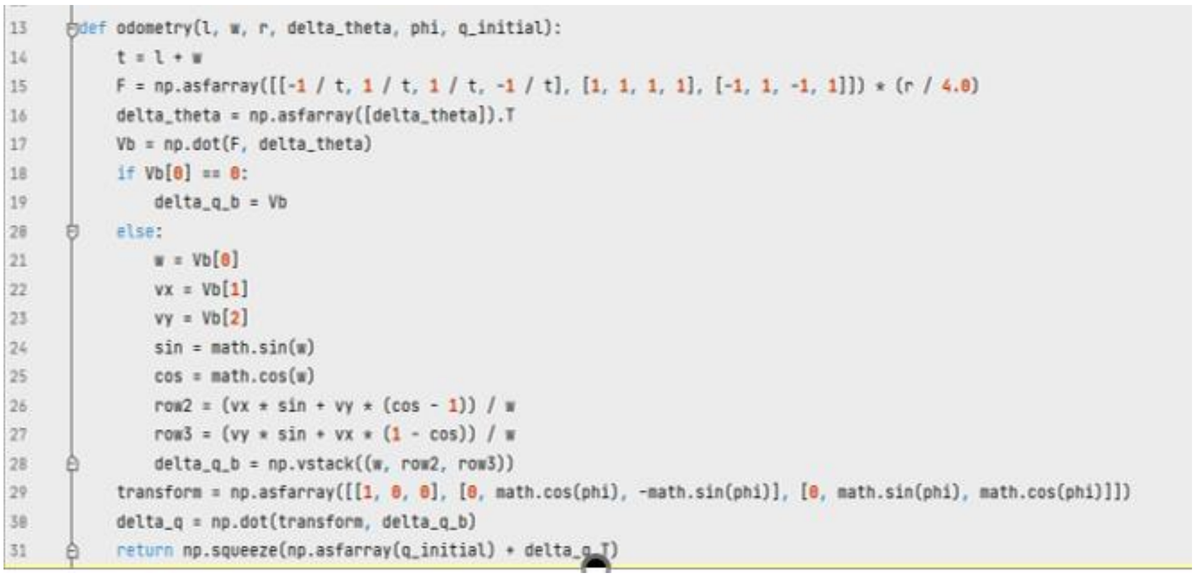
Then it is necessary to represent Δq_b in {s}, as such:

$$\Delta q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos \, \varphi_k & -sin \, \varphi_k \\ 0 & sin \, \varphi_k & cos \, \varphi_k \end{bmatrix}\Delta q_b$$

The final updated configuration is for next timestep k+1:

$$q_{k+1} = q_k + \Delta q$$

In Figure 2 the piece of software for this formula in Python 3.7 is shown:



```python
def odometry(l, w, r, delta_theta, phi, q_initial):
    t = l + w
    F = np.asfarray([[-1 / t, 1 / t, 1 / t, -1 / t], [1, 1, 1, 1], [-1, 1, -1, 1]]) * (r / 4.0)
    delta_theta = np.asfarray([delta_theta]).T
    Vb = np.dot(F, delta_theta)
    if Vb[0] == 0:
        delta_q_b = Vb
    else:
        w = Vb[0]
        vx = Vb[1]
        vy = Vb[2]
        sin = math.sin(w)
        cos = math.cos(w)
        row2 = (vx * sin + vy * (cos - 1)) / w
        row3 = (vy * sin + vx * (1 - cos)) / w
        delta_q_b = np.vstack((w, row2, row3))
    transform = np.asfarray([[1, 0, 0], [0, math.cos(phi), -math.sin(phi)], [0, math.sin(phi), math.cos(phi)]])
    delta_q = np.dot(transform, delta_q_b)
    return np.squeeze(np.asfarray(q_initial) + delta_q.T)
```

Figure 2 Python 3.7 code for q_{k+1}

Initial conditions

The following figure illustrates the arm at its home configuration (all joint angles zero) and the frames {s}, {b}, {0}, and {e}. For the image on the right, joint axes 1 and 5 (not shown) point upward and joint axes 2, 3, and 4 are out of the screen.

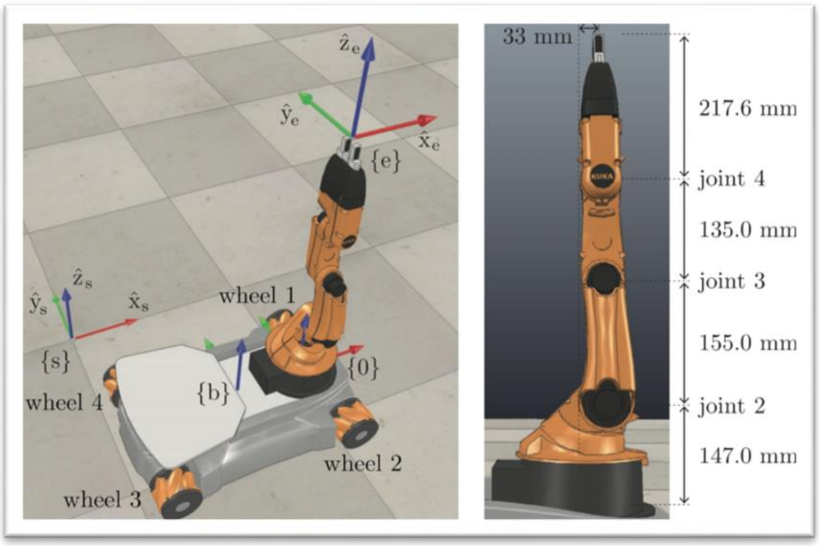


Figure 3. http://hades.mech.northwestern.edu/

The matrix T_{b0} represents {0} – base of the robotic arm in {b}:

$$T_{b0} = \begin{bmatrix} 1 & 0 & 0 & 0.033 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.6546 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

When robot is at home position (Figure 3) each joint – rotation axis is represented as a screw-axis $\mathcal{B}_i$ in the end-effector frame {e}:

$$\mathcal{B}_1 = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0.033 \\ 0 \end{bmatrix}$$

$$\mathcal{B}_2 = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ -0.5076 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B}_3 = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ -0.3526 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B}_4 = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ -0.2176 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B}_5 = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In this work for demonstration purposes a cube with the side of 5 cm was chosen.

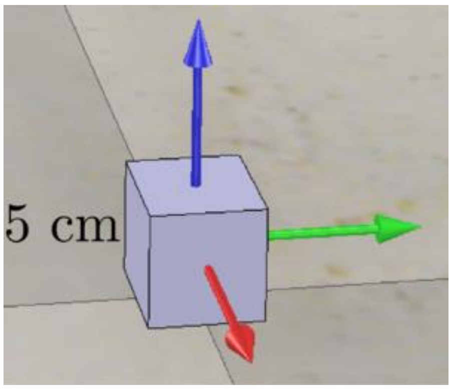


Figure 4. The object being manipulated is a cube, 5 cm x 5 cm x 5 cm. The cube's frame {c} is at its center, and the axes are aligned with the edges of the cube.

Cube initial position {c}, expressed in space frame {s}:

$$T_{sc\ initial} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The goal configuration:

$$T_{sc\ goal} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

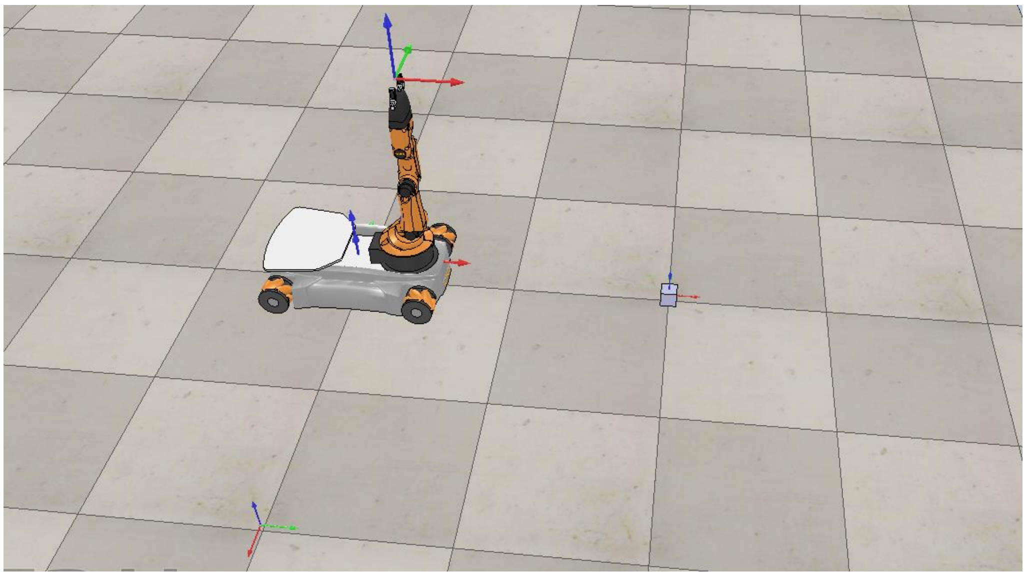


Figure 5. Initial configuration.

3.6. Reference trajectory generation

Any motion of a robot is a process of continuous interaction between controllers and the software. Software is responsible for correct command to the controllers, and controllers are there for relentless execution and constant monitor of present state.

As an example, consider regular human driving regular car. The person is somewhat similar to the software – he decides where they should move at ay particular moment of time. Decision comes from goal destination, current destination, road surface, nearby vehicles and obstacles. Human translate desired change of direction and speed via steering wheel and pedals.

Car, on the other hand, is responsible for the utmost precise execution of the commands, made by the driver. Trough set of electric impulses, wires and linkages desired change is executed. The successful combination of commands and execution leads to safe driving.

In this project ultimate goal is simplicity. User can only set initial configuration and goal configuration, and software should command the path and speeds to achieve these conditions (if physically feasible). As such, almost every piece the trajectory is generated by the software. A trajectory is defined as smooth function, dependent on time, it also should consider any preset limits on joint speeds and positions.

Besides the path – a purely geometric representation of the sequence of desired configuration, achieved by the robot, also a time scaling is necessary, which declares the time steps when some configuration must be met.

3.7. Time scaling

A path or trajectory $\theta(s)$ maps a scalar parameter s, which is set to be 0 at the beginning of the path, and 1 at the very end. As s goes from 0 to 1, the robot moves along the path. In some cases, s is allowed to lie within 0 and T – total motion time, however, it is useful to separate the role of geometric path parameter s from the time parameter t. What time scaling does – it assigns a value s to each time $t \in [0, T], s : [0, T] \rightarrow [0, 1]$.

Time scaling is useful for smooth motion planning. It allows to gradually accelerate and decelerate robot without dealing any damage to links and/or environments, it works in.

Consider an example where robot end-effector at initial position $X_{start} \in SO(3)$ after some period of time must be at end configuration $X_{end} \in SO(3)$.

Using subscript cancellation rule to express the end configuration in the start frame:

$$X_{start,end} = X_{start,s}X_{s,end} = X^{-1}{}_{s,start}X_{s,end}$$

After this, the matrix representation of a twist, that takes X_{start} to X_{end} in unit time is $log\ (X^{-1}{}_{s,start}X_{s,end})$. Then path can be written as:

$$X(s) = X_{start}exp\ (log\big(X^{-1}{}_{s,start}X_{s,end}\big)\ s)$$

where X_{start} is post-multiplied by the matrix exponential since the twist is represented in the {start} frame, not the fixed world frame {s}.

This formula results in a “straight line” motion in terms of that screw axis is constant. If straight line in Cartesian space is desired, it is reasonable to use decoupled translation and rotation $X = (R, p)$:

$$p(s) = p_{start} + s(p_{end} - p_{start})$$

$$R(s) = R_{start} exp\ (log\ (R_{start}^T R_{end})s)$$

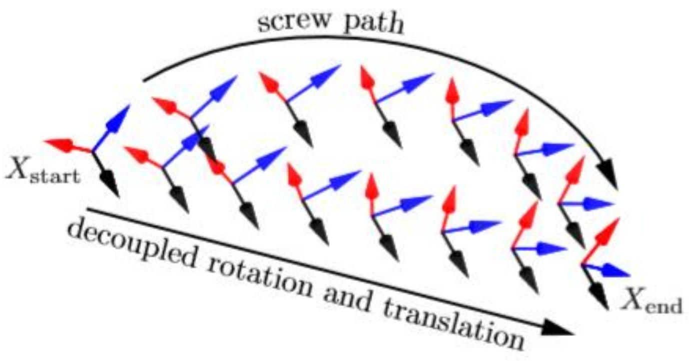


Figure 6. A path following a constant screw motion versus a decoupled path where the frame origin follows a straight line and the angular velocity is constant

3.8. Polynomial time scaling

The convenient form for the time scaling is some n-order polynomial of time. A 3rd order polynomial soothes the transition in terms of speed but does not consider jerk – time derivative of acceleration.

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Solving with initial $s(0) = \dot{s}(0) = 0$, and final conditions $s(T) = 1$, $\dot{s}(T) = 0$:

$$a_0 = 0, \qquad a_1 = 0, \qquad a_2 = \frac{3}{T^2}, \qquad a_3 = -\frac{2}{T^3}$$

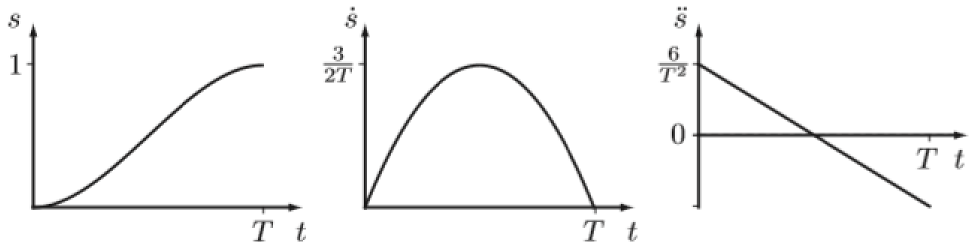


Figure 7. Plots of $s(t), \dot{s}(t), \ddot{s}(t)$ for a third-order polynomial time scaling

In this work author has chosen a 5th order time scaling, that takes into account jerk.

