

# Endoscopic end effector control using soft actuator

Jorn Jansen

June 29, 2018



## Abstract

The minimally invasive and non-invasive surgery are gradually replacing open surgery thanks to an instrument named endoscope. The significance of using soft actuators is to create a flexible instrument that can re-enact properties of a rigid instrument.

In this study, a modified design of an already existing modular continuum robot, STIFF-FLOP, is presented. The number of internal pneumatic chambers has been increased from 3 to 4 to enable antagonistic actuation.

An open-loop control system is implemented based on changing the pneumatic chamber lengths by controlling the applied air pressure. The use of pneumatic actuation in this open-loop control requires the relation between the applied pressure and the pneumatic chamber lengths. This relation is experimentally obtained on one soft actuator in 2 antagonistic directions and with the use of image software the lengths are extracted. To these results a polynomial curve was fitted. A proof of concept for non-antagonistic bending has also been provided in this study.

The control system calculates the chamber lengths based on the desired rotation and bending, it would then, with the use of the pressure-chamber length relation, determine the required air pressure. Because both the actuated chamber and the non-actuated chamber lengths are included in this relation, the algorithm would compensate the actuated chamber length for any changes in the non-actuated chamber length.

With the use of the passive chamber compensation, the angular error was decreased with an average of 70.1% compared to not compensating. Final angular error (with compensation) results have a maximum of  $4.74^\circ$  and  $4.31^\circ$  for the left and right bending respectively.

The single section modelling and characterization serves as a basis for a more complex multi section endoscope. A possible application is an endoscope that remembers the set angle, by its operator, depending on the insertion of the endoscope, this has been given the name Insertion Dependent Angle Memory Control. This data can then be used to keep a specific angle at a specific spot by translating the angle between the multiple sections. .



## Acknowledgement

I would like to thank dr. ir. M. Abayazid and H. Naghibi Beidokhti MSc. for guiding me through this research.

I would also like to thank Muhammad Wildan Gifari for giving me insight in the soft actuators and for constructing the specific model used in this study.

Furthermore, I would like dr. ir. P. Breedveld for being the chair and dr. ir. B.J.F. van Beijnum for being the external member of my bachelor assignment committee and everyone reading my bachelor thesis.



---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Soft Robotics Endoscope . . . . .	1
1.2	Soft Actuator Module . . . . .	2
<b>2</b>	<b>Modelling the Soft Actuator Module</b>	<b>4</b>
2.1	Robot Independent Mapping . . . . .	4
2.2	Robot Specific Mapping . . . . .	6
2.2.1	Parameters . . . . .	6
2.2.2	Inverse Solution . . . . .	6
2.2.3	Chamber Length to Pressure Relation . . . . .	7
2.3	Constraints . . . . .	7
<b>3</b>	<b>Constant Curvature Kinematics</b>	<b>8</b>
3.1	Single Module Kinematics . . . . .	8
3.1.1	Forward Kinematics . . . . .	8
3.1.2	Forward Differential Kinematics . . . . .	8
3.1.3	Inverse Differential Kinematics . . . . .	9
3.2	Multi Module Kinematics . . . . .	10
3.2.1	Forward Kinematics . . . . .	10
3.2.2	Forward Differential Kinematics . . . . .	10
3.2.3	Inverse Differential Kinematics . . . . .	10
<b>4</b>	<b>Implementation</b>	<b>11</b>
4.1	Soft Actuator Module . . . . .	11
4.2	2D Implementation . . . . .	11
4.2.1	Calibration . . . . .	11
4.2.2	Validation . . . . .	12
4.3	3D Implementation . . . . .	13
4.3.1	Calibration . . . . .	13
4.3.2	Validation . . . . .	13
4.4	Control Algorithm . . . . .	13
4.5	Control System . . . . .	14
<b>5</b>	<b>Simulations</b>	<b>16</b>
5.1	2D Single Module . . . . .	16
5.1.1	Forward Kinematics . . . . .	16
5.2	3D Single Module . . . . .	16

---

5.2.1	Forward Kinematics . . . . .	17
5.3	3D Multi Module . . . . .	17
5.3.1	Forward Kinematics . . . . .	18
<b>6</b>	<b>Results</b>	<b>20</b>
6.1	Validation: Piecewise Constant Curvature Assumption . . . . .	20
6.2	Chamber Length - Pressure Relation . . . . .	20
6.2.1	2D: Calibration . . . . .	20
6.2.2	2D: Validation . . . . .	22
6.3	3D: Proof of Concept . . . . .	22
<b>7</b>	<b>Discussion</b>	<b>24</b>
<b>8</b>	<b>Conclusions and Recommendations</b>	<b>27</b>
8.1	Conclusions . . . . .	27
8.2	Recommendations . . . . .	27
<b>A</b>	<b>Single Module Simulation MATLAB Code</b>	<b>29</b>
<b>B</b>	<b>Multi Module Simulation MATLAB Code</b>	<b>32</b>
<b>C</b>	<b>2D Control System MATLAB Code</b>	<b>35</b>
<b>D</b>	<b>2D Insertion Dependent Angle Memory Control MATLAB Code</b>	<b>39</b>
	<b>Bibliography</b>	<b>43</b>



# 1 Introduction

## 1.1 Soft Robotics Endoscope

During surgical operations, less pain, scarring and trauma will greatly increase the recovery rate of the patient. Minimal Invasive Surgery (MIS) and Natural Orifice Transluminal Endoscopic Surgery (NOTES) are able to reduce these factors compared to traditional open surgery. One of the key techniques used in MIS and NOTES is endoscopy. Traditionally, the endoscope was used to look inside the patient's lumen, but with the help of modern robotics this use has been extended to also perform surgical tasks.

Current endoscopic instruments are either rigid or flexible. Rigid instruments give a high precision and force, at the cost of very poor manipulability. Flexible instruments often have a much higher dexterity, and can better circumvent healthy organs to reach a distant organ and give better vision of the surgical site but mostly lack the precision of rigid instruments [1].

Combining characteristics of both rigid and flexible instruments can be embodied by soft robotics. These robotics uses actuators made of soft and compliant materials. These materials are usually biocompatible, which means the material is not harmful or toxic to living tissue. Furthermore, due to the soft and compliant nature of these materials, they allow for a safer interaction with the human body as they can deform and shape with the body, which prevents large stresses on the living tissue.

While the flexible aspects of soft robotics allow for a high dexterity, it lacks the precision and capability to apply a force. A technique called granular jamming, which is commonly used in these kinds of applications [6], is able to manipulate the stiffness of the soft robotics actuator. Granular jamming works by vacuuming the capsule with a granular material (e.g. coffee powder), increasing the particle density and as a result increasing the viscosity of the material. This design is inspired by biological “manipulators”, such as elephant trunks and octopus arms. These “manipulators” can act as a soft manipulator, but also control their stiffness to act as a rigid manipulator [8].

Another major advantage of using soft robotics is the ability to be used inside an MRI environment. The materials and the actuation method do not distort the MR signals, which makes it possible to precisely locate the soft robotic device during an MRI scan. This information can then be used to precisely guide the device through the body.

The aim of this study is to develop a control system to control the motion and bending of a four-chamber pneumatic controlled soft actuator, further referred to as “soft actuator” or “soft actuator module”, in 3 dimensions using forward kinematics. This is done by first creating a 2D version of the soft actuator module and modelling the characteristics. These characteristics are then used to create a realistic model, and subsequently, a control system for the actual soft actuator module. This is then followed by constructing a 3D version using the same method as for the 2D version, with more complex characteristics due to the increase in number of chambers.

An addition to this is to create a multi module robot by concatenating multiple single soft actuator modules and applying an inverse kinematic method, which allows for higher dex-

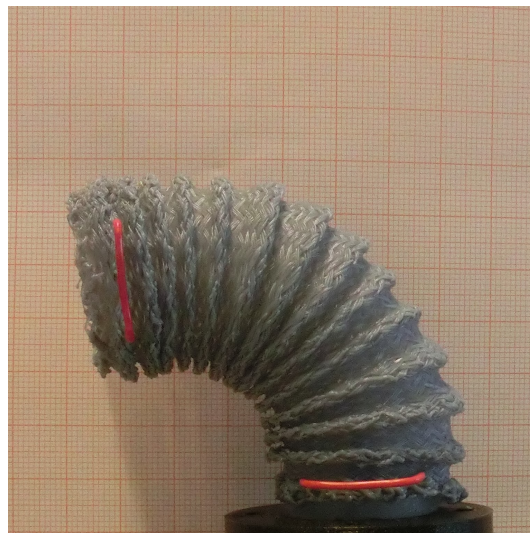
terity. Another application for a multi module robot is a guided endoscope that can follow a described path.

This soft actuator module can be used as an end effector for an MRI-compatible robotic endoscope, or by using multiple modules as the whole endoscope.

## 1.2 Soft Actuator Module

The soft actuator module has been designed to meet specific requirements and is based on the STIFF-FLOP design [3]. However, the actuator presented in this report has four internal pneumatic chambers, instead of three in the STIFF-FLOP design, placed at  $90^\circ$  intervals. This enables antagonistic actuation. Antagonistically arranged chambers would allow more complicated motion trajectories and precise amount of exerted forces [2].