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Solving with initial $s(0) = \dot{s}(0) = \ddot{s}(0) = \ddot{s}(T) = 0$, and final conditions $s(T) = 1$, $\dot{s}(T) = 0$:

$$a_0 = 0, \qquad a_1 = 0, \qquad a_2 = 0, \qquad a_3 = \frac{10}{T^3}$$

$$a_4 = -\frac{15}{T^4}, \quad a_5 = \frac{6}{T^5}$$

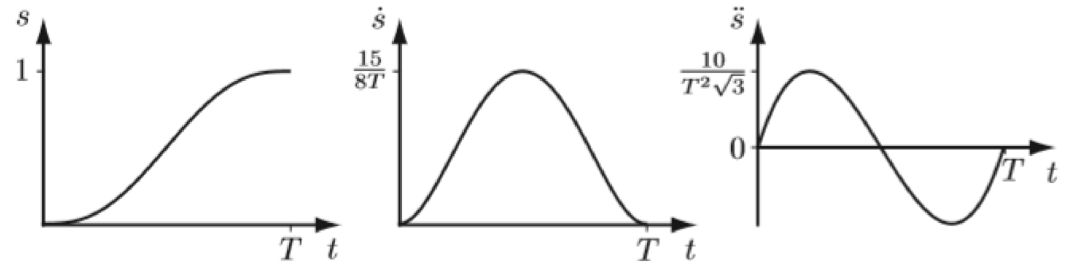


Figure 8. Plots of $s(t), \dot{s}(t), \ddot{s}(t)$ for a fifth-order polynomial time scaling

All the transformations above are used to generate the trajectory for end-effector manipulator. The whole path in this project is subdivided in 8 smaller paths as follows:

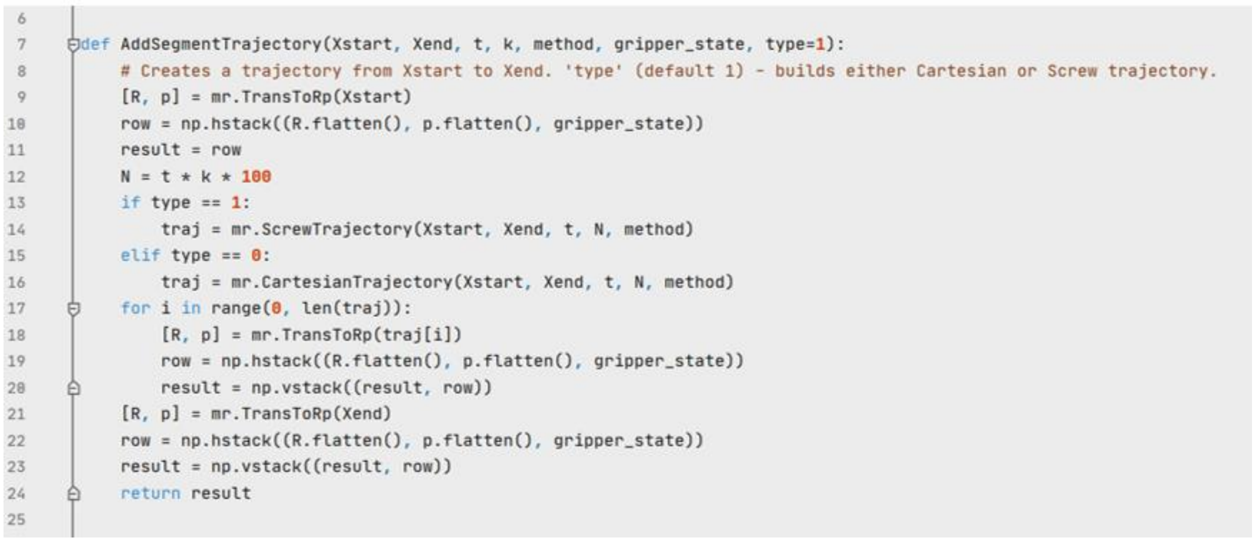
1. The robot moves from its initial configuration to the standoff – position above the cube, but not sufficiently close to grasp it;
2. Robot moves to grasp position
3. Robot grasps the cube
4. obot moves to standoff position, with cube being rigidly held in claws
5. Robot moves to a standoff position before final goal
6. Robot puts down the cube
7. Robot releases the cube
8. Robot moves to the standoff position

Such complications in the trajectory are necessary because of real life applications. In the segments 1 and 5 maximum speed is desired, during the others, however, more gentle movement is required, so manipulator won`t damage the object.

As a result, the following configurations are needed to be specified:

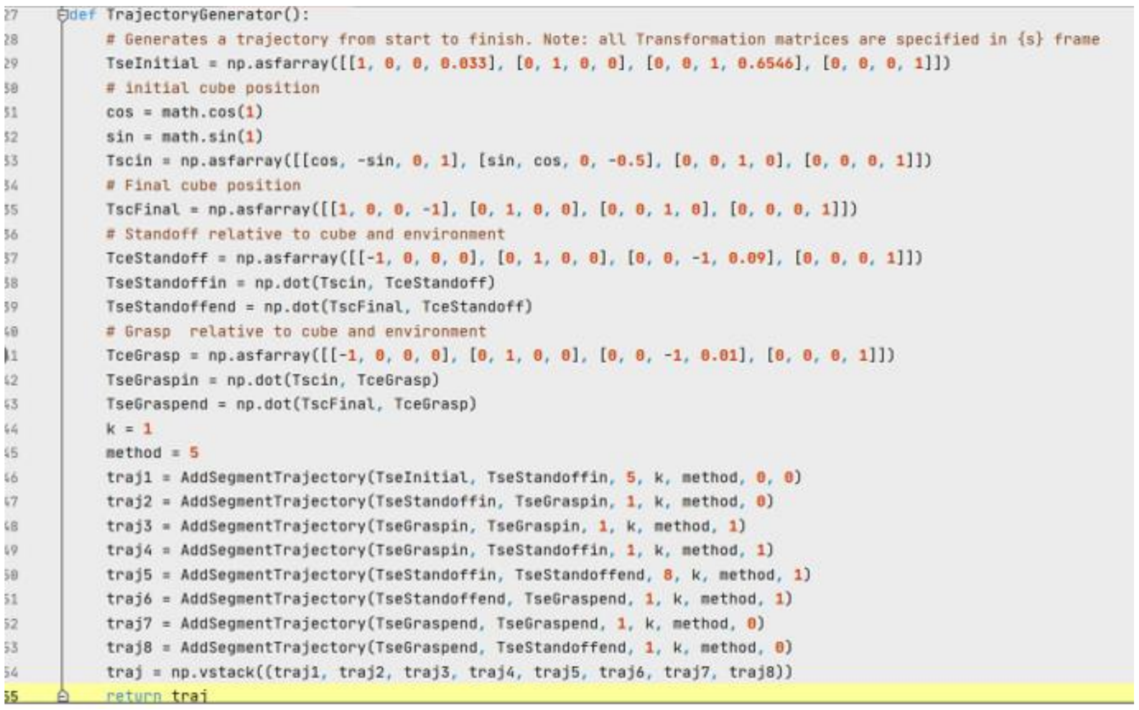
- The initial configuration of the end-effector in the reference trajectory: $T_{se,initial}$.
- The cube's initial configuration: $T_{sc,initial}$.
- The cube's desired final configuration: $T_{sc,initial}$.
- The end-effector's configuration relative to the cube when it is grasping the cube: $T_{ce,grasp}$.
- The end-effector's standoff configuration above the cube, before and after grasping, relative to the cube: $T_{ce,standoff}$. This specifies the configuration of the end-effector {e} relative to the cube frame {c} before lowering to the grasp configuration $T_{ce,grasp}$., for example.
- The number of trajectory reference configurations per 0.01 seconds: k. The value k is an integer with a value of 1 or greater. Although the final animation will be based on snapshots separated by 0.01 seconds in time, the points of generated reference trajectory (and he chosen controller servo cycle) can be at a higher frequency. For example, if user wants his controller to operate at 1000 Hz, he should choose k = 10 (10 reference configurations, and 10 feedback servo cycles, per 0.01 seconds). In this project k = 1 was set.

The resulting code to add each particular trajectory segment is given in a Figure 9.



```python
def AddSegmentTrajectory(Xstart, Xend, t, k, method, gripper_state, type=1):
    # Creates a trajectory from Xstart to Xend. 'type' (default 1) - builds either Cartesian or Screw trajectory.
    [R, p] = mr.TransToRp(Xstart)
    row = np.hstack((R.flatten(), p.flatten(), gripper_state))
    result = row
    N = t * k * 100
    if type == 1:
        traj = mr.ScrewTrajectory(Xstart, Xend, t, N, method)
    elif type == 0:
        traj = mr.CartesianTrajectory(Xstart, Xend, t, N, method)
    for i in range(0, len(traj)):
        [R, p] = mr.TransToRp(traj[i])
        row = np.hstack((R.flatten(), p.flatten(), gripper_state))
        result = np.vstack((result, row))
    [R, p] = mr.TransToRp(Xend)
    row = np.hstack((R.flatten(), p.flatten(), gripper_state))
    result = np.vstack((result, row))
    return result
```

Figure 9. The Python 3.7 code for generating a path segment. It takes X_{start} -starting configuration, X_{end} – end configuration, total duration of segment t in sec., method – 3 for cubic time scaling and 5 for quantic time scaling, gripper_state – 0 or 1 open or closed respectively, type – type of trajectory to generate.



```python
def TrajectoryGenerator():
    # Generates a trajectory from start to finish. Note: all Transformation matrices are specified in {s} frame
    TseInitial = np.asfarray([[1, 0, 0, 0.033], [0, 1, 0, 0], [0, 0, 1, 0.6546], [0, 0, 0, 1]])
    # initial cube position
    cos = math.cos(1)
    sin = math.sin(1)
    Tscin = np.asfarray([[cos, -sin, 0, 1], [sin, cos, 0, -0.5], [0, 0, 1, 0], [0, 0, 0, 1]])
    # Final cube position
    TscFinal = np.asfarray([[1, 0, 0, -1], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
    # Standoff relative to cube and environment
    TceStandoff = np.asfarray([[-1, 0, 0, 0], [0, 1, 0, 0], [0, 0, -1, 0.09], [0, 0, 0, 1]])
    TseStandoffin = np.dot(Tscin, TceStandoff)
    TseStandoffend = np.dot(TscFinal, TceStandoff)
    # Grasp  relative to cube and environment
    TceGrasp = np.asfarray([[-1, 0, 0, 0], [0, 1, 0, 0], [0, 0, -1, 0.01], [0, 0, 0, 1]])
    TseGraspin = np.dot(Tscin, TceGrasp)
    TseGraspend = np.dot(TscFinal, TceGrasp)
    k = 1
    method = 5
    traj1 = AddSegmentTrajectory(TseInitial, TseStandoffin, 5, k, method, 0, 0)
    traj2 = AddSegmentTrajectory(TseStandoffin, TseGraspin, 1, k, method, 0)
    traj3 = AddSegmentTrajectory(TseGraspin, TseGraspin, 1, k, method, 1)
    traj4 = AddSegmentTrajectory(TseGraspin, TseStandoffin, 1, k, method, 1)
    traj5 = AddSegmentTrajectory(TseStandoffin, TseStandoffend, 8, k, method, 1)
    traj6 = AddSegmentTrajectory(TseStandoffend, TseGraspend, 1, k, method, 1)
    traj7 = AddSegmentTrajectory(TseGraspend, TseGraspend, 1, k, method, 0)
    traj8 = AddSegmentTrajectory(TseGraspend, TseStandoffend, 1, k, method, 0)
    traj = np.vstack((traj1, traj2, traj3, traj4, traj5, traj6, traj7, traj8))
    return traj
```

Figure 10. Function in Python 3.7 that generates the complete trajectory.

The code in Figure 10 outputs a representation of the *N* configurations of the end-effector along the entire concatenated eight-segment reference trajectory. Each of these *N* reference points represents a transformation matrix T_{se} of the end-effector frame {e} relative to {s} at an instant in time, plus the gripper state (0 or 1). For example, if the entire eight-segment trajectory takes 30 seconds, for example, the program results in approximately $N \approx 30k/0.01$ sequential reference configurations each separated by 0.01 / *k* seconds in time. These reference configurations will be used by controller. The representation of the reference configurations is an $N\ x$ 13 matrix, for example, each of the *N* rows would represent a configuration of the end-effector frame {e} relative to {s} at that instant in time. Twelve of the 13 entries of a matrix row are the top three rows of the transformation matrix T_{se} at that instant of time, i.e., *r*11, *r*12, *r*13, *r*21, *r*22, *r*23, *r*31, *r*32, *r*33, *px*, *py*, *pz* from

$$T_{se} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calculating position at the next time step

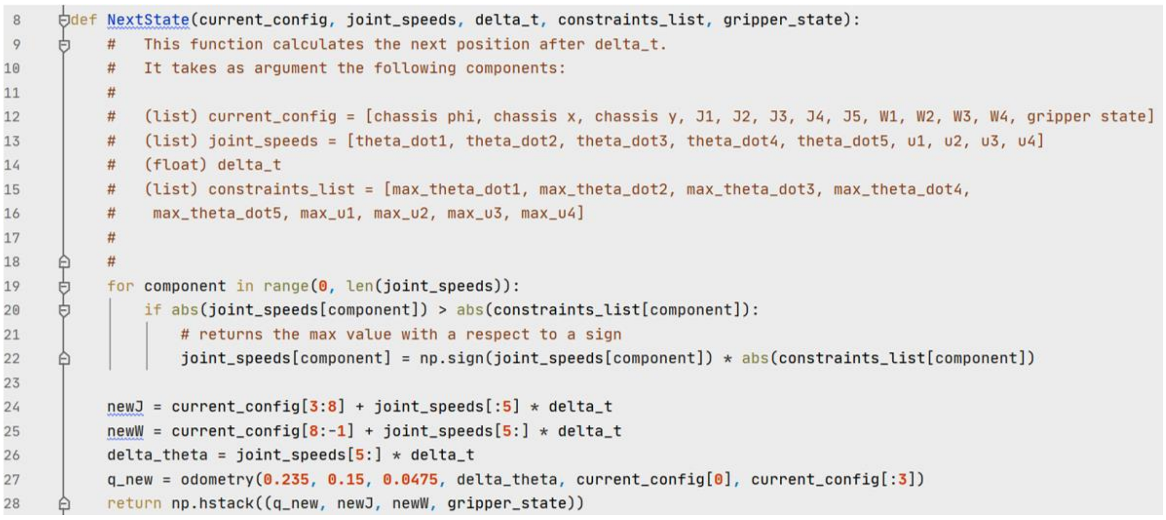
In this work author used a simple first-order Euler step, i.e.,

- Arm joint angles after a small timestep Δt = (previous angles) + (joints speed) Δt
- Wheels angles after a small timestep Δt = (previous wheel angles) + (wheels speed) Δt
- New chassis configuration is obtained from odometry (see Omnidirectional wheels)

The following function takes as an input:

- A 12-vector representing the current configuration of the robot (3 variables for the chassis configuration, 5 variables for the arm configuration, and 4 variables for the wheel angles).
- A 9-vector of controls indicating the arm joint speeds $\dot{\theta}$ (5 variables) and the wheel speeds u (4 variables).
- A small timestep $\Delta t = 0.01$
- A positive real value indicating the maximum angular speed of the arm joints and the wheels. For example, if this value is 12.3, the angular speed of the wheels and arm joints is limited to the range [-12.3 radians/s, 12.3 radians/s]. Any speed in the 9-vector

of controls that is outside this range will be set to the nearest boundary of the range. In this work every joint limit can be specified.



```python
def NextState(current_config, joint_speeds, delta_t, constraints_list, gripper_state):
    #   This function calculates the next position after delta_t.
    #   It takes as argument the following components:
    #
    #   (list) current_config = [chassis phi, chassis x, chassis y, J1, J2, J3, J4, J5, W1, W2, W3, W4, gripper state]
    #   (list) joint_speeds = [theta_dot1, theta_dot2, theta_dot3, theta_dot4, theta_dot5, u1, u2, u3, u4]
    #   (float) delta_t
    #   (list) constraints_list = [max_theta_dot1, max_theta_dot2, max_theta_dot3, max_theta_dot4,
    #    max_theta_dot5, max_u1, max_u2, max_u3, max_u4]
    #
    #
    for component in range(0, len(joint_speeds)):
        if abs(joint_speeds[component]) > abs(constraints_list[component]):
            # returns the max value with a respect to a sign
            joint_speeds[component] = np.sign(joint_speeds[component]) * abs(constraints_list[component])

    newJ = current_config[3:8] + joint_speeds[:5] * delta_t
    newW = current_config[8:-1] + joint_speeds[5:] * delta_t
    delta_theta = joint_speeds[5:] * delta_t
    q_new = odometry(0.235, 0.15, 0.0475, delta_theta, current_config[0], current_config[:3])
    return np.hstack((q_new, newJ, newW, gripper_state))
```

Figure 11. Python 3.7 code for calculating next state according to single first-order Euler step

3.9. Feedforward and feedback control

A desired joint trajectory $\theta_d(t)$ can be commanded directly resulting in a simplest type of control for commanded velocity $\dot{\theta}(t)$:

$$\dot{\theta}(t) = \dot{\theta}_d(t)$$

Such form is referred to as feedforward or open-loop controller, since no feedback (sensor data) is provided.

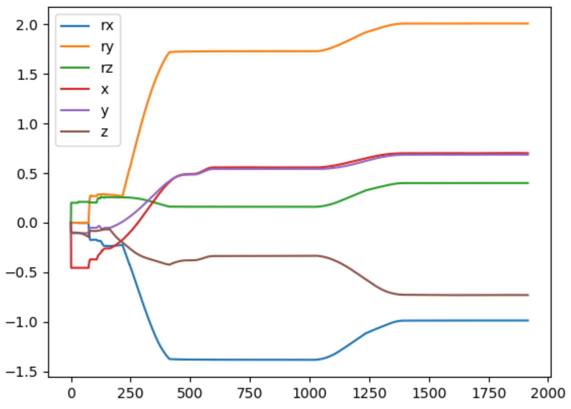


Figure 12. Feedforward control error for every component of commanded twist. The final configuration is not met

In real life, however, it is useful and most common to use some sorts of sensors to measure actual turn angles, speeds, etc. And with such data feedback controller is possible.

The simplest feedback controller is a P control and First-Order Error Dynamics:

$$\dot{\theta}(t) = K_p\big(\theta_d(t) - \theta(t)\big) = K_p\theta_e(t)$$

Where $K_p > 0$ is a control gain and acts as a sort of virtual spring that drives each link and joint of the robot to the desired position.

$\theta_e(t)$ is a position error, that has its maximum value at the moment when new desired position was commanded and gradually decreases as robot moves to it.

The P controller represents class of linear controllers because it constructs signal that is a linear combination of the error $\theta_e(t)$ and possibly its time derivatives and time integrals.

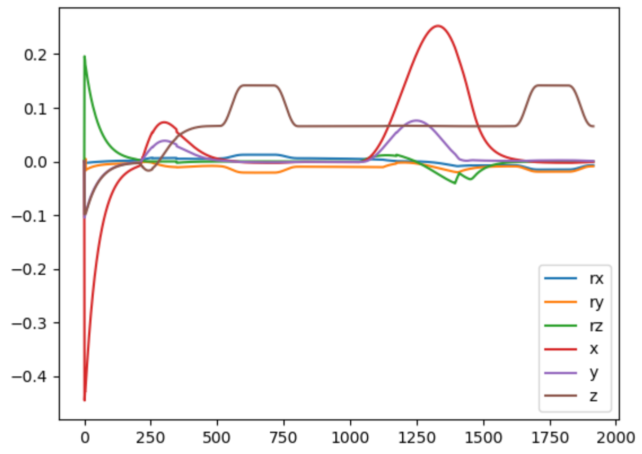
If $\theta_d(t)$ is viewed in a small period of time it can be treated as constant, i.e. $\dot{\theta}_d(t) = 0$, resulting in a setpoint control. In setpoint control, the error dynamics:

$$\dot{\theta}_e(t) = -\dot{\theta}(t)$$

And in a form of P controller:

$$\dot{\theta}_e(t) + K_p\theta_e(t) = 0$$

The 2% settling time – the time, during which signal comes and remains in a 2% range of desired level, is $4/K_p$. A larger $K_p$means faster response and shorter settling time, however, in real life it cannot be infinitely big because of mechanical and electrical



properties of actuators and controllers.

Figure 13. P Control with initial error. Final configuration is almost met

3.10. PI Control and Second-Order Error dynamics

An alternative to using a large gain $K_p$and simple P controller is to introduce another term in the control law. A proportional-integral controller, or simply PI controller, adds a term to the whole system that is proportional to the time-integral of the error:

$$\dot{\theta}(t) = K_p\theta_e(t) + K_i\int_0^t \theta_e(t)\,dt$$

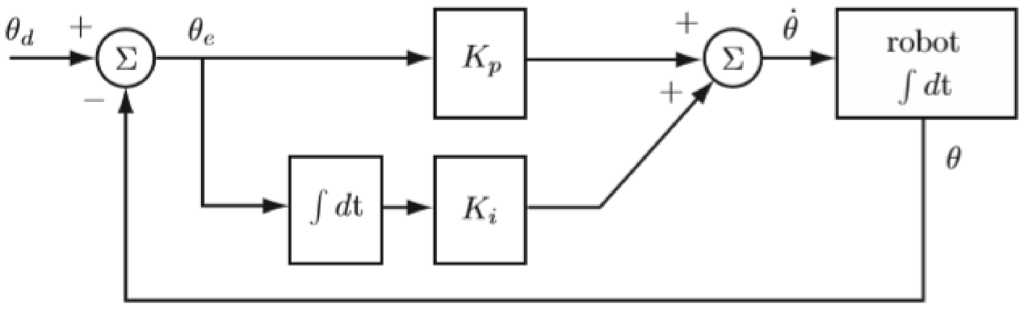


Figure 14. The block diagram of a PI controller that produces a commanded velocity $\dot{\theta}(t)$ as input to the robot.

Taking the time derivative:

$$\ddot{\theta}(t) + K_p\dot{\theta}_e(t) + K_i\theta_e(t) = 0$$

The standard second-order form equation is:

$$\ddot{\theta}_e(t) + 2\zeta\omega_n\dot{\theta}_e(t) + \omega_n^2\theta_e(t) = 0$$

Where ω_n is called a natural frequency and ζ is called the damping ratio. In this case $\omega_n = \sqrt{K_i}$ and $\zeta = K_p/2\sqrt{K_i}$. ω_n and ζ play critical role in the system behavior. Depending on ζ (>1, =1, <1) roots of the second-order form equation can be real and unequal, real and equal, or complex conjugates respectively. Resulting in

- Overdamped
- Critically damped
- Underdamped

outcomes.

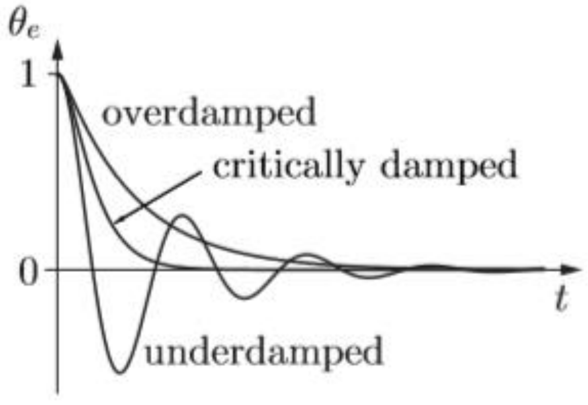


Figure 15. Different ζ results in different outcomes

A good rule is to choose K_p and K_i accordingly, so $K_i = K_p^2/4$, this would result in critical damping without overshoot. In practice these constants have some limits, so they should be chosen to yield critical damping.

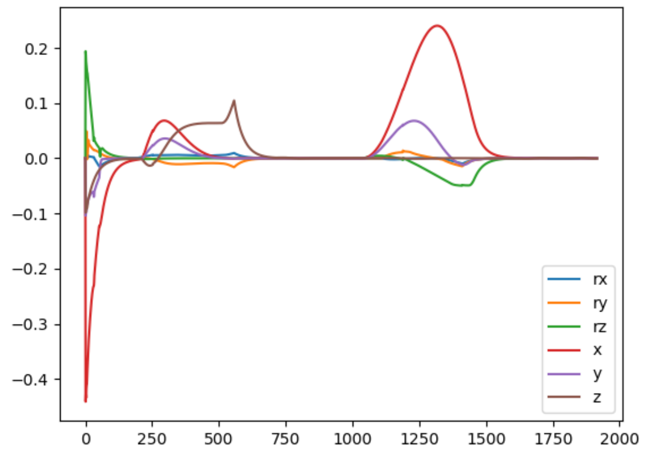


Figure 16. PI control. With initial error final configuration was met precisely

One drawback of any feedback control is that an error is required before the joint begins to move. It would be preferable to use our knowledge of the desired trajectory $\theta_d(t)$ to initiate motion before any error accumulates. We can combine the advantages of feedforward control, which commands motion even when there is no error, with the advantages of feedback control, which limits the accumulation of error, as follows:

$$\dot{\theta}(t) = \dot{\theta}_d(t) + K_p\theta_e(t) + K_i\int_0^t \theta_e(t)\,dt$$

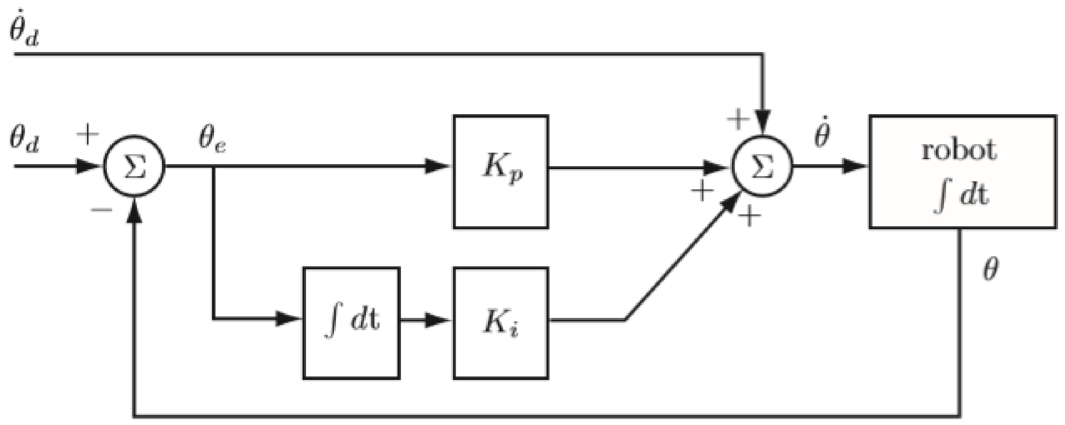


Figure 17. The block diagram of a Feedforward plus PI feedback controller that produces a commanded velocity $\dot{\theta}(t)$ as input to the robot.

The feedforward plus feedback control law can be expressed in task space.

Let $X(t) \in SE(3)$ be an end-effector configuration as a function of time, $V_b(t)$ be the end-effector twist, expressed in the {b} frame. Desired configuration $X_d(t)$ and $[V_d] = X_d^{-1}\dot{X}_d$. A task space version of the control law is:

$$V_b(t) = \left[Ad_{X^{-1}X_d}\right]V_d(t) + K_p X_e(t) + K_i \int_0^t X_e(t)\,dt$$

$\left[Ad_{X^{-1}X_d}\right]$ is a skew-symmetric matrix, used to change a reference frame for $V_d(t)$, so it would be in actual end-effector frame X, rather than in desired frame X_d. In general, adjoint representation $[Ad_T]$ is structured as follows:

$$[Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \in \mathbb{R}^{6\times6}$$

$V_d(t)$ is a twist, that takes X_d to $X_{d,next}$, if followed for Δt:

$$V_d(t) = (1/\Delta t)log\,(X_d^{-1}X_{d,next})$$

X_e – is a twist, which takes X to X_d if followed for unit time. So

$$[X_e] = log\,(X^{-1}X_d)$$

The K_p and K_i are no longer a number, but rather a diagonal matrix, with diagonal elements equal K_p and K_i respectively.

Commanded wheel and arm joint speeds $(u, \dot{\theta})$:

$$\begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = J_e^{\dagger} V_b$$

$J_e^{\dagger}$ is a pseudoinverse of manipulator Jacobian – a horizontally stacked wheel and arm Jacobians. Because wheel and manipulator Jacobians are treated as a union, desired twist will be executed by a simultaneous motion of both platform and arm. Such configuration is more mobile than “decoupled” analogs.

As every mobile platform has matrix F that maps platform twist to individual wheels speed.

$$V_b = Fu$$

To create a six-dimensional twist V_{b6} corresponding to the planar twist V_b, the 6 x m matrix was defined:

$$F_6 = \begin{bmatrix} 0_m \\ 0_m \\ F \\ 0_m \end{bmatrix}$$

where two rows of m zeros are stacked above F and one row is situated below it. This is because platform only moves in a plane.