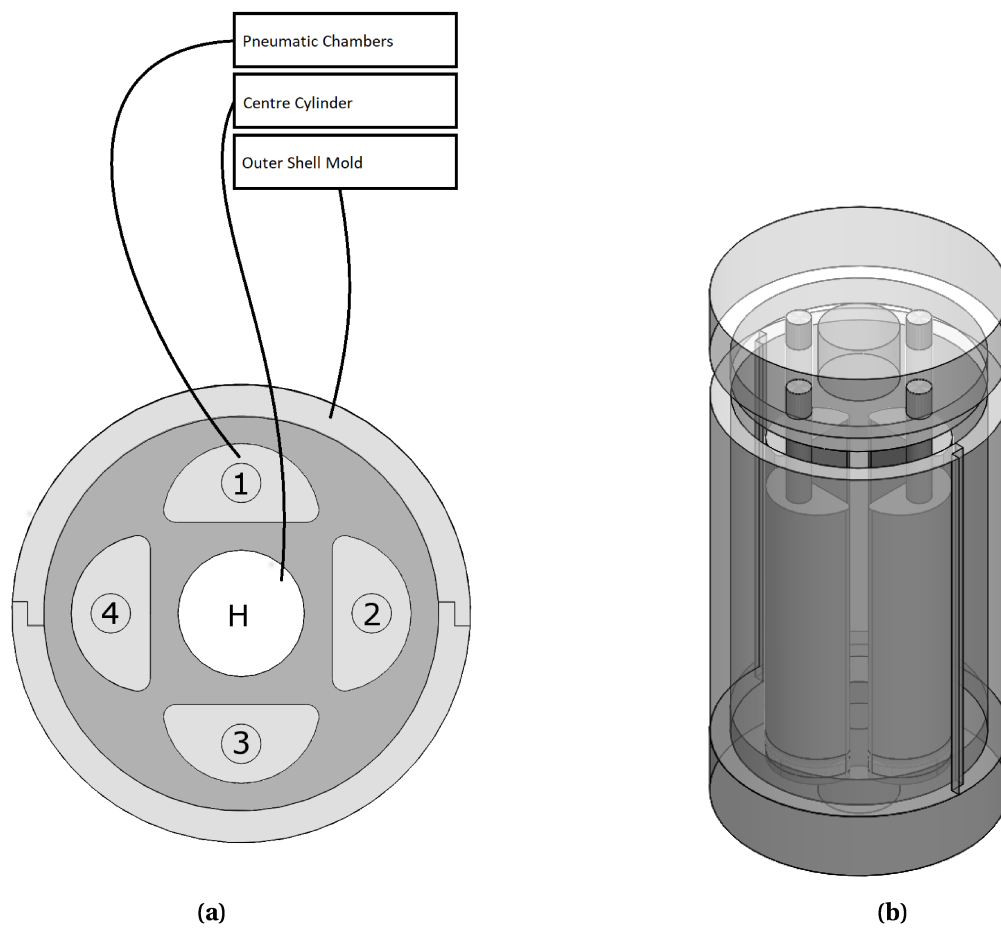
The pneumatic chambers, denoted by the numbers 1, 2, 3, & 4 in figure 1.2a, must be able to elongate individually to create an effective bending of the soft actuator module in a specific direction. A hollow cylinder in the middle of the soft actuator module, denoted by the letter H in figure 1.2a can be used for data cables for e.g. a camera, pneumatic tubes for the next section(s), or other instruments or cables.



**Figure 1.1:** Soft actuator module bending

The soft actuator module, shown in figure 1.1, is made by 3D printing a mold, which is shown in figure 1.2. This mold is then filled with elastomer to create the actual soft actuator module.

To control the stiffness of the soft actuator module, granular jamming has been implemented by inserting a flexible capsule of very fine coffee powder inside the pneumatic chambers (1, 2, 3, & 4 in figure 1.2a). This capsule has been fixated to the top and bottom, to ensure it gives a uniform stiffness. The way it is activated, is by vacuuming the capsule, which will decrease the volume only slightly but will increase the particle density of the coffee powder resulting in an increased viscosity of the material creating a stiff structure. The granular jamming is part of the actuator, but will not be discussed any further in this report.



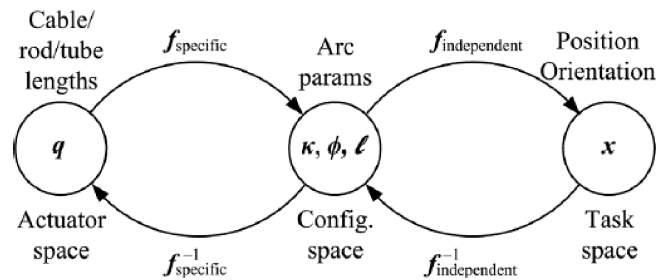
**Figure 1.2:** Top view (a) and side view (b) of the soft actuator module mold

## 2 Modelling the Soft Actuator Module

The soft actuator module is controlled with air pressure to achieve the desired bending. This control relies on a correlation between the air pressure and the bending, which is again dependent on the material specifications of the soft actuator module.

The modelling of the soft actuator module relies on the assumption of piecewise constant-curvature. This implies that all along the arc, the backbone of the soft actuator module, the curvature is constant, in other words, the it is assumed the bending of the soft actuator module is uniform along the pneumatic chamber length. This assumption simplifies the kinematic modeling and enables the kinematics to be decomposed into two mappings, as described by Jones and Walker (2006) [5]. The use of these two mappings splits the kinematics and robot specific control parameters. Making it possible to use standard kinematics techniques, e.g. the use of the Jacobian for inverse kinematics.

The model consists of two mappings. First, a robot independent mapping,  $f_{independent}$ . This contains all arc parameters and relates these to the configuration of the soft actuator module. With the use of this mapping the end point of the soft actuator module in space can be determined. This mapping is applicable for all designs based on piecewise constant-curvature assumption. This gives the basis of forward kinematics. Second, a robot specific mapping,  $f_{specific}$ , which contains the relation between the applied air pressure and pneumatic chamber length will be determined by experimentation. The robot specific mapping then relates this relation to the arc parameters. Combining these two mapping, the shape of the soft actuator module can be determined by the applied pressure. A overview of the two mappings and their relations can be seen in figure 2.1 [9].

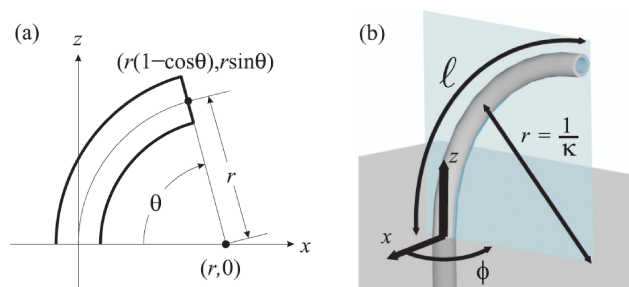


**Figure 2.1:** Three spaces mappings between them. Taken from [9].

The arc parameters, which defines the configuration of the robot consist of  $\kappa(\mathbf{q})$  describing the curvature of the robot,  $\phi(\mathbf{q})$  the angle of the plane containing the arc and  $\ell(\mathbf{q})$  the length of the arc, thus the soft actuator length. How these arc parameters are related to the model can be seen in figure 2.2b.

### 2.1 Robot Independent Mapping

The robot independent mapping can be obtained in multiple ways, and can be seen as a homogeneous transformation matrix, mapping the arc parameters to the end effector position. Similar to a regular homogeneous transformation matrix, the robot independent mapping can thus be obtained by the same methods as for obtaining a regular homogeneous transformation matrix. In this study it is chosen to use a geometry approach, because it can be constructed



**Figure 2.2:** (a)  $x-z$  plane, when  $\phi = 0$  (b) shown is how the plane in (a) rotates around the  $z$ -axis with  $\phi$  containing the circular arc with (constant) curvature  $\kappa$  and length  $\ell$ . Taken from [9].

using standard rotational matrices together with a single translation.

First, a frame convention needs to be chosen. Similar to Webster III and Bryan (2010) it is considered to have the  $+z$ -axis tangent to the base of the robot. At  $\phi = 0$  a positive bending ( $\kappa > 0$ ) will be around the  $+y$ -axis. An angle of  $\pi$  radians (for  $\phi = 0$ ) will then result in the backbone touching the  $+x$ -axis. This frame convention can be seen in figure 2.2a, with the  $+y$ -axis going into the paper and in figure 2.2b with the angle  $\phi$  going in the direction of the  $+y$ -axis [9].

Considering that the angle of bending,  $\theta$ , takes place in a 2D plane that rotates with  $\phi$  around the  $z$ -axis. This 2D plane equals the  $x-z$  plane if  $\phi = 0$ , seen in figure 2.2a. A point along the backbone of the arc with radius  $r$ , centered at  $[r \ 0 \ 0]^T$ , is given by  $p = [r(1 - \cos(\theta)) \ 0 \ r \sin(\theta)]^T$ . To correctly rotate the point with the arc, a rotation  $R_y(\theta) \in SO(3)$  is included. Together with the rotation  $\phi$  of the 2D plane about the  $z$ -axis,  $R_z(\phi) \in SO(3)$ , this results in the equation shown in (2.1), similar to Webster III and Bryan (2010) [9].

$$T = \begin{bmatrix} R_z(\phi) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_y(\theta) & \mathbf{p} \\ 0 & 1 \end{bmatrix} \quad (2.1)$$

When substituting  $\kappa = 1/r$  and  $\theta = \kappa s$  (Simaan, et. al. (2009)) where  $s$  is a point along the backbone of the arc ( $s \in [0, \ell]$ ), the equation can be written in terms of the arc parameters, shown in (2.2) [7].

$$T = \begin{bmatrix} \cos(\phi) \cos(\kappa s) & -\sin(\phi) & \cos(\phi) \sin(\kappa s) & \frac{\cos(\phi)(1 - \cos(\kappa s))}{\sin(\phi)(1 - \cos(\kappa s))} \\ \sin(\phi) \cos(\kappa s) & \cos(\phi) & \sin(\phi) \sin(\kappa s) & \frac{\sin(\phi)(1 - \cos(\kappa s))}{\sin(\kappa s)} \\ -\sin(\kappa s) & 0 & \cos(\kappa s) & \frac{\kappa}{1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

In (2.2) the orientation is so that the  $x$ -axis of the end effector points towards the center of the circle of the arc. To have the frame of the end of the soft actuator module in the same orienta-

tion as the base frame, an additional multiplication with homogeneous transformation matrices with pure rotation  $R_z(\phi)$  and  $R_y(\theta)$  is needed, shown in equation 2.3 and 2.4 respectively.

$$T_{Rot-z}(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

$$T_{Rot-y}(\theta) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

## 2.2 Robot Specific Mapping

While the robot independent mapping gave a solution for mapping the piecewise constant curvature arc from configuration space to task space, the robot specific mapping will give a solution for mapping the actuator space to the configuration space.

In contrast to the independent mapping, the specific mapping is unique to the exact type of robot. The specific mapping requires measurements on the actual robot, by relating the inputs and outputs of actuators [9]. In case of the soft actuator module the inputs will be the air pressure, related to the pneumatic chamber lengths, resulting in the arc parameters as output.

### 2.2.1 Parameters

There are three parameters that define the shape and orientation of the soft actuator module. In the section 2.1 it is described how the parameters are defined in the frame convention. The curvature of the soft actuator module  $\kappa(\mathbf{q})$ , which is related to the angle  $\theta$  with  $\theta = \kappa s$ . The angle of the plane containing the arc  $\phi(\mathbf{q})$  and lastly, the length of the arc  $\ell(\mathbf{q})$ , which describes the length of the backbone of the soft actuator module. The centre denoted by H and shown in figure 1.2a. The dependency on  $\mathbf{q}$  describes that all the parameters are dependent on the joint parameters  $\ell_i$  with  $i \in [1, 2, 3, 4]$ .

### 2.2.2 Inverse Solution

To go from arc parameters to chamber lengths, which can be related to the applied air pressure, a modified version of the solution by Bryan and Walker (2006) is used. The design in their paper is based on three tendon lengths, but the solution can be applied to pneumatic actuators such as the soft actuator module and modified to a four chamber length case [5].

The solution is based on the angle at which the pneumatic chambers are placed inside the soft actuator. When using three equally spaced tendons, the angle between them is  $2/3\pi$ , as shown in their result. When using four tendons, or in case of the soft actuator module pneumatic chambers, this angle becomes  $1/2\pi$ . The angles then become:  $\phi_1 = \phi$ ,  $\phi_2 = \phi + 1/2\pi$ ,  $\phi_3 = \phi + \pi$  and  $\phi_4 = \phi + 3/2\pi$ .

To obtain the pneumatic chamber length, the simplified result from Bryan and Walker (2006), shown in equation 2.5, is used as a basis. It describes the length of the tendon based on the

main trunk length  $s$ , curvature  $\kappa$ , rotation  $\phi$ , distance from the centre of the trunk to the tendon  $d$  and the number of (cable) segments  $n$ .

$$\ell_i = 2n \sin\left(\frac{\kappa s}{2n}\right) \left(\frac{1}{\kappa} - d \sin(\phi_i)\right) \quad (2.5)$$

Equation 2.5 can be modified to work for a pneumatic chamber instead of a tendon. Parameter  $d$  then becomes the distance from the centre of the trunk to the centre of the pneumatic chamber and because pneumatic chambers are assumed to inhibit a constant continuous curvature, the numbers of cable segments is equal to infinity. Setting  $n \rightarrow \infty$  and substituting the appropriate values for  $\phi$ , the specific mapping for the four pneumatic chamber actuator is shown in equation 2.6.

$$f_{specific}^{-1} = \begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \\ \ell_4 \end{bmatrix} = \begin{bmatrix} \ell(1 - \kappa d \sin(\phi)) \\ \ell(1 - \kappa d \cos(\phi)) \\ \ell(1 + \kappa d \sin(\phi)) \\ \ell(1 + \kappa d \cos(\phi)) \end{bmatrix} \quad (2.6)$$

Equation 2.6 shows each pneumatic chamber length based on the arc parameters  $\kappa$ ,  $\phi$ ,  $d$  and  $\ell$ , but to achieve the required length a pneumatic actuation method is used. The relation between the two will be described in the next section.

### 2.2.3 Chamber Length to Pressure Relation

To relate the air pressure applied to the pneumatic chambers to the length of these pneumatic chambers, the chamber lengths need to be measured at different air pressures. A curve fitting technique is then used to obtain a continuous curve of the chamber length to pressure relation.

Both the actuated and the non-actuated pneumatic chamber lengths are measured. The effects inflicted (upon each other) by both the actuated and the non-actuated pneumatic chamber are used to create a more accurate simulation, and thus a more accurate control system.

## 2.3 Constraints

A prediction is made that the piecewise constant curvature only holds for a specific range of angles. By adding constraints to the system, it is made sure that the piecewise constant curvature will always hold for any output angle and that desired bending is realizable. The way these constraints are determined is by examining the physical soft actuator module and checking for which angles the piecewise constant curvature holds.

Another constraint is a kinematic constraint. For a single section the device is underactuated, this means it has less number of actuators than it has degrees of freedom (DoF). As a result, it can not follow all paths in the work space. This can be solved by increasing the number of actuators, thus adding more modules till the number of actuators equals the degrees of freedom. In that case, the robot becomes fully actuated.

### 3 Constant Curvature Kinematics

Using the modelling of section 2, kinematics of single and multi module cases can be derived.

#### 3.1 Single Module Kinematics

To get the actuator(s) in their desired position and motion, kinematics is needed. First, the forward kinematics for a single module is determined. This way the shape of the soft actuator module and end effector position is determined by setting the angles. Next, using differential kinematics the end effector velocity can be determined by the rate of change of the angles.

##### 3.1.1 Forward Kinematics

Using equation 2.2 the homogeneous transformation matrix can directly be extracted. Given the angle  $\phi$ , curvature  $\kappa$  and length  $\ell$  the end effector position in Cartesian space can be determined.

Taking equation 2.2 as the transformation matrix  $T_1^0(\mathbf{q})$ , the transformation from 0 to 1 depend on  $\mathbf{q} = [\kappa \ \phi]^T$ , with 0 being the base and 1 being the end effector of the (first) soft actuator module, denoted by  $P_1$ . Then, by multiplying  $T_1^0(\mathbf{q})$  with the location of the base will give the end effector position  $P_1$ , shown in (3.1), if the base of the soft actuator module was placed in the origin,  $\mathbf{O} = [0 \ 0 \ 0]^T$  aiming upwards in the  $+z$ -direction.

$$P_1 = T_1^0(\mathbf{q}) \cdot \mathbf{O} \quad (3.1)$$

##### 3.1.2 Forward Differential Kinematics

Differential kinematics relates the velocities of the joints,  $\dot{\mathbf{q}} = [\dot{\phi} \ \dot{\kappa}]^T$ , in joint space to the end effector velocity in task space, Cartesian space,  $\mathbf{v} = [\dot{x} \ \dot{y} \ \dot{z}]^T$ . This relation is given by the Jacobian of the robot, denoted by  $J$ .

By taking the time derivative of the transformation matrix  $T$ , the result can be used to determine the Jacobian.