$$V_{b6} = F_6 u$$

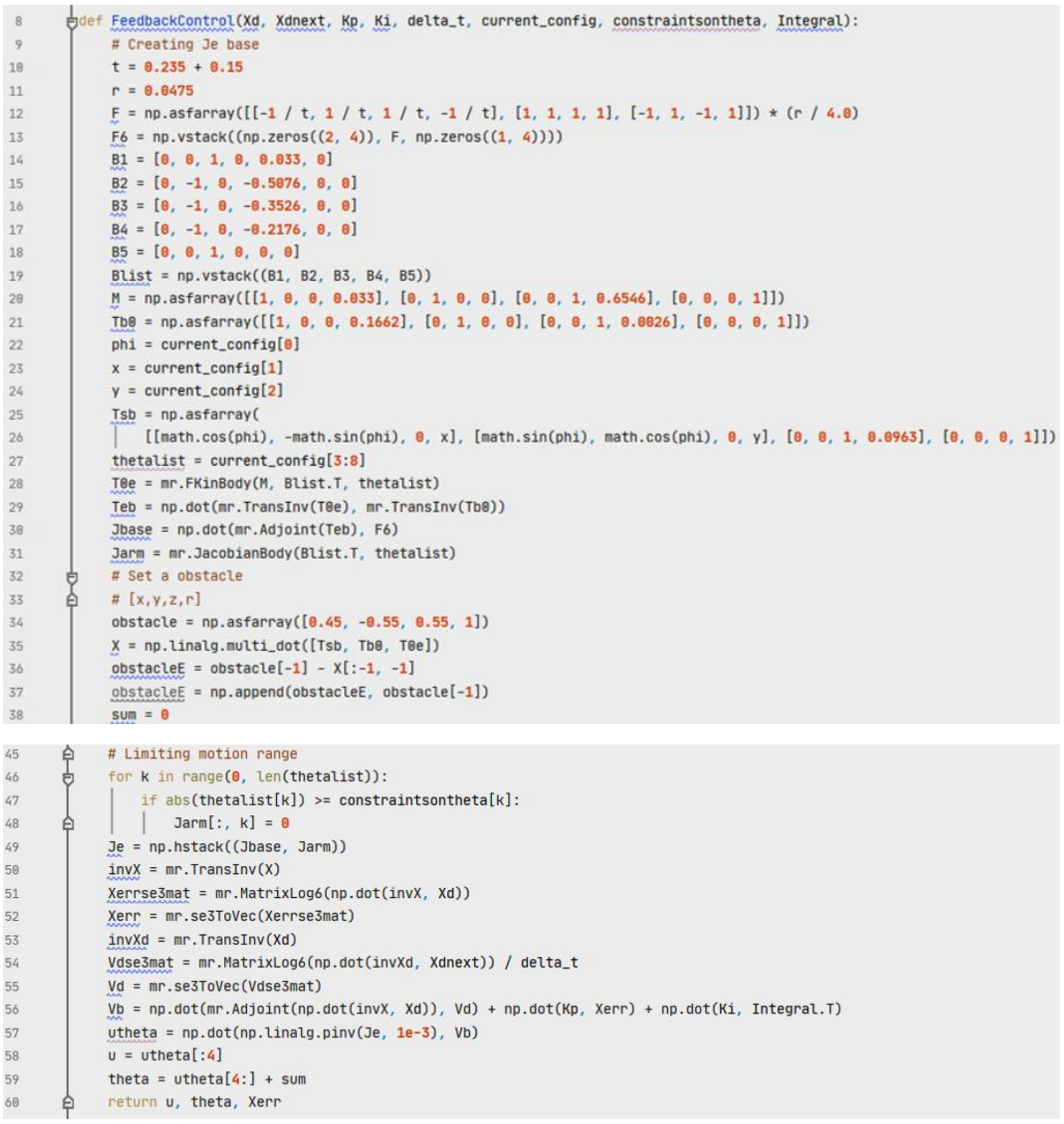
This chassis twist can be expressed in the end-effector frame as

$$\left[Ad_{T_{eb}(\theta)}\right]V_{b6} = \left[Ad_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}\right]V_{b6} = \left[Ad_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}\right]F_6 u = J_{base}(\theta)u$$

Therefore:

$$J_{base}(\theta) = \left[Ad_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}\right]F_6$$

The resulting code:



```
8   def FeedbackControl(Xd, Xdnext, Kp, Ki, delta_t, current_config, constraintsontheta, Integral):
9       # Creating Je base
10      t = 0.235 + 0.15
11      r = 0.0475
12      F = np.asfarray([[-1 / t, 1 / t, 1 / t, -1 / t], [1, 1, 1, 1], [-1, 1, -1, 1]]) * (r / 4.0)
13      F6 = np.vstack((np.zeros((2, 4)), F, np.zeros((1, 4))))
14      B1 = [0, 0, 1, 0, 0.033, 0]
15      B2 = [0, -1, 0, -0.5076, 0, 0]
16      B3 = [0, -1, 0, -0.3526, 0, 0]
17      B4 = [0, -1, 0, -0.2176, 0, 0]
18      B5 = [0, 0, 1, 0, 0, 0]
19      Blist = np.vstack((B1, B2, B3, B4, B5))
20      M = np.asfarray([[1, 0, 0, 0.033], [0, 1, 0, 0], [0, 0, 1, 0.6546], [0, 0, 0, 1]])
21      Tb0 = np.asfarray([[1, 0, 0, 0.1662], [0, 1, 0, 0], [0, 0, 1, 0.0026], [0, 0, 0, 1]])
22      phi = current_config[0]
23      x = current_config[1]
24      y = current_config[2]
25      Tsb = np.asfarray(
26          [[math.cos(phi), -math.sin(phi), 0, x], [math.sin(phi), math.cos(phi), 0, y], [0, 0, 1, 0.0963], [0, 0, 0, 1]])
27      thetalist = current_config[3:8]
28      T0e = mr.FKinBody(M, Blist.T, thetalist)
29      Teb = np.dot(mr.TransInv(T0e), mr.TransInv(Tb0))
30      Jbase = np.dot(mr.Adjoint(Teb), F6)
31      Jarm = mr.JacobianBody(Blist.T, thetalist)
32      # Set a obstacle
33      # [x,y,z,r]
34      obstacle = np.asfarray([0.45, -0.55, 0.55, 1])
35      X = np.linalg.multi_dot([Tsb, Tb0, T0e])
36      obstacleE = obstacle[-1] - X[:-1, -1]
37      obstacleE = np.append(obstacleE, obstacle[-1])
38      sum = 0
```

```
45      # Limiting motion range
46      for k in range(0, len(thetalist)):
47          if abs(thetalist[k]) >= constraintsontheta[k]:
48              Jarm[:, k] = 0
49      Je = np.hstack((Jbase, Jarm))
50      invX = mr.TransInv(X)
51      Xerrse3mat = mr.MatrixLog6(np.dot(invX, Xd))
52      Xerr = mr.se3ToVec(Xerrse3mat)
53      invXd = mr.TransInv(Xd)
54      Vdse3mat = mr.MatrixLog6(np.dot(invXd, Xdnext)) / delta_t
55      Vd = mr.se3ToVec(Vdse3mat)
56      Vb = np.dot(mr.Adjoint(np.dot(invX, Xd)), Vd) + np.dot(Kp, Xerr) + np.dot(Ki, Integral.T)
57      utheta = np.dot(np.linalg.pinv(Je, 1e-3), Vb)
58      u = utheta[:4]
59      theta = utheta[4:] + sum
60      return u, theta, Xerr
```

Figure 18. Python 3.7 code for feedforward and feedback, considering self-intersection.

3.11. Avoiding self-intersection

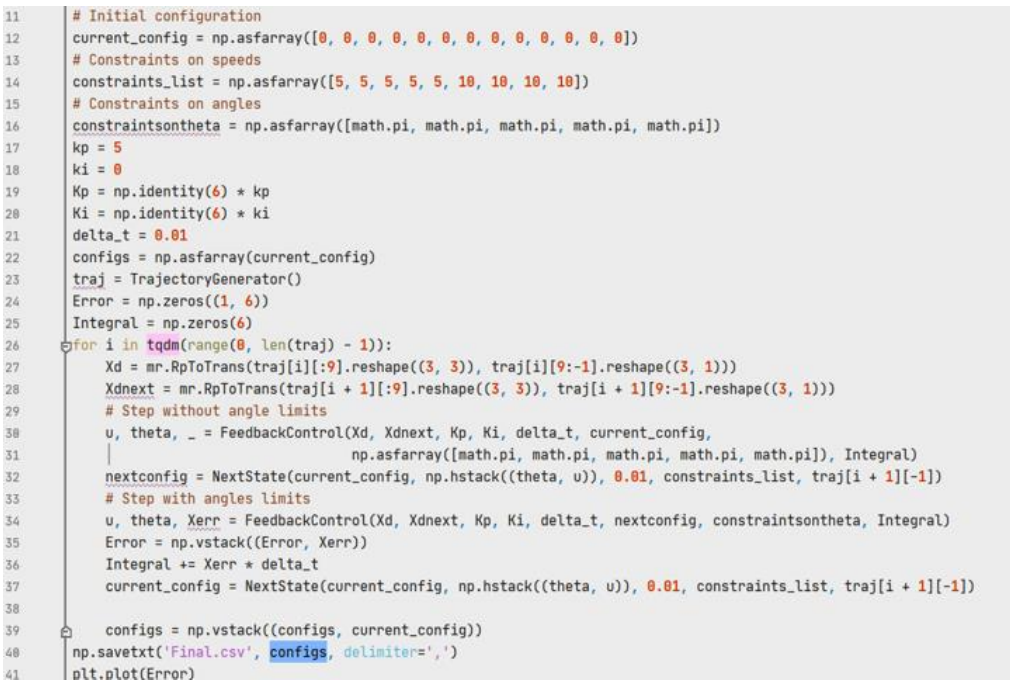
In this work problem of generating a trajectory and subsequently following it without self-intersection was addressed. The algorithm is rather simple, yet effective. It incorporates setting boundary conditions on every joint angle of rotation and checking it at every timestep. Whenever angle, desired by the feedforward law, violates these limits according column in Jacobian matrix is set to zero, thus no movement of this joint further affects the end-effector movement. This issue can be seen at rows 46-48 in Figure 18.

As mentioned before, this method is simple and comes at cost of preventing the manipulator to perform some movements. It is still capable of vast variety of tasks and working space, however more complex method with point clouds and Oriented Bounding Boxes was addressed in [2]. This will be incorporated in the future versions of this program, since it requires great amount of knowledge, time, and computational power.

3.12. Final program

In the final part of the program all previous pieces of code work together to create simulation in CoopeliaSim software, which is used for visualization. One particular note is that program at each time step runs function $NextState$ twice in order to check if at the next timestep joint limits, dictated by self-intersection check, will be violated.

The resulting code:



```python
# Initial configuration
current_config = np.asfarray([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
# Constraints on speeds
constraints_list = np.asfarray([5, 5, 5, 5, 5, 10, 10, 10, 10])
# Constraints on angles
constraintsontheta = np.asfarray([math.pi, math.pi, math.pi, math.pi, math.pi])
kp = 5
ki = 0
Kp = np.identity(6) * kp
Ki = np.identity(6) * ki
delta_t = 0.01
configs = np.asfarray(current_config)
traj = TrajectoryGenerator()
Error = np.zeros((1, 6))
Integral = np.zeros(6)
for i in tqdm(range(0, len(traj) - 1)):
    Xd = mr.RpToTrans(traj[i][:9].reshape((3, 3)), traj[i][9:-1].reshape((3, 1)))
    Xdnext = mr.RpToTrans(traj[i + 1][:9].reshape((3, 3)), traj[i + 1][9:-1].reshape((3, 1)))
    # Step without angle limits
    u, theta, _ = FeedbackControl(Xd, Xdnext, Kp, Ki, delta_t, current_config,
                                  np.asfarray([math.pi, math.pi, math.pi, math.pi, math.pi]), Integral)
    nextconfig = NextState(current_config, np.hstack((theta, u)), 0.01, constraints_list, traj[i + 1][-1])
    # Step with angles limits
    u, theta, Xerr = FeedbackControl(Xd, Xdnext, Kp, Ki, delta_t, nextconfig, constraintsontheta, Integral)
    Error = np.vstack((Error, Xerr))
    Integral += Xerr * delta_t
    current_config = NextState(current_config, np.hstack((theta, u)), 0.01, constraints_list, traj[i + 1][-1])

    configs = np.vstack((configs, current_config))
np.savetxt('Final.csv', configs, delimiter=',')
plt.plot(Error)
```

Figure 19. The final part in Python 3.7. The outcome of each timestep is a 13-vector: chassis phi, chassis x, chassis y, J1, J2, J3, J4, J5, W1, W2, W3, W4, gripper state

2. Stepper motor calculation

For the calculation purposes a stepper motor was chosen, as the most crucial and relied-upon component in industrial robots. It is very important to pick the right one, because otherwise robot may fail to execute certain motions within appropriate tolerance or may even fail to move completely.

Stepper motors require a drive – a device that converts specified turning angle to sets of electric impulses, which go directly to the motor. The driver and motor can be connected in different ways, according to the number of input pins to the motor.

1. The easiest option with a 4-wire stepper motor. The main thing here is to correctly connect the conclusions of the A + motor with the A + driver, the A- motor with the A- driver, and so on.

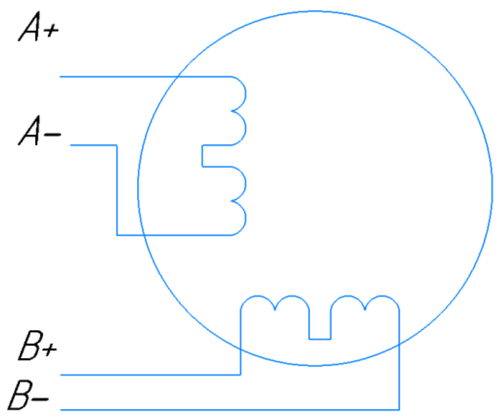


Figure 20. 4-wire stepper motor connection

2. Next comes the 8-wire engine. It is characterized by two connection options.

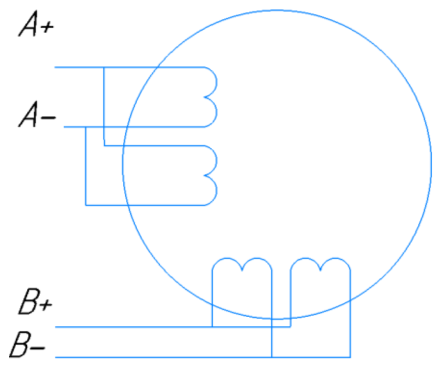


Figure 21. 8-wire engine with parallel connection

This is a parallel connection of the stepper windings. With this connection, the total inductance of the windings is reduced, which allows to increase the maximum shaft rotation speed. The magnitude of the inductance of the windings affects the frequency characteristics of the motor, especially at high frequencies of the control signals. It is worth striving for such a connection if the high speed of the step-changer is really important and the accuracy and efficiency at high speeds are critical.

With serial connection (see Figure 20), the engine will behave like a normal 4-wire.

3. The connection in Figure 22 allows to reduce the inductance and thereby improve the quality of the engine at high frequencies (revolutions). But at the same time, the

efficiency of the engine and its power are reduced, and the control current is increased. I would advise such an inclusion option only for temporary high-speed operations that do not require frequent braking and acceleration, for example, during the return of the 3D printer carriage. At the same time, a mechanism is required to automatically switch the engine from full-winding to semi-winding.

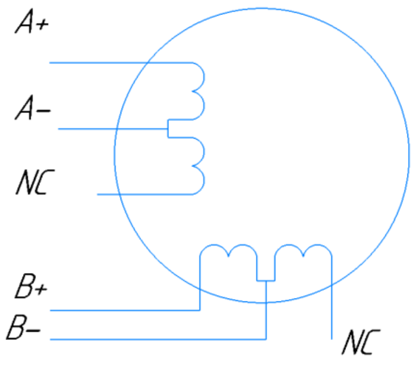


Figure 22. 6-wire engine with semi-winding regime

And the second option to turn on the 6-wire stepper motor is as follows. The middle pins of each winding are simply not involved, and the step-changer works exactly like a 4-wire:

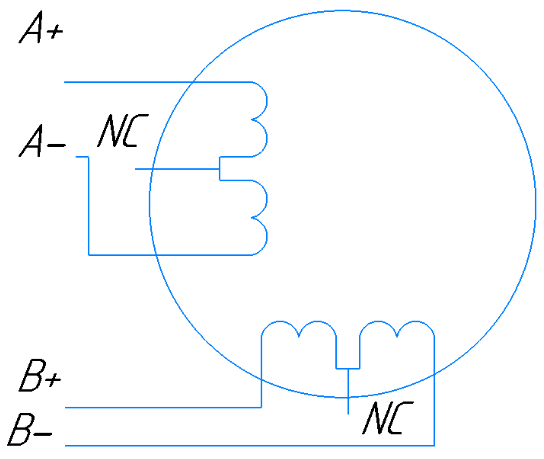


Figure 23. 6-wire engine connected to resemble 4-wire engine

The maximum speed $Speed_{max}$, minimum time per step t_{step}, and power P are important markers to motor performance, as they show how fast will the motor turn, and how much power it requires.