Similar to the work of Hannan et. al. (2003), the only velocities of interest are those in the  $x$ ,  $y$ , and  $z$  direction. In such a case, if the time derivative of matrix  $T$  looks like equation 3.2, then  $J\dot{\mathbf{q}}$  can be constructed by taking specific elements of equation 3.2, shown in equation 3.3 [4].

$$\dot{T} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \quad (3.2)$$

$$J\dot{\mathbf{q}} = [\alpha_{14} \ \alpha_{24} \ \alpha_{34}]^T \quad (3.3)$$

Factoring out  $J$  from (3.3) gives the result in equation 3.4.

$$J = \begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \\ \beta_{31} & \beta_{32} \end{bmatrix} \quad (3.4)$$



With elements:

$$\beta_{11} = \frac{1}{\kappa}(-\sin(\phi) + \sin(\phi) \cos(\kappa s)) \quad (3.5)$$

$$\beta_{21} = -\frac{1}{\kappa}(\cos(\phi) - \cos(\phi) \sin(\kappa s)) \quad (3.6)$$

$$\beta_{31} = 0 \quad (3.7)$$

$$\beta_{12} = \frac{1}{\kappa^2}(\cos(\phi) - \cos(\phi) \sin(\kappa s)) + \frac{s}{\kappa} \cos(\phi) \sin(\kappa s) \quad (3.8)$$

$$\beta_{22} = \frac{1}{\kappa^2}(\sin(\phi) - \sin(\phi) \cos(\kappa s)) + \frac{s}{\kappa} \sin(\phi) \sin(\kappa s) \quad (3.9)$$

$$\beta_{32} = \frac{1}{\kappa^2} \sin(\kappa s) + \frac{s}{\kappa} \sin(\phi) \sin(\kappa s) \quad (3.10)$$

Using this Jacobian, the end effector velocity can be determined based on the joint velocity, shown in equation 3.11.

$$\mathbf{v} = J\dot{\mathbf{q}} \quad (3.11)$$

The Jacobian also serves as a basis for the inverse kinematics problem [5].

### 3.1.3 Inverse Differential Kinematics

Obtaining the joint angles based on the end effector position is called the inverse kinematics problem. Multiplying equation 3.11 with the inverse Jacobian on both sides is a solution, however, in this case the Jacobian is not square. This is solved by taking the pseudo inverse of the Jacobian. The pseudo inverse Jacobian, denoted by  $J^+$ , is defined as shown in equation 3.12.

$$J^+ = (J^T J)^{-1} J^T \quad (3.12)$$

This way the end effector velocities can be set, and the joint velocities be determined.

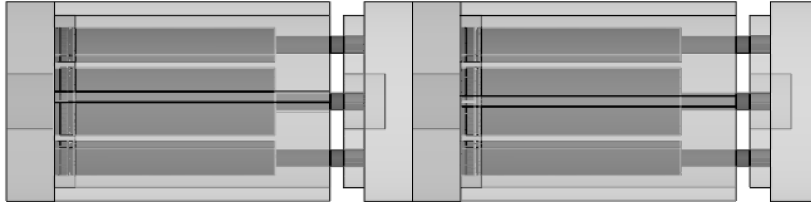
$$\dot{\mathbf{q}} = J^+ \mathbf{v} \quad (3.13)$$

As a single module system is underactuated, it cannot follow all arbitrary trajectories. The  $x$ ,  $y$  and  $z$  coordinates are dependent on each other. A certain  $x$  and  $y$  in the work space will result in a forced  $z$  coordinate. This does not add any benefit over using the less-complex forward kinematics method, and will thus not be tested on the physical soft actuator. However, the method described is the same as for a multi section inverse differential kinematics problem.

The current end-effector frame rotates together with  $\theta$  and  $\phi$ . As explained in section 2.1 an additional multiplication with  $R_z(\phi)$  and  $R_y(\theta)$  is needed to have the end-effector frame parallel to the base frame in the  $x$ ,  $y$  and  $z$ -direction. This will then put the velocities in a more intuitive frame. The velocities are then in the same direction as the base frame.

### 3.2 Multi Module Kinematics

Expanding on the single module, it is possible to determine the shape and end-effector position of a robot consisting of multiple soft actuator modules, as shown in figure 3.1. This can be done by setting the angles of individual modules, forward kinematics. By using inverse (differential) kinematics, it is possible to set the end effector position and velocities and calculate the angles from those, which is a more practical approach.



**Figure 3.1:** Illustration a multi module robot using their molds. Two modules are concatenated here with their internal pneumatic chambers visible in the dark shade.

#### 3.2.1 Forward Kinematics

Deriving the forward kinematics for a multiple module case can be achieved by multiplying the transformation matrices of single sections. This is possible because all modules are identical.

As shown in equation 3.14, by multiplying the homogeneous transformation matrix of each module will result in a homogeneous transformation matrix from base to the end effector of the multi module robot. Each single module has their own set of variables,  $q_i$ , which are  $\phi_i$  and  $\kappa_i$  with  $i$  denoting the section, from 0 to  $n$ .

$$T_n^0(\mathbf{q}) = T_1^0(\mathbf{q})T_2^1(\mathbf{q})\dots T_n^{n-1}(\mathbf{q}) \quad (3.14)$$

#### 3.2.2 Forward Differential Kinematics

Differential kinematics will be derived in the same manner has with a single module in section 3.1.2. The time derivative of  $T_n^0$  is taken, and following the same procedure, the Jacobian can be derived from this time derivative. Depending on the amount of modules used, the resulting Jacobian will differ.

#### 3.2.3 Inverse Differential Kinematics

With the use of multiple modules, the problem of an underactuated system can be solved. Together with inverse differential kinematics, it is possible to follow any arbitrary trajectory inside the workspace (if there are no physical constraints).

The inverse differential kinematics problem is solved in the same manner as the single module problem, with the use of the pseudo inverse Jacobian, as shown in section 3.1.3. Again, depending on the amount of modules used, the resulting pseudo inverse Jacobian will differ.

---

## 4 Implementation

The realization of the control system of the soft actuator module has been done in multiple, separate, steps. First, the physical soft actuator module (section 1.2) is tested in order to characterize the bending versus chamber length (section 2.2.3). After, the characteristics of the soft actuator module is determined, namely the relation of the elongation of the pneumatic chambers and the applied pressure. This relation is then used, together with equation 2.6 to determine the needed pressure to achieve the required bending angle(s). To verify the simulations and the calibration, the above is evaluated on the physical soft actuator module.

### 4.1 Soft Actuator Module

The soft actuator module has been created as described in chapter 1.2. The mold, shown in figure 1.2a and 1.2b has been 3D printed and assembled. The mold has then been filled with Ecoflex™ 00-50 Platinum Cure Silicone Rubber Compound by Smooth-On. The material was then vacuumed for 3 minutes to remove any air bubbles present. It was let to cure for 4 hours before taking the mold apart and applying the shielding.

The shielding makes sure the pneumatic chambers do not expand outwards, and the full deformation is in the elongated direction. The shielding has been made to fit the soft actuator module closely and only allow expansion in the upwards direction.

When referred to a 2D, scenario, chambers 1 and 3, which can be seen in figure 1.2a are connected to a pressure regulator (manual or voltage controlled). Chambers 2 and 4 in a 2D scenario are left unconnected. In this scenario when referred to the actuated chamber or actuated pneumatic chamber is the chamber with the controlled air pressure in it. The passive, or non-actuated, chamber is at vacuum to stay as close to the original 3.0 cm chamber length.

In a 3D scenario, all 4 chambers are connected to pressure regulators. Furthermore, the 2D and 3D scenario use the same algorithm, only with different calibrations. The above steps are first done in a 2D implementation ( $\phi = 0$ ), which is followed by a 3D implementation.

### 4.2 2D Implementation

#### 4.2.1 Calibration

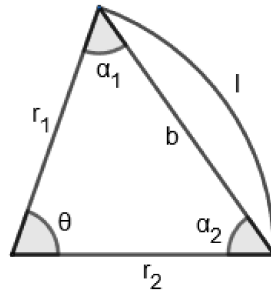
The calibration is performed to establish the relation between the pneumatic chamber length and the applied pressure. The way this has been performed is by placing specific physical markers on the soft actuator module. A series of increasing known pressures are applied into one chamber, and the bending is examined. This bending is captured with a digital camera, and digitally the size of both the actuated pneumatic chamber and the opposite pneumatic chamber (chamber (1) and (3) in figure 1.2a) is determined. The measuring setup can be seen in figure 4.2.

Physical markers have been placed 5 mm above and 2 mm below the pneumatic chamber, this margin has been introduced to avoid puncturing the chamber with the needle markers. Needle markers have been used to avoid measuring errors of the shielding moving independently from the internal soft actuator module. The total distance from the bottom marker to the top marker in rest position ( $\theta = 0$ ) equals those two margins plus the chamber length of 30mm, resulting in 37mm.

When the digital images have been acquired, the markers, including the above mentioned margins, are measured. A program called “ImageJ” is used, which allows measuring aspects of the image when the scale is known. The process of measuring the chambers can be seen in figure 4.3.

In this program the lengths (1) and (2) in figure 4.3 and the angle  $\theta$  are measured. The lengths (1) and (2), referred to as “link length” is used, together with the piecewise constant curvature assumption, to determine the pneumatic chamber lengths, as shown in figure 4.1 and equation 4.1. Both chamber lengths are determined to model how they affect each other, as described in section 2.2.3.

$$\ell = r_1 \cdot \theta = \frac{b/2}{\sin(\theta/2)} \cdot \theta \quad (4.1)$$



**Figure 4.1:** Visualisation link length  $b$  and arc (chamber) length  $l$ .  $r_1 = r_2$  and  $\alpha_1 = \alpha_2$ .

To verify the repeatability of the measurement, all calibration measurements have been done three times. Once all three measurements are done, the average is taken and a polynomial fit is applied to this average for both the actuated and passive chamber measurements. The degree of polynomial will be determined by comparing the fitted curve to the measurement points. This fit is then used as the pressure to chamber length relation and to see the effects of the actuator chamber on the passive chamber.

To validate the piecewise constant curvature assumption, the lengths  $r_1$  and  $r_2$ , from figure 4.1, are measured on the physical actuator for multiple angles. The piecewise constant curvature assumption holds if these two lengths are equal (not taking into account measurement uncertainties).

The above has been done for a bending to the left by actuating chamber 1 and also for a bending to the right by actuating chamber 3. As described in section 4.1, the other, opposite, chamber has been set at vacuum.

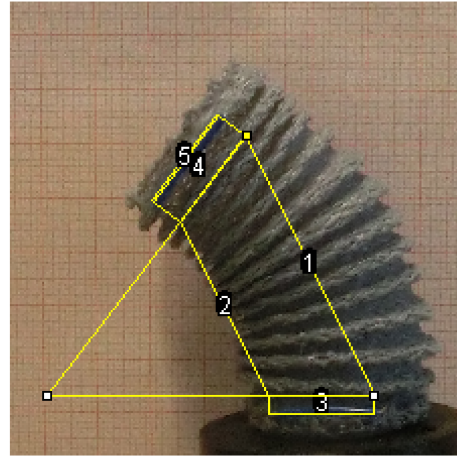
#### 4.2.2 Validation

The validation of the simulations and calibrations has been done in a similar fashion to the calibration. The same measuring setup, shown in figure 4.2, has been used.

For the 2D scenario a wanted angle  $\theta$  is chosen, the control algorithm will then calculate the chamber lengths based on this angle. The chamber lengths are then related to an air pressure, obtained by the calibration, as explained above. This pressure is then applied manually and



**Figure 4.2:** The calibration experimental set-up including the soft actuator module, scale (ruler), and pressure read-out.



**Figure 4.3:** Structural mapping of the soft actuator module using ImageJ.

the bending is examined by capturing an image. This image is then processed using ImageJ and the angle is determined and compared to the input angle.

### 4.3 3D Implementation

#### 4.3.1 Calibration

To allow 3D motion, an extension on the 2D calibration has to be performed. In the 3D motion multiple chambers will work together to achieve an effective bending ( $\theta$ ) and rotation ( $\phi$ ). As with the 2D, an actuated chamber will affect the other chamber lengths. These effects will need to be modelled.

As this study currently focuses on the 2D motion, only a prediction can be made about the 3D motion. Because when elongating the pneumatic chamber, its natural bending direction is in the opposite direction of the position of the chamber in the soft actuator module. But when  $\phi$  is not equal to  $n \cdot 90^\circ$  with  $n \in \mathbb{N}$  the bending is not in the natural direction.

The prediction is made that more pressure is needed to deviate from this natural direction to achieve the same bending  $\theta$ .

#### 4.3.2 Validation

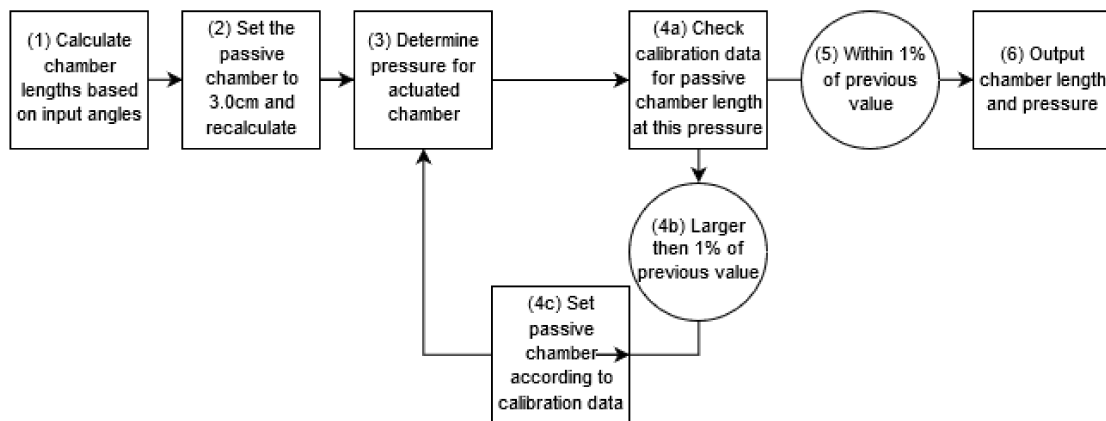
There will be no in-depth validation of the 3D bending. However, a proof-of-concept of the 3D motion will be given.

### 4.4 Control Algorithm

With the math described in section 2 and the calibration data, a merged algorithm is created. The algorithm has one or two input variables (in addition to the physical characterisation parameters of the soft actuator module),  $\theta$  only for a 2D scenario, and both  $\theta$  and  $\phi$  for the 3D scenario. The control algorithm will output an air pressure level.

Equation 2.6 is used to determine the pneumatic chamber length, but relies on the assumption that these chambers can shrink. However, this is not the case with the soft actuator module in this study. Together with the calibration data of both the actuated chamber and passive chamber a practical solution has been created to counteract this limitation.

First, based on the input parameters, the chamber lengths will be calculated according to (2.6) ((1) in figure 4.4). Then the passive chamber length will be set to 3.0cm, the smallest physically possible chamber length. The difference between the calculated value and the smallest physical chamber length will be added to the main backbone length (this can be seen as the red line in figure 5.1 and 6.7),  $\ell$ , and with this new main length new chamber lengths will be calculated (2). With the calibration data, the corresponding pressure will be determined for the actuated chamber length (3). At this pressure level, the passive chamber length will be determined (4a). If this deviates more than 1% from the previous smallest passive chamber length (which will be 3.0cm in the first iteration) the passive chamber length is adjusted (4c), and with actuated chamber length is again calculated, until the passive chamber length is within 1% of the previous value (5). Finally, the output pressure of the actuated chamber is given (6). This method is referred to as “Passive Chamber Compensation” in this report and the full schematic of this can be seen in figure 4.4. The control algorithm MATLAB code can be found in appendix C.

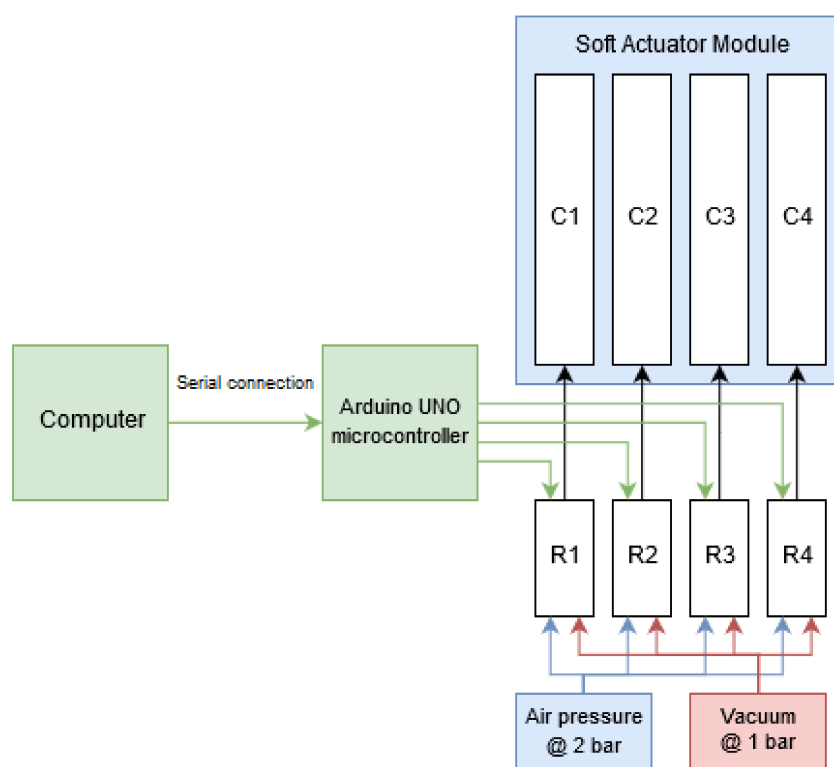


**Figure 4.4:** Passive chamber compensation diagram

## 4.5 Control System

The control algorithm, as described above, has to be written in C and loaded on a microcontroller, e.g. an Arduino UNO.

With a serial connection to a Windows/Unix based computer the angle  $\theta$  (and  $\phi$  for the 3D scenario) can be set, in which the Arduino will execute the control algorithm, which will result in the required air pressure for each chamber. This air pressure is then realized by setting the correct analog output voltage on one of the designated GPIO pins connected to the FESTO pressure regulator (VEAB-L-26-D7-Q4-V1-1R1). This voltage is then read by the FESTO pressure regulator, which will apply the correct pressure to a specific chamber in the soft actuator module. An overview schematic can be seen in figure 4.5. Depending on the scenario (2D/3D), the pneumatic chambers C1-C4 have been connected to pressure regulators R1-R4 respectively. Each pressure regulator has been connected to 2 bars of air pressure and 1 bar of vacuum. As described above, the Arduino is then able to control the pressure or vacuum of the chambers using these pressure regulators.