The input values for the motor are:

- Nominal Voltage (V): 24V DC
- Nominal Current (A): 2.32 A
- Terminal Inductance (L): 0.573 mH
- Counts per Revolution (cpr): 4000

$$Speed_{max} = \frac{V}{2LI \cdot cpr} = \frac{24}{2 \cdot 0.573 \cdot 2.32 \cdot 4000} = 2.26\, rev/sec$$

$$t_{step} = \frac{2LI}{V} = \frac{2 \cdot 0.573 \cdot 2.32}{24} = 0.11\, msec$$

$$P = IV = 55.7\, W$$

3. Hydraulic system calculation

This project is being designed to handle number of different tasks in lab or workshop. Because only robot is insufficient to solely perform precision tasks some sort of grasp-and-fix device is required. As for example, consider a motorcycle workshop, where the bike has to be fixed properly without slipping, in order for robot to perform tasks on it. The cycle of the main hydraulic cylinder allows you to safely move heavy objects, as motorcycle for example up and down, and at the same time 4 additional cylinders allocated on the boundary of the ramp ensure firm grip. At an operating pressure of 4 MPa from able to create a useful force of more than 23 kN.

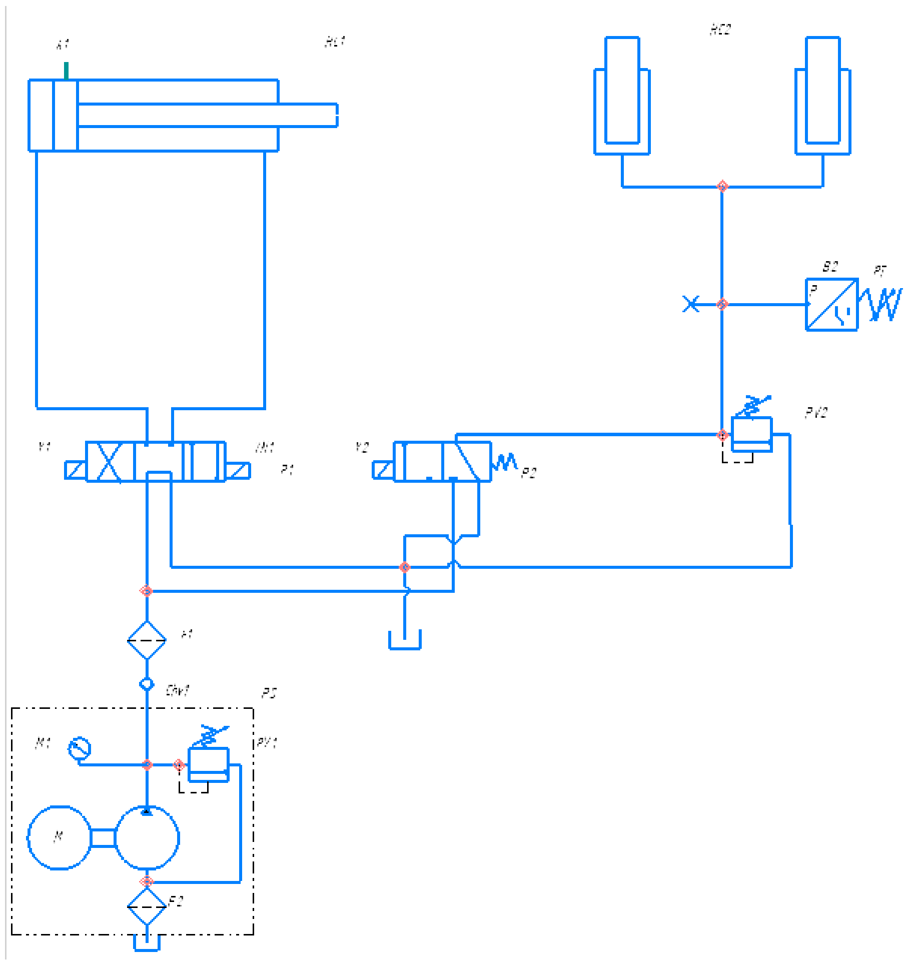


Figure 24. Hydraulic scheme

This build works as follows:

1. Set Y2 – 2 cylinders clamp the object or motorcycle.
2. Only when pressure control B2 fires, the Y1 and YN1 commands can be executed.
3. Operator can arbitrarily set Y1 or YN1 to move object up or down.
4. Only when Cylinder 1 is at home position A1 - Y2 can be dismissed.

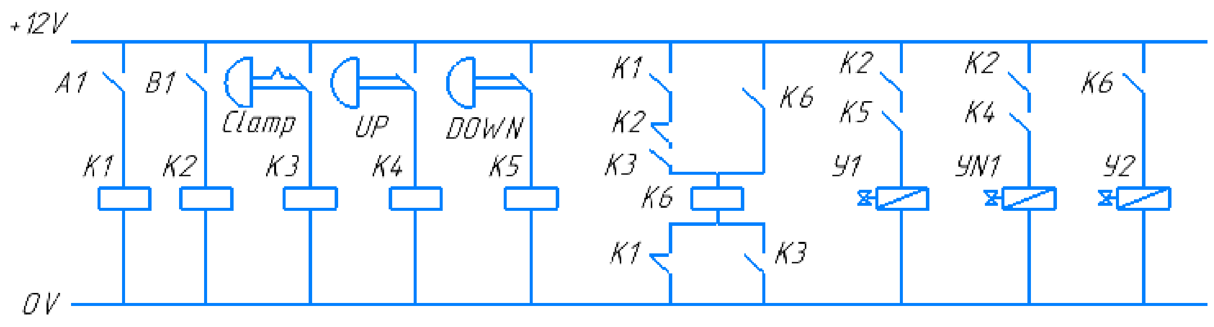


Figure 25. Relay control scheme. Three physical buttons are connected to the electric scheme, and that solenoids are powered according to the rules.

					DP MA61206. 00.000 EN	Page
						33
Ed.	Page	№ doc.	Sign	Date		

3.1 Choice of working liquid

Hydraulic oil used in machine hydraulics should have the following characteristics: stable viscosity over a wide temperature range, good wear resistance, high demulsifying and antifoam properties to contain additives that improve lubrication, anti-corrosion and antioxidant properties.

The author chooses the working fluid HLP 32 which provides high operational characteristics, meets the requirements of the main manufacturers of hydraulic equipment. Made on the basis of high-purity mineral oils. It is widely used in machine hydraulics (cranes, manipulators, presses) and in mobile hydraulics.

Benefits of HLP 32

- high anti-wear properties
- good thermal and oxidative stability
- low tendency to sediment formation
- low susceptibility to foaming, rapid air separation
- good filtering

HLP 32 general purpose hydraulic oil. It is used in hydraulic equipment operating in severe operating conditions, at moderately high operating temperatures. HLP 32 is suitable for all types of hydraulic pumps. This oil is mainly used in hydrostatic systems of earlier production systems, where there may be a problem of oil leakage, and where complete protection of the equipment is required. HLP 46 can be used in equipment where an oil with anti-corrosion, antioxidant and / or anti-wear properties is required.

Specifications and approvals

- Approved by Denison HF0,1,2;
- Cincinnati Machine P-68, P-70, P-69;
- Eaton Vickers 35VQ25; Bosch Rexroth 90220.
- Meets the requirements of DIN 51524.

Fluid properties

Property	Typical value

Density at 20 ° C [g / cm³]	0,870
Kinematic viscosity at 40 ° C [mm² / s]	32,29
Kinematic viscosity at 100 ° C [mm² / s]	5,43
Viscosity index	104
Flash point (Cleveland) [° C]	225
Freezing point [° C]	-30
Density at 20 ° C (68 ° F) g / cm3	0.870
Acid number, mg KOH per 1g of oil, no more than	0,7
Mass fraction of sulfur,%, not more than	0,5
Mass fraction of mechanical impurities	absent
Ash content, not more than	0,2
Prone to foaming, cm^3, not more than	
at 24 ° C, sequence I	50
at 94 ° C, sequence II	20
Corrosion of metals in lubricant	Withstands

Table 1. Properties of HLP 32

Instructions for storage and transportation

Fire hazard: Class IV.

Recommended storage temperature: max. 40 ° C

3.2 Hydraulic cylinder calculation

Determining the size for main cylinder 1:

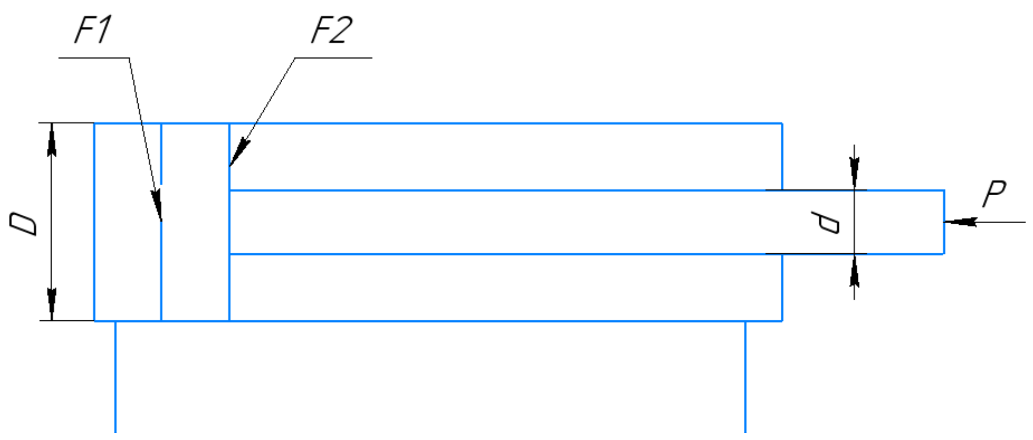


Figure 26. Hydraulic cylinder scheme

For hydraulic cylinder with one-way rod, working on compression during extension:

$$D = \sqrt{\frac{4\,P}{\pi\left(p_1 - \frac{p_1}{\psi}\right)\mu_m}} = \sqrt{\frac{4 * 23 * 10^3}{3{,}14\left(4 * 10^6 - \frac{0{,}4 * 10^6}{1{,}33}\right)0{,}93}} \approx 0{,}092\,m$$

So, we choose the closest from the standard values: D = 100mm

Rod diameter:

$$d = D\sqrt{1 - 1/\psi} = 100 * 0{,}5 = 50\,mm$$

Determine the diameter of the supply pipelines:

$$Q = f_п * V = 0{,}25\pi D^2 * \frac{3}{60} = 4 * 10^{-4}\,m^3/s$$

$$d_п = \sqrt{\frac{4Q}{\pi V_п}} = \sqrt{\frac{4 * 4 * 10^{-4}}{3{,}14 * 5}} \approx 0{,}01\,m = 10\,mm$$

Which coincides with the standard value of 10 mm, which we choose.

Determining the size of the auxiliary cylinders:

Since we have 4 auxiliary cylinders, then, accordingly, the load on each of them will be equal to 0.25P, and the total consumption of 4 times bigger.

For hydraulic cylinder with one-way rod, working on compression during extension:

$$D = \sqrt{\frac{4\,P}{\pi\left(p_1 - \frac{p_1}{\psi}\right)\mu_m}} = \sqrt{\frac{4 * 1{,}25 * 10^3}{3{,}14\left(3{,}5 * 10^6 - \frac{0{,}4 * 10^6}{1{,}33}\right)0{,}93}} \approx 0{,}022\,m$$

So we choose the closest from the standard values: D = 20mm.

Rod diameter:

$$d = D\sqrt{1 - 1/\psi} = 20 * 0{,}5 = 10\, mm$$

Determine the diameter of the supply pipelines:

$$Q = f_{п} * V_d = 0{,}25\pi D^2 * \frac{l_d}{t_d} = 0{,}25 * 3{,}14 * 0{,}022^2 * \frac{0{,}2}{7} * 4 \approx 0{,}4 * 10^{-4}\, m^3/c$$

$$d_{п} = \sqrt{\frac{4Q}{\pi V_{п}}} = \sqrt{\frac{4 * 0{,}4 * 10^{-4}}{3{,}14 * 5}} \approx 0{,}003\, m = 3\, mm$$

From the standard we choose the closest d = 6 mm.

3.3 Calculation of hydraulic cylinder for durability

Calculation of wall thickness

To calculate thin-walled cylinders and pipes (S / D <0,1) we use the following formula:

$$S = \frac{pD}{2\sigma_{max}}$$

The author has chosen Steel40X as a material for his hydro cylinders, so σ_{max} with normalization, variable load and tensile = 315 MPa. So:

$$S_{min} = \frac{4.5 * \ 100}{2 * 315} = 0.8\, mm$$

According to GOST 8732-78, as well as the presence of pipes at manufacturers, we choose 108 x 4 mm; S = 4mm.

3.4 Calculation of threaded connections

Tensile stresses for the threaded rod:

$$\sigma = \frac{4k_tP}{\pi\left(d_0 - 1,2 * S_p\right)^2 z} = \frac{4 * 3 * 23000}{3,14 * (0,108 - 1,2 * 0,002)^2 * 2} = \frac{276000}{0,071} \approx 3,9\ MPa$$

Where:

k_t - tightening factor (for variable load we choose = 3)

d_0 - outer thread diameter

S_p - thread pitch, according to GOST 24705-81 choose = 2 mm

z - the number of threaded connections that bear the load

Tangential stresses in the thread:

$$\tau = \frac{k_tPd_0k_1}{0,2\left(d_0 - 1,2 * S_p\right)^3} = \frac{3 * 23000 * 0,108 * 0,12}{0,2(0,108 - 1,2 * 0,002)^3} = \frac{894,24}{0,00024} \approx 3,7\ MPa$$

The given tension in a thread:

$$\sigma_{giv} = \sqrt{\sigma^2 + 3\tau^2} = \sqrt{15,21 + 41,07} = 7,5\ MPa$$

For Steel 40X σ_t = 315 MPa

Safe ratio for plastic deformations:

$$n = \frac{\sigma_t}{\sigma_{giv}} = \frac{315}{7,5} = 42 \gg 2,5$$

3.5 Calculation of fastening of covers on external semirings

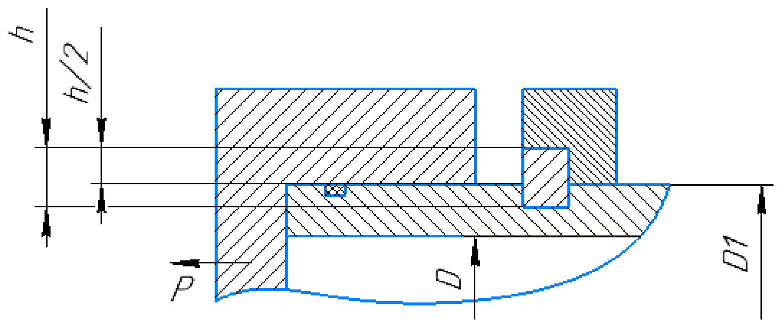


Figure 27. Scheme of semirings connection

Semi-rings are designed for cutting and crumpling:

$$\tau_{cut} = \frac{\text{P}}{F_{3\text{P}}} \leq [\tau_{cut}]$$

$$\sigma_{crump} = \frac{\text{P}}{F_{3\text{M}}} \leq [\sigma_{crump}]$$

Cut and crumple areas:

Fcut = πD1l

Fcrump = 0.25* π(2D1h-h2)

Force acting on the lids:

Blind: P = 0.25πD2p

Rod: P’ = 0.25π(D2 – d2)p

p, Pa	*4500000*			***[τcut], MPa***	*95*
l, m	*0.005*			***[σcrump], MPa***	*230*
h, m	*0.005*				
D, m	*0.1*				
D1, m	*0.108*				
d, m	*0.05*				
				Blind	
F cut, m2	*0.001696*			***τcut, MPa***	*20.83*
Fcrump, m2	*0.000828*			***σcrump, MPa***	*42.65*
Pblind, H	*35325*			***Rod***	
Prod, H	*26493.75*			***τcut, MPa***	*15.63*
				σcrump, MPa	*31.99*

Figure 28. Resulting forces and pressures

After analyzing the data, we recognize that this thickness of 5 mm and the material Steel 40X is enough.