**Figure 4.5:** Control system overview diagram

## 5 Simulations

The purpose of the simulations is to validate the modelling of the soft actuator module of chapter 2 together with the kinematics of chapter 3.

The simulations are done using the same order as described in section 4. First, a 2D scenario is simulated, which is followed with a 3D scenario. After the single module has been simulated in both 2D and 3D, this is expanded into a 3D multi module simulation.

The purpose of these simulations is to verify the kinematics of the robot. It also gives a visual representation of the shape of the module together with its chamber lengths. This could help to spot errors before the physical testing stage.

### 5.1 2D Single Module

The 2D simulations are based on the equations in section 3.1 with  $\phi = 0$ , the physical limitations, described in section 4.4, have also been included in this simulation. A 2D Cartesian system is created in which the base of the module has been placed in  $[0 \ 0]^T$ . The backbone of the module is then plotted according to the homogeneous transformation matrix given in equation 2.2 given a specific  $\theta$  in the range 0 to  $1/2\pi$ .  $\ell$  is set to 3.00cm, equal to the physical chamber length.  $\phi$  is set to 0, as mentioned above. A continuous curvature has been achieved by plotting 240 points for  $s$  in (2.2), from 0.0cm to 3.0cm, with a resolution of 80 dots per cm.

The pneumatic chambers have been visualized in the same manner as the main backbone, but their bases have been placed in  $[d \ 0]^T$  and  $[-d \ 0]^T$  with  $d$  being the distance from the backbone, the centre of the soft actuator module, to the centre of the pneumatic chamber, which equals 0.8cm.

#### 5.1.1 Forward Kinematics

Figure 5.1a - 5.1d show a 2D plot of the main trunk, displayed in red, with the left and right chamber, displayed in black at the angles,  $\theta$ , of  $0^\circ$ ,  $15^\circ$ ,  $45^\circ$  and  $90^\circ$ , respectively. An angle  $\theta$  of  $0^\circ$  is considered the rest state, however when  $\theta = 0$  is applied to relation with  $\kappa = \theta/s$ , this results in a division by zero in equation 2.2. This has been prevented by adding the code below, which results in the angle  $\theta$  being, effectively, 0 while not causing a division by zero. The MATLAB code can be found in appendix A with  $\phi = 0$ .

```

1  if kappa == 0           % if 'kappa' equals zero, set to
2      kappa = 10^-32;    % an extremely small value, which
3  end                    % is effectively zero

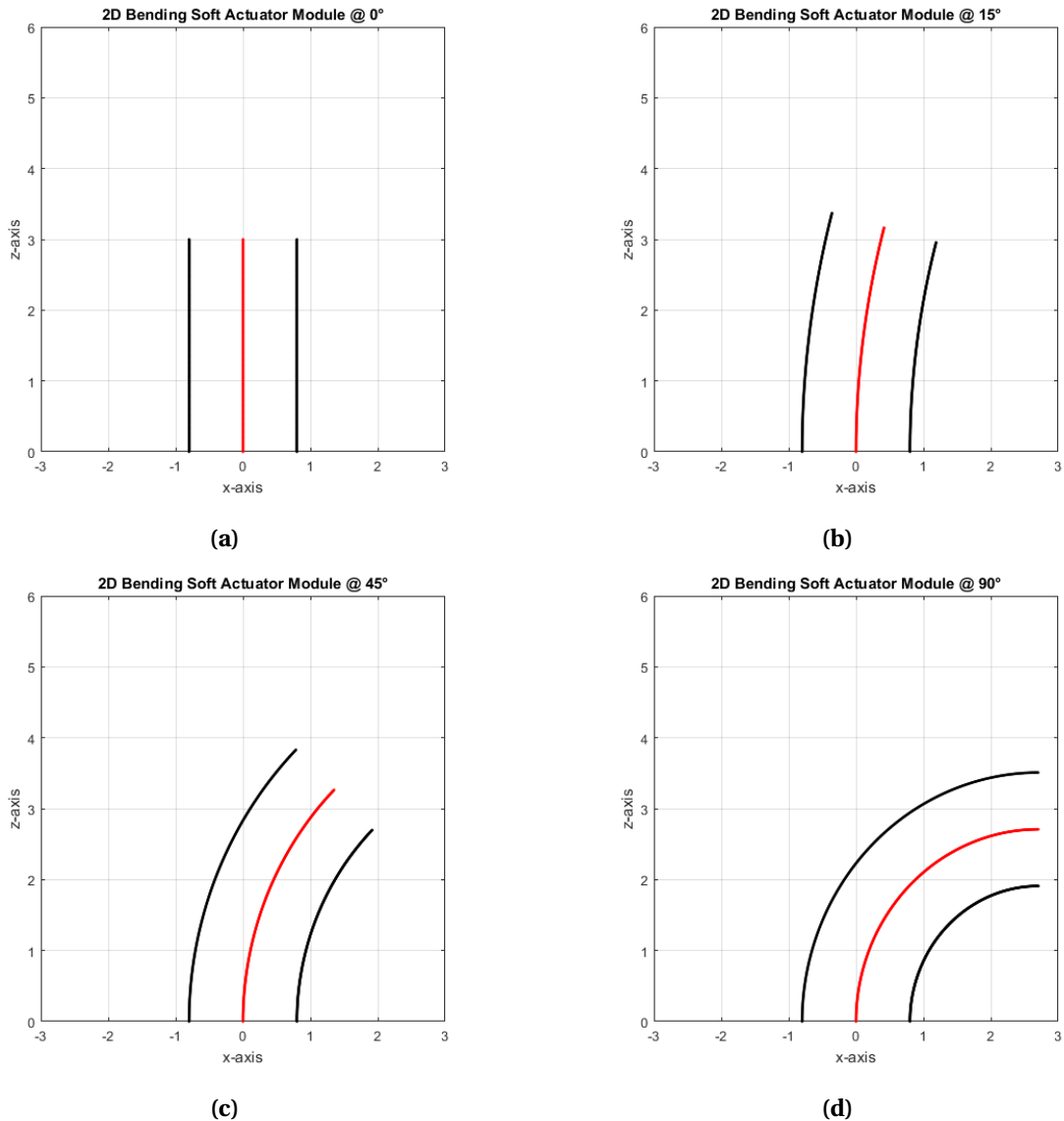
```

### 5.2 3D Single Module

The 3D simulations are done in the same fashion as in section 5.1, but in a 3D Cartesian system, and placing the base of the module in  $[0 \ 0 \ 0]^T$ . The backbone of the module has been plotted as described in section 5.1, with the same resolution,  $\ell$  and  $\theta$  range. But, in the 3D scenario  $\phi$  has been made variable from 0 to  $2\pi$  instead of equal, and fixed, to 0.

The four bases of the pneumatic chambers have been placed around the backbone's base at  $90^\circ$  intervals at a distance  $d$  of 0.8cm.





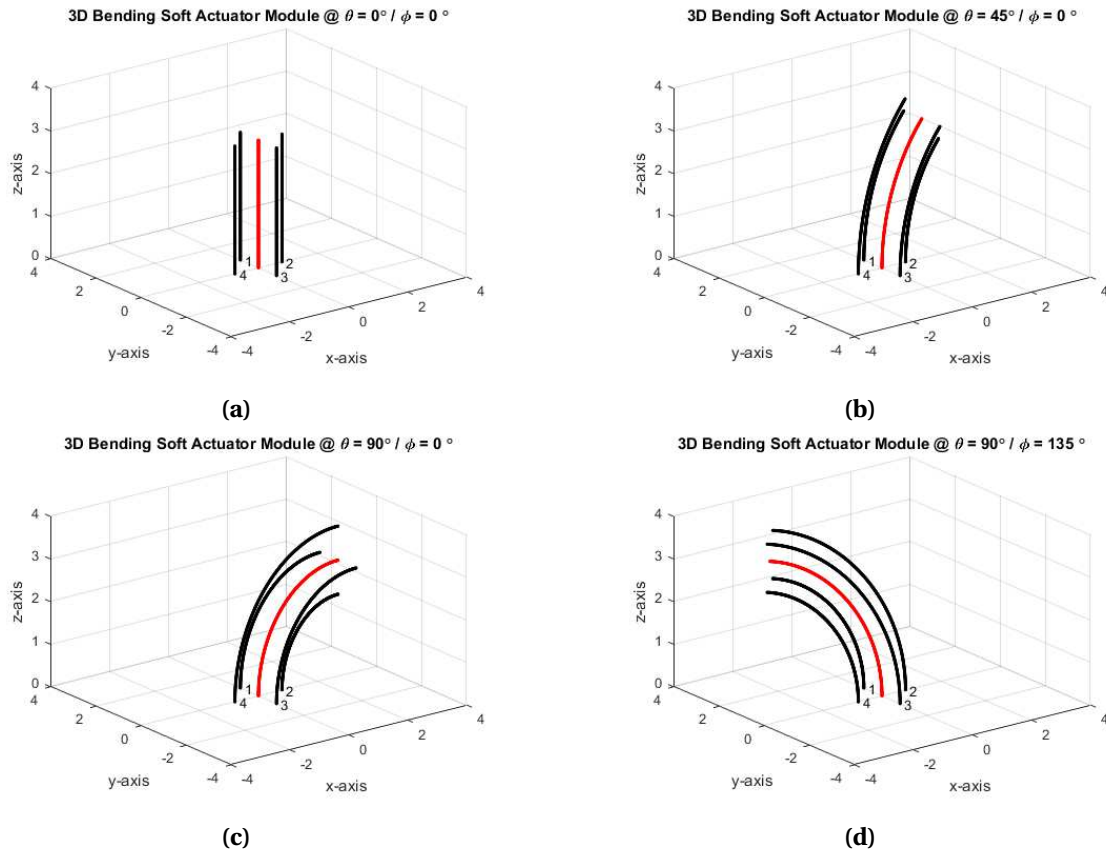
**Figure 5.1:** 2D simulations of the soft actuator module bending at multiple angles  $\theta$