4. Part manufacturing

The part for manufacturing was chosen to be a holding or a fixture plate – the basic standardized plate for mounting gripper on manipulators end. For the blueprint, the Universal Robots standard plate was chosen.

A stock is an aluminum (D16T) cylinder $\emptyset\, 50 \times 40\, mm$. It yields both sufficient for its purpose's strength and extremely amenable to machining.

As a CNC machine 5 axis stand was chosen because of great versatility, commonness, and convenience for uninterrupted process. As such HAAS UMC-500 is a great choice.

Figure 29. Haas UMC-500

https://www.haascnc.com/index.html

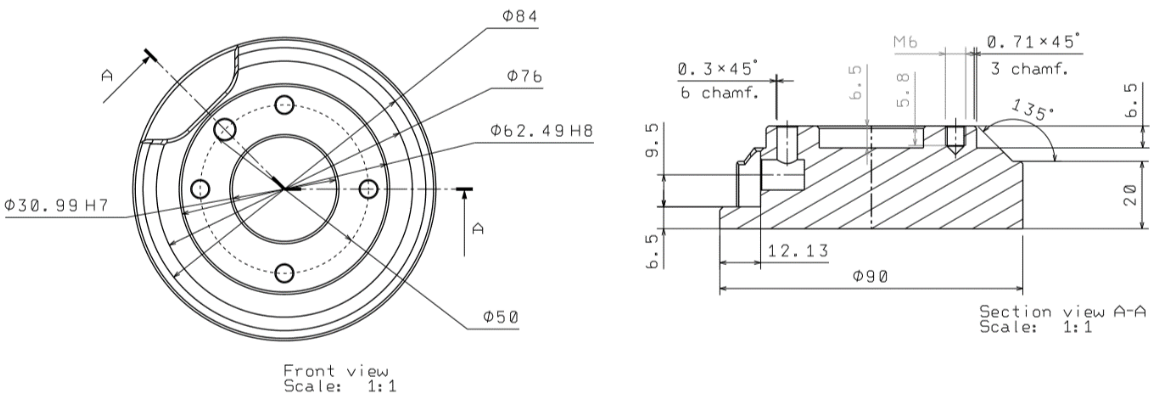


Figure 30. Blueprint of UR fixture plate.

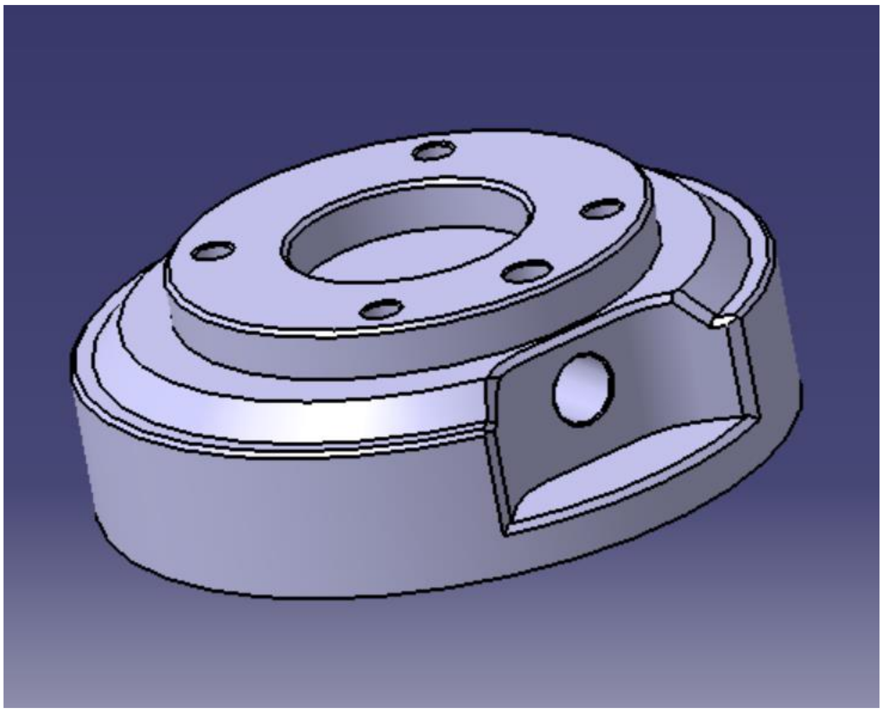
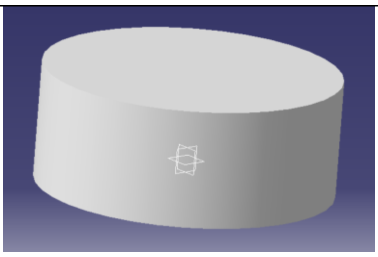
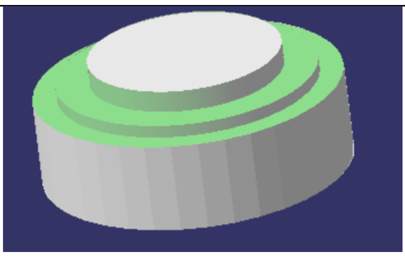
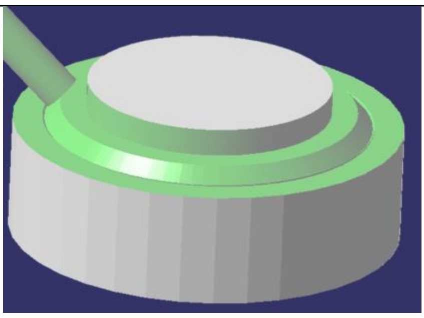
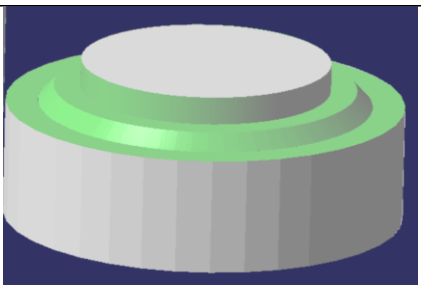
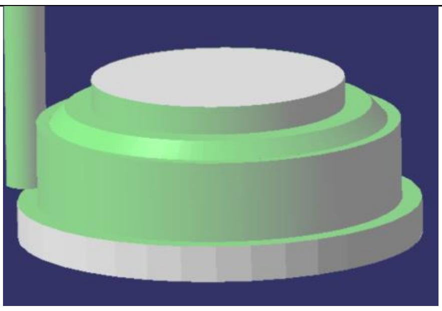
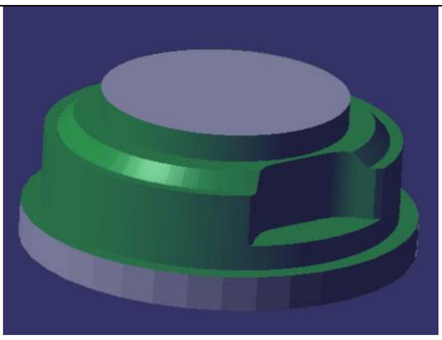
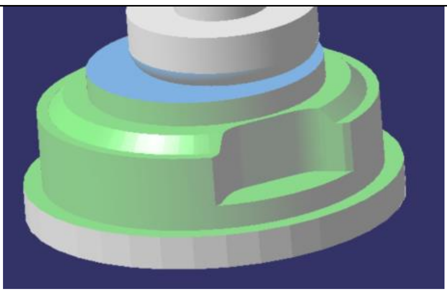
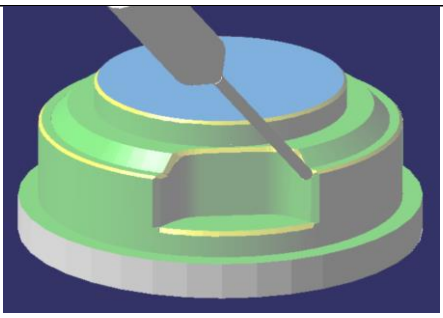
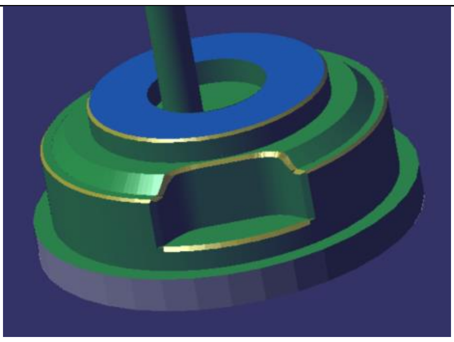
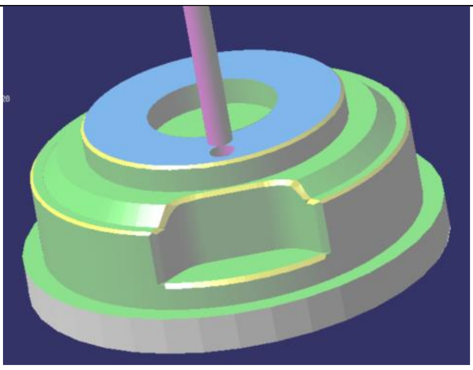


Figure 31. 3D model in CATIA V5 of UR fixture plate.

The process of machining goes as follows:

#	Name	Result	Time, sec.	Instrument
1	Stock		0	Power chuck
2	Profile Contouring		62	End Mill D10
3	Multi-axis machining		9.8	End Mill D10
4	Profile Contouring		16	End Mill D10
5	Profile Contouring		97	End Mill D10

6	Profile Contouring		88	End Mill D10
7	Facing		8	Face Mill D50
8	Multi-axis Machining		16	End Mill D3
9	Pocketing		24	End Mill D10
10	Drilling		7.5	Drill D6

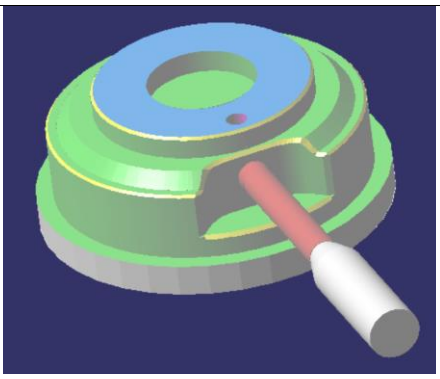
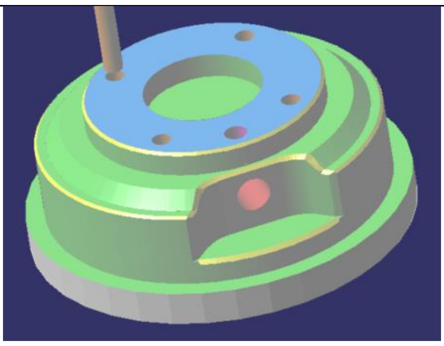
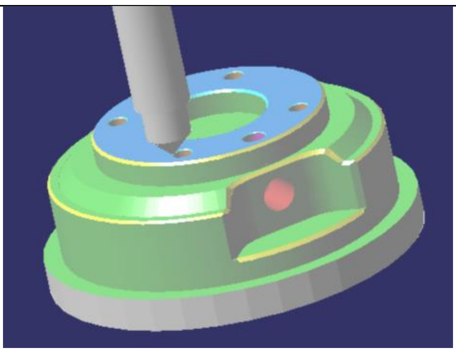
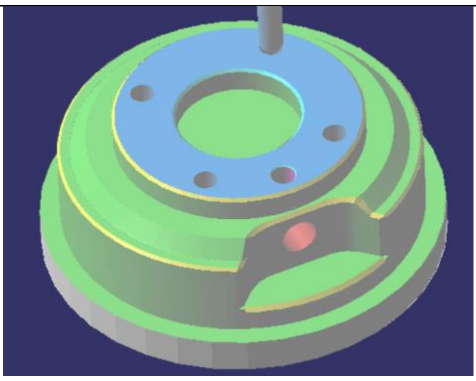
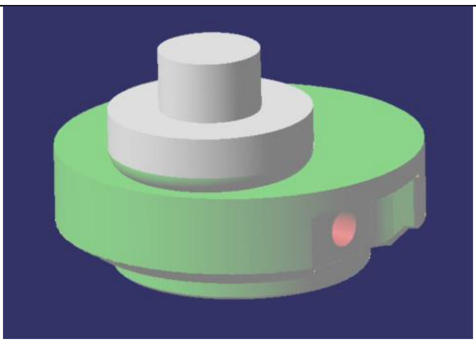
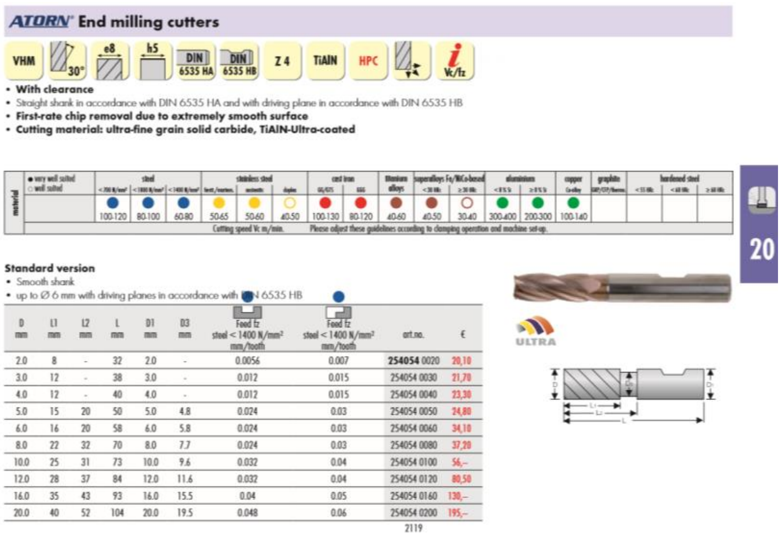
11	Drilling		7.4	Drill D9
12	Drilling		5	Drill D5
13	Curve Following		18	Countersink D20
14	Threading		20	Tap D6 (M6)
15	Facing		32	Face Mill D50

Table 2. Machining process

Equipment necessary for the task (Everything is from Sartorius catalogues):


ATORN® End milling cutters

VHM | 30° | e8 | h5 | DIN 6535 HA | DIN 6535 HB | Z 4 | TiAlN | HPC | Vc/fz

- **With clearance**
- Straight shank in accordance with DIN 6535 HA and with driving plane in accordance with DIN 6535 HB
- **First-rate chip removal due to extremely smooth surface**
- **Cutting material: ultra-fine grain solid carbide, TiAlN-Ultra-coated**

material	● very well suited ○ well suited	steel			stainless steel			cast iron		titanium alloys	superalloys Fe/NiCo-based		aluminium		copper	graphite	hardened steel		
		< 700 N/mm²	< 1000 N/mm²	< 1400 N/mm²	ferrit./martens.	austenitic	duplex	GG/GTS	GGG		< 38 HRC	≥ 38 HRC	< 8 % Si	≥ 8 % Si	Cu-alloy	GRP/CFP/thermo.	< 55 HRC	< 60 HRC	≥ 60 HRC
		●	●	●	●	●	○	●	●	●	●	○	●	●	●				
		100-120	80-100	60-80	50-65	50-60	40-50	100-130	80-120	40-60	40-50	30-40	300-400	200-300	100-140				

Cutting speed Vc m/min. Please adjust these guidelines according to clamping operation and machine set-up.

20

Standard version
- Smooth shank
- up to Ø 6 mm with driving planes in accordance with DIN 6535 HB

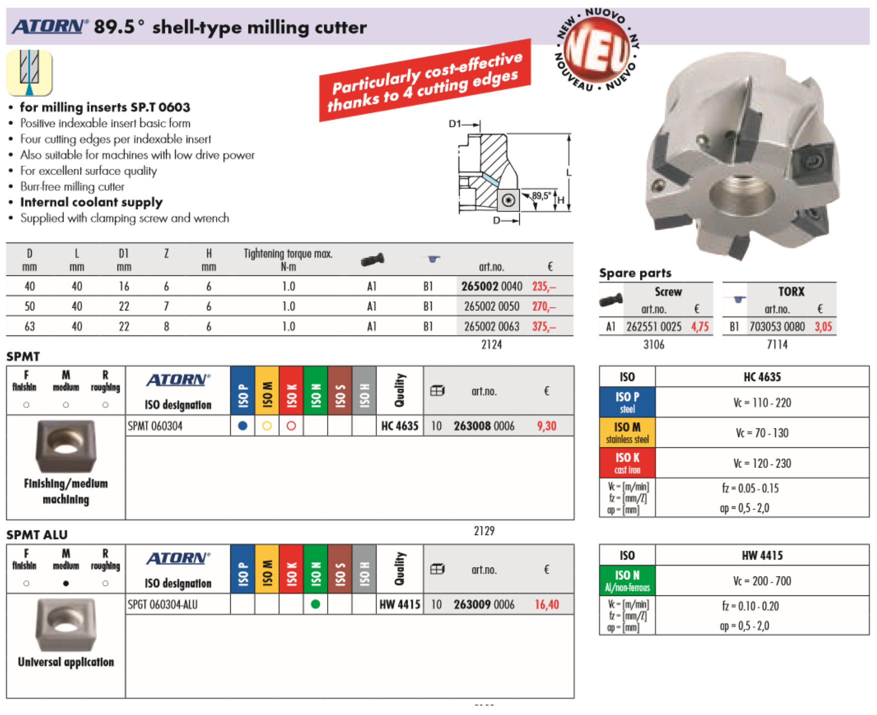
D mm	L1 mm	L2 mm	L mm	D1 mm	D3 mm	Feed fz steel < 1400 N/mm² mm/tooth	Feed fz steel < 1400 N/mm² mm/tooth	art.no.	€
2.0	8	-	32	2.0	-	0.0056	0.007	**254054** 0020	20,10
3.0	12	-	38	3.0	-	0.012	0.015	254054 0030	21,70
4.0	12	-	40	4.0	-	0.012	0.015	254054 0040	23,30
5.0	15	20	50	5.0	4.8	0.024	0.03	254054 0050	24,80
6.0	16	20	58	6.0	5.8	0.024	0.03	254054 0060	34,10
8.0	22	32	70	8.0	7.7	0.024	0.03	254054 0080	37,20
10.0	25	31	73	10.0	9.6	0.032	0.04	254054 0100	56,–
12.0	28	37	84	12.0	11.6	0.032	0.04	254054 0120	80,50
16.0	35	43	93	16.0	15.5	0.04	0.05	254054 0160	130,–
20.0	40	52	104	20.0	19.5	0.048	0.06	254054 0200	195,–

2119

ULTRA

End Mill D3: 22€

End Mill D10: 56€


ATORN® 89.5° shell-type milling cutter

Particularly cost-effective thanks to 4 cutting edges

NEW · NUOVO · NY · NUEVO · NOUVEAU · NEU

- **for milling inserts SP.T 0603**
- Positive indexable insert basic form
- Four cutting edges per indexable insert
- Also suitable for machines with low drive power
- For excellent surface quality
- Burr-free milling cutter
- **Internal coolant supply**
- Supplied with clamping screw and wrench

D1 L 89,5° H D

D mm	L mm	D1 mm	Z	H mm	Tightening torque max. N·m			art.no.	€
40	40	16	6	6	1.0	A1	B1	**265002** 0040	235,–
50	40	22	7	6	1.0	A1	B1	265002 0050	270,–
63	40	22	8	6	1.0	A1	B1	265002 0063	375,–

2124

Spare parts

	Screw art.no.	€		TORX art.no.	€
A1	262551 0025	4,75	B1	703053 0080	3,05

3106 7114

SPMT

F finishin ○	M medium ○	R roughing ○	ATORN® ISO designation	ISO P	ISO M	ISO K	ISO N	ISO S	ISO H	Quality		art.no.	€
Finishing/medium machining			SPMT 060304	●	○	○				HC 4635	10	**263008** 0006	9,30

2129

ISO	HC 4635
ISO P steel	Vc = 110 - 220
ISO M stainless steel	Vc = 70 - 130
ISO K cast iron	Vc = 120 - 230
Vc = [m/min] fz = [mm/Z] ap = [mm]	fz = 0.05 - 0.15 ap = 0,5 - 2,0

SPMT ALU

F finishin ○	M medium ●	R roughing ○	ATORN® ISO designation	ISO P	ISO M	ISO K	ISO N	ISO S	ISO H	Quality		art.no.	€
Universal application			SPGT 060304-ALU				●			HW 4415	10	**263009** 0006	16,40

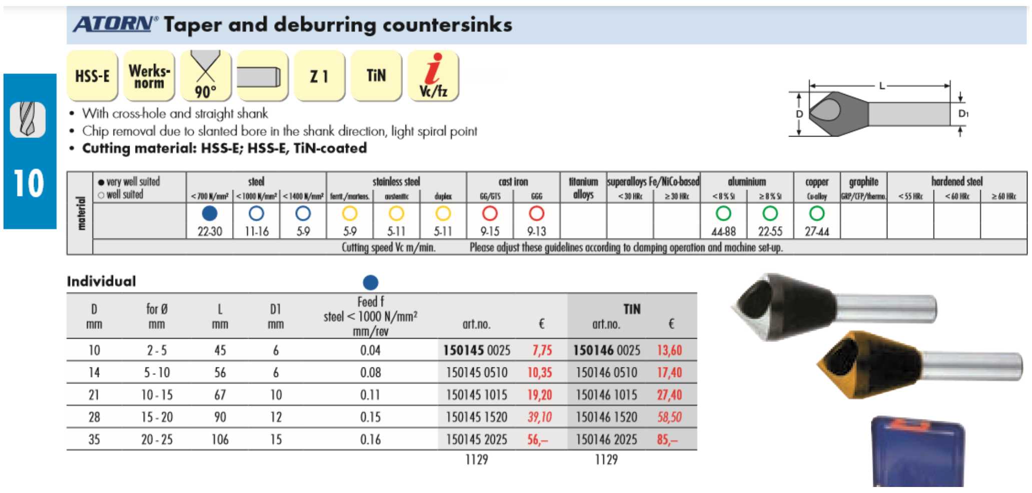
ISO	HW 4415
ISO N Al/non-ferrous	Vc = 200 - 700
Vc = [m/min] fz = [mm/Z] ap = [mm]	fz = 0.10 - 0.20 ap = 0,5 - 2,0

End Mill D50: 270€

7 Inserts SPMT ALU: 16.40€

7 Screws: 35€

					DP MA61206. 00.000 EN	Page
						45
Ed.	Page	№ doc.	Sign	Date		



ATORN® Taper and deburring countersinks

10

HSS-E | Werks-norm | 90° | Z 1 | TiN | Vc/fz

- With cross-hole and straight shank
- Chip removal due to slanted bore in the shank direction, light spiral point
- **Cutting material: HSS-E; HSS-E, TiN-coated**

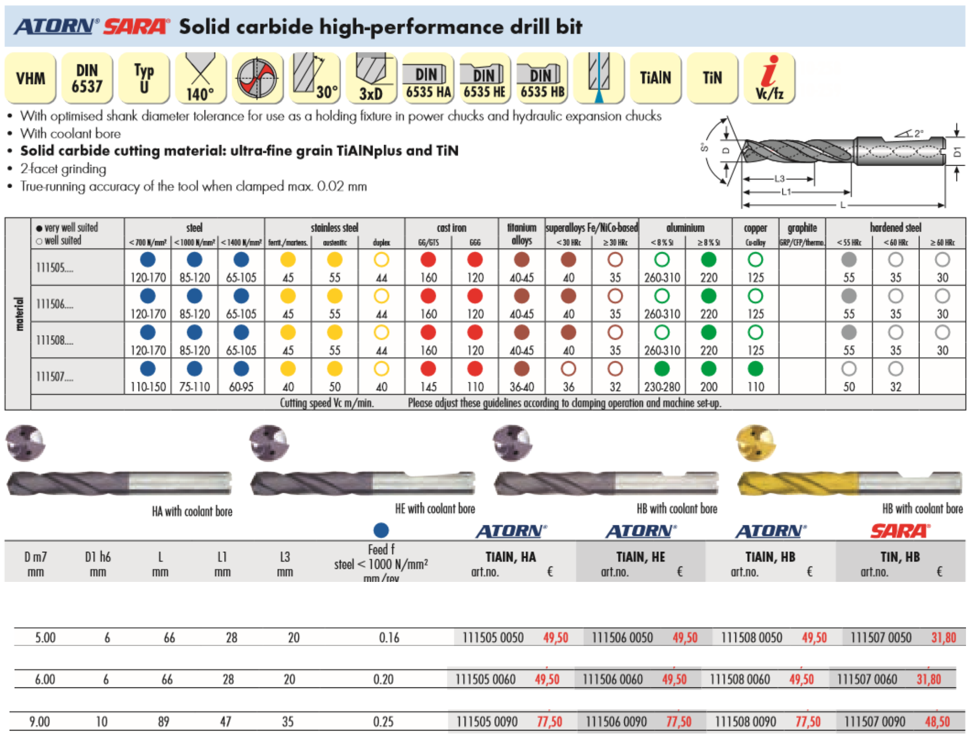
material	● very well suited ○ well suited	steel < 700 N/mm²	steel < 1000 N/mm²	steel < 1400 N/mm²	stainless steel ferrit./martens.	stainless steel austenitic	stainless steel duplex	cast iron GG/GTS	cast iron GGG	titanium alloys	superalloys Fe/NiCo-based < 30 HRc	superalloys Fe/NiCo-based ≥ 30 HRc	aluminium < 8 % Si	aluminium ≥ 8 % Si	copper Cu-alloy	graphite GRP/CFP/thermo.	hardened steel < 55 HRc	hardened steel < 60 HRc	hardened steel ≥ 60 HRc
		● 22-30	○ 11-16	○ 5-9	○ 5-9	○ 5-11	○ 5-11	○ 9-15	○ 9-13				○ 44-88	○ 22-55	○ 27-44				

Cutting speed Vc m/min. Please adjust these guidelines according to clamping operation and machine set-up.

Individual

D mm	for Ø mm	L mm	D1 mm	● Feed f steel < 1000 N/mm² mm/rev	art.no.	€	TiN art.no.	€
10	2 - 5	45	6	0.04	**150145** 0025	7,75	**150146** 0025	13,60
14	5 - 10	56	6	0.08	150145 0510	10,35	150146 0510	17,40
21	10 - 15	67	10	0.11	150145 1015	19,20	150146 1015	27,40
28	15 - 20	90	12	0.15	150145 1520	39,10	150146 1520	58,50
35	20 - 25	106	15	0.16	150145 2025	56,–	150146 2025	85,–
					1129		1129	

Countersink D20: 27€



ATORN® SARA® Solid carbide high-performance drill bit

VHM | DIN 6537 | Typ U | 140° | 30° | 3xD | DIN 6535 HA | DIN 6535 HE | DIN 6535 HB | TiAlN | TiN | Vc/fz

- With optimised shank diameter tolerance for use as a holding fixture in power chucks and hydraulic expansion chucks
- With coolant bore
- **Solid carbide cutting material: ultra-fine grain TiAlNplus and TiN**
- 2-facet grinding
- True-running accuracy of the tool when clamped max. 0.02 mm

material	● very well suited ○ well suited	steel < 700 N/mm²	steel < 1000 N/mm²	steel < 1400 N/mm²	stainless steel ferrit./martens.	stainless steel austenitic	stainless steel duplex	cast iron GG/GTS	cast iron GGG	titanium alloys	superalloys Fe/NiCo-based < 30 HRc	superalloys Fe/NiCo-based ≥ 30 HRc	aluminium < 8 % Si	aluminium ≥ 8 % Si	copper Cu-alloy	graphite GRP/CFP/thermo.	hardened steel < 55 HRc	hardened steel < 60 HRc	hardened steel ≥ 60 HRc
	111505....	● 120-170	● 85-120	● 65-105	● 45	● 55	○ 44	● 160	● 120	● 40-45	● 40	○ 35	○ 260-310	● 220	○ 125		● 55	○ 35	○ 30
	111506....	● 120-170	● 85-120	● 65-105	● 45	● 55	○ 44	● 160	● 120	● 40-45	● 40	○ 35	○ 260-310	● 220	○ 125		● 55	○ 35	○ 30
	111508....	● 120-170	● 85-120	● 65-105	● 45	● 55	○ 44	● 160	● 120	● 40-45	● 40	○ 35	○ 260-310	● 220	○ 125		● 55	○ 35	○ 30
	111507....	● 110-150	● 75-110	● 60-95	● 40	● 50	○ 40	● 145	● 110	● 36-40	○ 36	○ 32	● 230-280	● 200	● 110		○ 50	○ 32	

Cutting speed Vc m/min. Please adjust these guidelines according to clamping operation and machine set-up.