### 5.2.1 Forward Kinematics

Figure 5.2a - 5.2d show a 3D plot of the main trunk, displayed in red, with the four pneumatic chambers in the configuration described above. A multiple of configurations are displayed for display-purposes. Similar to section 5.1.1, in case of the rest state, that is  $\theta = 0$ , the same code is used to prevent a division by zero. The MATLAB code can be found in appendix A.

### 5.3 3D Multi Module

The simulations in section 5.2 serve as a basis for the simulations for a 3D multi module endoscope. Every module inside the multi module endoscope can be seen as an individual module, such as simulated in section 5.2. The simulations of multi module has been performed in the same 3D Cartesian space as the 3D single module simulations (see section 5.2). The base of the first module has been placed in  $[0 \ 0 \ 0]^T$ , similar to the 3D single module. But, every succeeding module is placed on the frame of the end effector of the previous module. This can be seen as multiplying the individual homogeneous transformation matrices, shown in equation



**Figure 5.2:** 3D simulations of the soft actuator module bending at multiple angles of  $\theta$  and  $\phi$

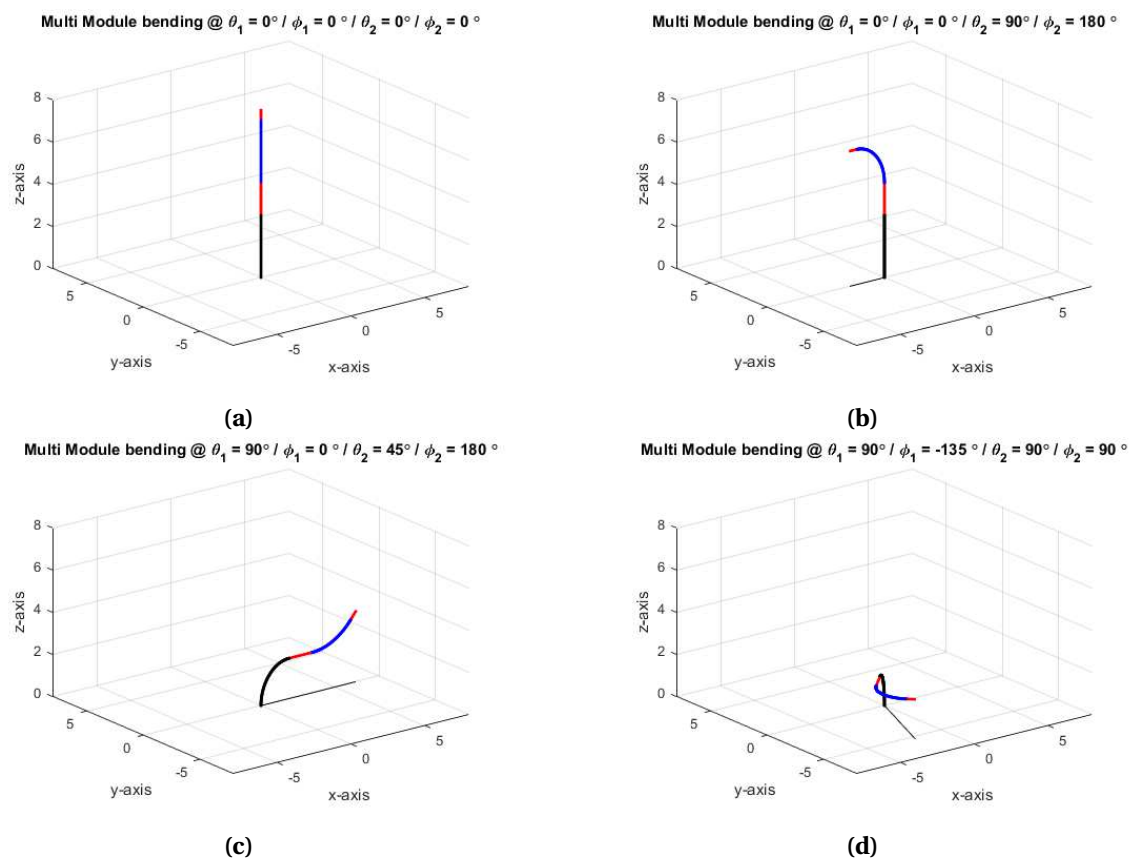
3.14. For the simulations for the multi module robot a 2 module case has been created. This way the simulations stay relatively simple compared to a higher number of modules, while still being able to show the possibilities of a multi module robot.

Because the soft actuator module has a non-bending part at the top and bottom, this has also been modelled in the simulations by adding a pure translation transformation matrix in between the homogeneous transformation matrices of the first and second module. This is 0.5cm at the top, and 1.0cm at the bottom.

The pneumatic chamber lengths of each module are also calculated based on the parameters of each individual module. This can be compared with the simulations shown in section 5.2. The individual chambers are not shown in the 3D multi module simulations to avoid clutter.

### 5.3.1 Forward Kinematics

Figure 5.3a - 5.3d show the simulation results for the multi module robot following the criteria stated above. Multiple configurations are displayed to show the possibilities of this robot. The rotation  $\phi_i$ , with  $i \in [1, 2]$ , is done at the beginning of each new module. The considered rest state has been displayed in figure 5.3a. The MATLAB code for the 3D simulations can be found in appendix B.



**Figure 5.3:** Multi module simulations showing multiple configurations with the black module being the first one, the blue module the second one and the red pieces are the non-bending parts of the actuators. A line from the origin to the projection of the end effector on the  $x-y$  plane can be seen as the thin black line in the  $x-y$  plane.

## 6 Results

The results are obtained as described in chapter 4. These are compared to the simulations, the control algorithm, if applicable.

### 6.1 Validation: Piecewise Constant Curvature Assumption

Measurements have been done according to the method described in section 4.2.1. The radius lengths  $r_1$  and  $r_2$ , as shown in figure 4.1, have been measured and the absolute difference between these two is displayed in figure 6.1 for the left and right bending. Figures 6.2 and 6.3 show the soft actuator at multiple angles of  $\theta$ .

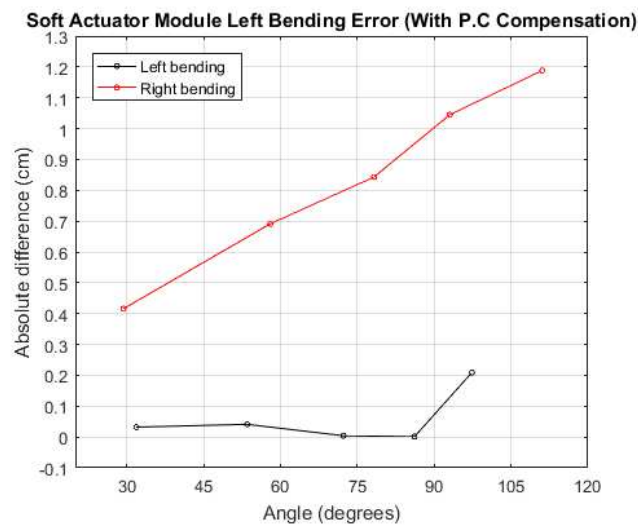


Figure 6.1: Absolute difference between lengths  $r_1$  and  $r_2$



Figure 6.2: Left bending at multiple angles of  $\theta$



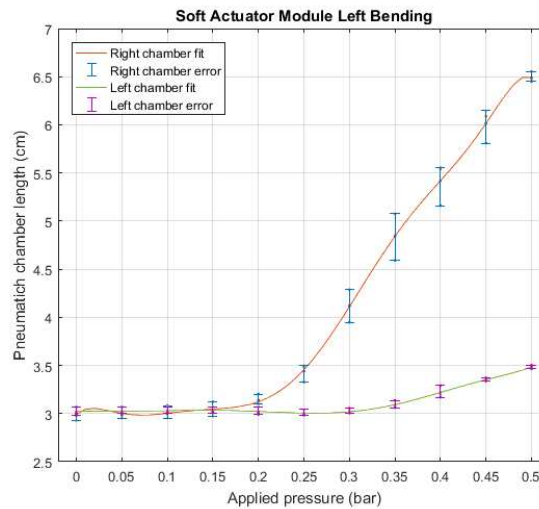
Figure 6.3: Right bending at multiple angles of  $\theta$