HA with coolant bore

HE with coolant bore

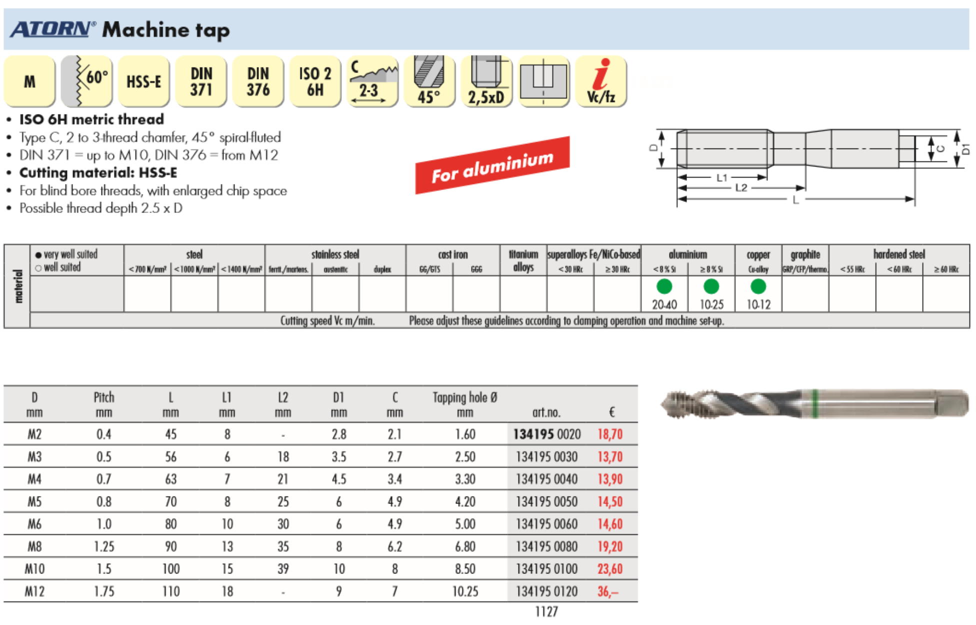
HB with coolant bore

HB with coolant bore

D m7 mm	D1 h6 mm	L mm	L1 mm	L3 mm	● Feed f steel < 1000 N/mm² mm/rev	ATORN® TiAlN, HA art.no.	€	ATORN® TiAlN, HE art.no.	€	ATORN® TiAlN, HB art.no.	€	SARA® TiN, HB art.no.	€
5.00	6	66	28	20	0.16	111505 0050	49,50	111506 0050	49,50	111508 0050	49,50	111507 0050	31,80
6.00	6	66	28	20	0.20	111505 0060	49,50	111506 0060	49,50	111508 0060	49,50	111507 0060	31,80
9.00	10	89	47	35	0.25	111505 0090	77,50	111506 0090	77,50	111508 0090	77,50	111507 0090	48,50

Drills D5, D6, D9: 178€

					DP MA61206. 00.000 EN
Ed.	Page	№ doc.	Sign	Date	


ATORN® Machine tap

M | 60° | HSS-E | DIN 371 | DIN 376 | ISO 2 6H | C 2-3 | 45° | 2,5xD | Vc/fz

- **ISO 6H metric thread**
- Type C, 2 to 3-thread chamfer, 45° spiral-fluted
- DIN 371 = up to M10, DIN 376 = from M12
- **Cutting material: HSS-E**
- For blind bore threads, with enlarged chip space
- Possible thread depth 2.5 x D

For aluminium

material	steel			stainless steel			cast iron		titanium alloys	superalloys Fe/NiCo-based		aluminium		copper	graphite	hardened steel		
● very well suited ○ well suited	< 700 N/mm²	< 1000 N/mm²	< 1400 N/mm²	ferrit./martens.	austenitic	duplex	GG/GTS	GGG		< 30 HRc	≥ 30 HRc	< 8 % Si	≥ 8 % Si	Cu-alloy	GRP/CFP/thermo.	< 55 HRc	< 60 HRc	≥ 60 HRc
												● 20-40	● 10-25	● 10-12				

Cutting speed Vc m/min. Please adjust these guidelines according to clamping operation and machine set-up.

D mm	Pitch mm	L mm	L1 mm	L2 mm	D1 mm	C mm	Tapping hole Ø mm	art.no.	€
M2	0.4	45	8	-	2.8	2.1	1.60	**134195** 0020	18,70
M3	0.5	56	6	18	3.5	2.7	2.50	134195 0030	13,70
M4	0.7	63	7	21	4.5	3.4	3.30	134195 0040	13,90
M5	0.8	70	8	25	6	4.9	4.20	134195 0050	14,50
M6	1.0	80	10	30	6	4.9	5.00	134195 0060	14,60
M8	1.25	90	13	35	8	6.2	6.80	134195 0080	19,20
M10	1.5	100	15	39	10	8	8.50	134195 0100	23,60
M12	1.75	110	18	-	9	7	10.25	134195 0120	36,–

1127

Tap M6: 14.6€

Total equipment cost: 619€

Stock cost: 35€

Total machining time: 411 sec. = 6.85 min.

5. Safety and precautions

In this work author made a software to control a robot in working environment, such as labs. The main hazards in these conditions are:

- Microclimate
- House light
- Electrical safety
- Fire safety

5.1 Object characteristics

The typical working environment for the robot is *20 sq.m.* and volume respectively $V=20\cdot 2{,}7=54m^3$. PC work is basic, so it must be provided with optimal microclimate parameters. In order to create these parameters, the room is ventilated by a window and air is ventilated by an air conditioner. Thanks to this, the PC user is able to receive clean air without contaminated particles and work normally. It should also be noted that computers are a source of heat, which in turn can reduce the humidity in the room and raise the air temperature. In those rooms where the computer is located, it is important to adhere to the permissible parameters of the microclimate that meet the standards specified in the document SSN - State sanitary norms of the microclimate of industrial premises. In addition to the above, it is important to ensure acceptable concentrations of negative and positive ions in the room air. After all, the results of research show the following: while negative ions have a positive effect on human health, positive ions in too large an amount adversely affect mental and physical performance (for example, a person is tired) and the cardiovascular system. So, it is really worth paying attention to the value of the parameters of the microclimate in the room. Since the work on the PC belongs to the category of Easy Ia, we can compare the values between the optimal and the actual parameters of the microclimate in Table 7.1.

Time of the year	*Temperature, ºC*			*Humidity, %*	
	Optimal	*Real*		*Optimal*	*Real*
		Upper bound	*Lower bound*		
Cold	*22 - 24*	*24*	*21*	*40 - 60*	*50*
Hot	*23 – 25*	*28*	*25*	*40 - 60*	*42*

Table 7.1. Optimal temperature and humidity

The room temperature in the warm period of the year is 24*ºC* and is maintained by the BK2000 air conditioner, the relative humidity of 42% is maintained by a humidifier. In the cold period of the year, the average temperature is 23*ºC*, which is maintained by central heating. The value of relative humidity is 57%.

All parameters of the microclimate of the room in the warm and cold period of the year are in the range of optimal values, so we can conclude that the microclimate of the room is favorable for work.

5.2 House Light

To carry out certain work indoors, and even more work on a PC, natural light is not enough, especially in the cold season. And insufficient lighting helps to reduce attention and concentration, leads to eyestrain and premature fatigue. Therefore, it is considered appropriate to use artificial lighting with a variety of lamps. However, it is necessary to calculate the optimal level of lighting, because not only lack but also excessive lighting causes eye pain, irritation and blindness. All this can lead to accidents or occupational diseases, which is why it is important to start the correct calculation of lighting, determine the number and type of lamps, their location, and so on before starting work. The illumination of the surface of the computer screen should be no more than 300 lux, and the surface of the table in the area of the working document - no more than 300-500 lux. Under no circumstances should the lighting create glare on the screen that prevents the PC user from operating normally without distraction. The workplace uses both natural side lighting (the sun's rays penetrate through the window) and artificial general lighting with incandescent lamps - electric energy sources. Based on the above information, calculate the level of illumination of the working space.

In this case, the work is done on a computer, so it refers to high-precision work, the class of visual work – Class III, type c or d (в или г) according to GBN B.2.5-28:2018 (ДБН В.2.5-28:2018) Table 1.

The calculation of lighting is performed for a room with an area of 20 m ^{2}, width of which is 4 m, height - 2.7 m. The calculation will be performed by the method of light flux.

Determine the luminous flux in the room and compare it with the allowable, according to the formula:

$$E_{eff.} = \frac{F_l N n \eta}{S \cdot k \cdot z}$$

Where

$E_{eff.}$ – the effective luminous flux;

F_l - normalized minimum illuminance, lm (determined by the table). The work of a programmer, according to this table, can be classified as high-precision work, therefore, the minimum illuminance will be F_l = 200 lm.

S - the area of the illuminated room (in our case S = 20m ^{2});

z - the ratio of average illuminance to the minimum (usually taken equal to 1.1 ... 1.2, let z = 1.1);

k - the safety factor, takes into account the reduction of the luminous flux of the lamp as a result of contamination of the lamps during operation (its value depends on the type of room and the nature of the work carried out in it and in our case k = 1.5);

N – number of lamps;

n – number of lightning bulbs in each lamp;

η - utilization factor, (expressed as the ratio of luminous flux incident on the calculated surface to the total flux of all lamps and calculated in fractions of a unit; depends on the characteristics of the lamp, room size, wall and ceiling painting, characterized by reflection coefficients from the walls (Pw) and ceiling (Rc)), the ceiling of the room is freshly painted white Pw = 70%, the walls are light beige Pw = 50%, lacquered parquet floor Rc = 30%. The value of η is determined by the table of coefficients of use of different lamps. To do this, calculate the index of the room by the formula:

$$I = \frac{S}{h(A+B)} = \frac{20}{1.9\cdot(4+5)} = 1.169$$

Where

h - estimated height of the mounting, h = 1.9 m;

A – width of the room, A = 4m;

B – length of the room, B = 5m.

Knowing the index of room I, is η = 0.47.

To illuminate the room, use one lamp, equipped with 2 lamps. According to the room index and luminous flux coefficients from the floor - 0.3, from the walls - 0.5 and from the ceiling - 0.7, we determine the value of the luminous flux utilization factor η = 0.47 for the GLOBAL A60 LED lamp.

Substitute all the values in the formula to determine the luminous flux $E_{eff.}$:

$$E_{eff.} = \frac{15000 \cdot 1 \cdot 2 \cdot 0{,}47}{20 \cdot 1{,}5 \cdot 1{,}1} = 427{,}27\ lm$$

Therefore, it can be concluded that the illumination of the room is not sufficient, which is why local lighting is used, in the form of a table lamp with an incandescent bulb with a power of 60 watts. But the brightness of the screen meets the standards.

To improve lighting, as well as to save electricity, replace the incandescent lamp in the chandelier with an LED lamp E27-E40 6400K LB-65 Feron.

5.3 Electrical safety

Electrical safety is a set of organizational and technical measures that protect people from the adverse effects of electric current, arc, static electricity and electromagnetic fields. The room is operated on one PC, which is connected to an outlet with a separate fuse in case of overload of the mains. For connection of other portable electric equipment flexible wires in reliable isolation equipped with the additional safety lock of switching off are used. Wiring to power sources from portable devices is the shortest way. Under no circumstances should wires be entangled in appliances or furniture.

The following warnings and cautions must be observed when a robot application is designed and installed. The warnings and cautions also apply for service work.

1. Never connect safety signals to a PLC which is not a safety PLC with the correct safety level. Failure to follow this warning could result in serious injury or death as one of safe stop functions could be overridden. It is important to keep safety interface signals separated from the normal I/O interface signals.
2. All safety-related signals are constructed redundantly (Two independent channels). Keep the two channels separate so that a single fault cannot lead to loss of the safety function.
3. Make sure that all equipment not rated for water exposure remains dry. If water comes inside the product, lockout and tag out all power and then contact your supplier.
4. Minus connections are referred to as “GND” and are connected to the shield of the robot and the controller box. All mentioned GND connections are only for powering and signaling. For PE (Protective Earth) use the M6 sized screw connections marked with earth symbols inside the control box. The grounding conductor shall have at least the current rating of the highest current in the system.

5.4 Fire safety

According to the fire hazard of the room, in which the PC is located, belong to category B. These rooms must meet the requirements for fire prevention and extinguishing according to NPB 166-97, telephone and fire alarm must also be available. In this case, since it is a normal room of a small area in the house, it becomes clear that it is not equipped with a special fire alarm. There is also no wired telephone in the room, but it is possible to make calls using modern mobile gadgets and mobile operators. According to SNiP 21-01-97 the building corresponds to the II degree of fire resistance, which requires a fairly quick response from firefighters. According to SNiP, the minimum limits of fire resistance of building structures are as follows: 1. columns, load-bearing walls, walls of stairwells - 2 hours; 2. self-supporting walls, stairwells, steps, beams - 1 hour; 3. slabs, decks (including with insulation) and other load-bearing structures of the floor - 0.75 hours; 4. external and internal (partitions) non-bearing walls, covering elements (slabs, decks, beams, arches, frames) - 0.25 hours. The house is built of brick and other natural materials that are not flammable, so it is safe to say that a fire that cannot be extinguished on one`s own, it is difficult to prevent.

According to NPB 166-97 the supposed fire category is D class – metal and metal containing structures. Thus, we chose powder fire extinguisher, that will perform great in given conditions, but require spray damper.

According to DSTU 3734-98 and for D-category room – maximum distance to expected fire source up to 70m. Thus, for our room 1 powder extinguisher is sufficient.

6. Conclusion

During the work on this project author has faced and solved many issues and problems regarding programming and control. Resulting software is fairly complex, where different solutions intertwine to result in digital simulation in CoppeliaSim.

This thesis lays a solid base for Magister thesis and future work. This includes, but not limited to: video-tracking system, self- and obstacle avoidance based on accurate model and real-life simulations, physical implementation and testing.

This program can be recommended for the fellow students to study basics of robot manipulation and how to program it.

7. References

[1] Kevin M. Lynch and Frank C. Park 30.11.19 “MODERN ROBOTICS MECHANICS, PLANNING, AND CONTROL”

[2] Anthony A. Maciejewski and Charles A. Klein The International Journal of Robotics Research, 1985, “Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environment” DOI: 10.1177/027836498500400308.

[3] Tuomo Kivel`a, Jouni Mattila, Jussi Puura, and Sirpa Launis 2017 “Redundant Robotic Manipulator Path Planning for Real-Time Obstacle and Self-Collision Avoidance” conference paper version for RAAD 2017 conference. DOI: 10.1007/978-3-319-61276-8_24

[4] Agostino De Santis, Alin Albu–Sch¨affer, Christian Ott, Bruno Siciliano, and Gerd Hirzinge 2014 “The skeleton algorithm for self-collision avoidance of a humanoid manipulator” DOI: 10.1109/AIM.2007.4412606

[5] David Beazey & Brian K. Jones “Python Cookbook” 3rd edition ISBN: 978-1-449-34037-7

[6] В.К. Буслов Методичні вказівки до курсового проекту за курсом «ПРОЕКТУВАННЯ ОБ’ЄМНИХ ГІДРОПРИВОДІВ» для студентів з фаху «ГІДРАВЛІЧНІ І ПНЕВМАТИЧНІ МАШИНИ»

[7] https://www.coppeliarobotics.com/

[8] https://www.kuka.com/