### 6.2 Chamber Length - Pressure Relation

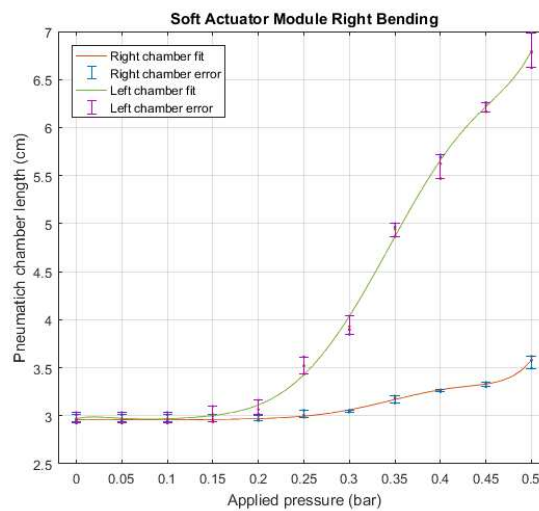
#### 6.2.1 2D: Calibration

Section 4.2.1 describes the process of obtaining the calibration results and calculations required for it. The results of these measurements and calculations are displayed in figure 6.4 and

figure 6.5. These figures include the original measuring data, with the maximum and minimum displayed by the errorbar. An 8th degree polynomial has been fitted to the mean value of the measurement data. An 8th degree polynomial is chosen because it would have the least deviation from the mean value points. Figure 6.6 shows the fitted curves of both actuated chambers, included the absolute difference between them.



**Figure 6.4:** 2D bending pressure and chamber length relation for a left bending



**Figure 6.5:** 2D bending pressure and chamber length relation for a right bending

With a maximum difference of 0.49 cm at 0.35 bar for the left bending (figure 6.4) and 0.36 cm at 0.50 bar for the right bending (figure 6.5). This difference in chamber length is equal to a  $17^\circ$  and  $12^\circ$  difference in bending ( $\theta$ ) respectively at that pressure level.

In figure 6.6 the maximum absolute difference of the two curves is 0.30 cm, at that specific pressure (0.50 bar) will result in a  $\theta = 10^\circ$  bending difference.

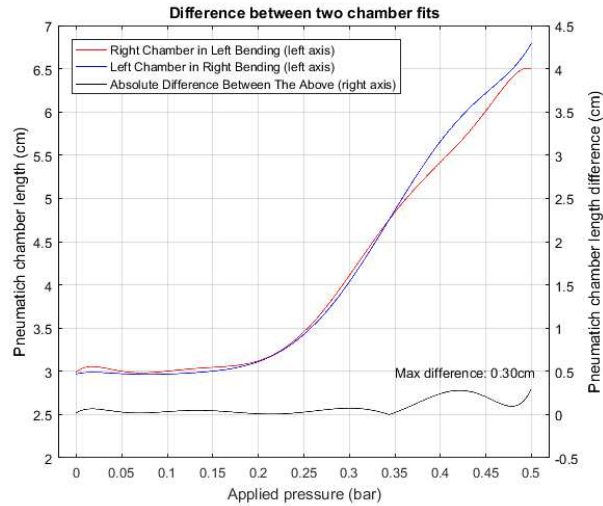


Figure 6.6: Difference of the fit of actuated chambers of figure 6.4 and 6.5

### 6.2.2 2D: Validation

Following the method described in 4.2.2 the validation of the 2D scenario is executed. The results, the difference of the input bending angle  $\theta$  (supplied to the control algorithm) and the actual bending, are shown in figure 6.7a for the left bending and figure 6.7b for the right bending of the soft actuator module. Both figures include the error when assuming the passive chamber is at 3.0cm, referred to as “Chamber error without compensation” and the error when compensating the acutated chamber length based on the passive chamber length, referred to as “Chamber error with compensation”.

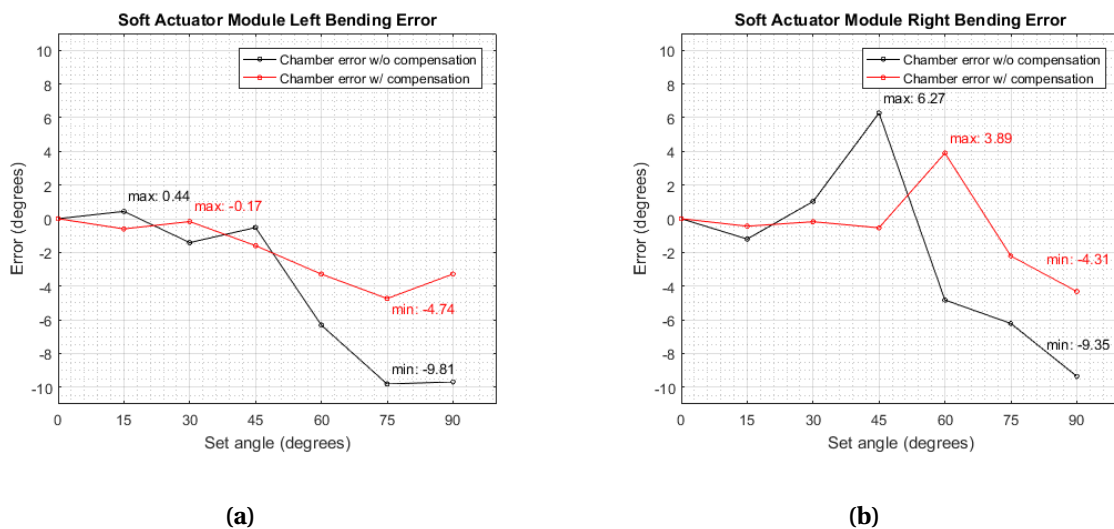
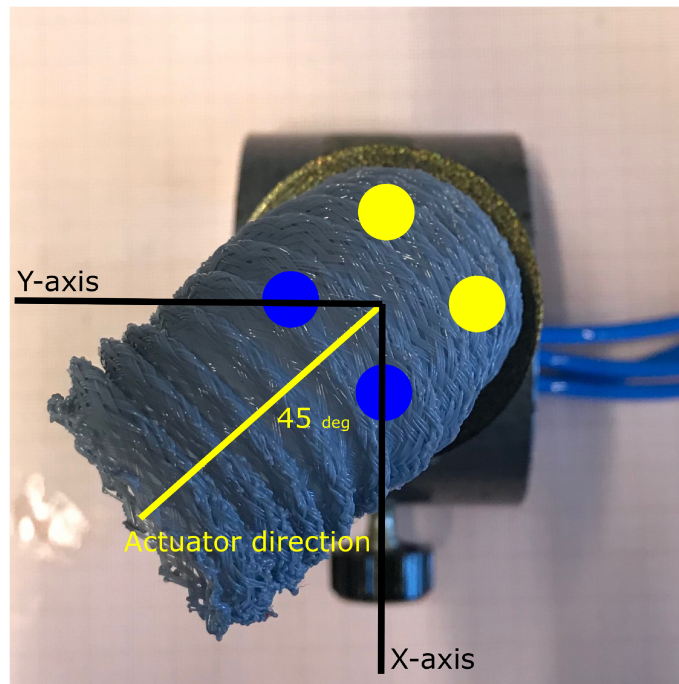


Figure 6.7: Difference between the input angle and the measured angle for left and right bending

### 6.3 3D: Proof of Concept

Figure 6.8 shows a top view of the soft actuator module with an overlay showing the actuated pneumatic chambers, as yellow circles, the passive pneumatic chambers, as blue circles, the  $x$  and  $y$ -axis and the direction of bending  $\phi$  as the yellow line. Figure 6.9 shows the perspective when looking from the  $x$ -axis and  $y$ -axis in a 3D scenario.



**Figure 6.8:** Proof of concept 3D bending



(a)



(b)

**Figure 6.9:** Non-antagonistic bending view from (a) the  $x$ -direction when actuating two chambers and (b) the  $y$ -direction when actuating two chambers.

## 7 Discussion

The results of the piecewise constant curvature validation show two different results. One, on the left bending, show that the difference between the radius lengths  $r_1$  and  $r_2$  is minimal, proving that the assumption of piecewise constant curvature is valid. However, the right bending shows a larger difference between the two lengths, showing that the assumption does not hold as well compared to the left bending. When looking at figure 6.3 it can also be observed that compared to figure 6.2 the curvature is less uniform. The soft actuator module elongates more in the right direction and the bending is mostly in the middle, with the top part (referring to the part that is not fixed to the base) containing almost no bending. This is likely caused by the non-uniformity and non-symmetry of the shielding, as it is shrunk by heating it by hand, and thus prone to imperfections. The error does increase with increased bending for both left and right bending, especially the left bending shows an increase in error after  $90^{circ}$ . This confirms the prediction made in section 2.3 regarding that the piecewise constant curvature will only hold for a certain range of  $\theta$ .

Comparing the calibration results of the left and right bending show similar trend, this can also be seen in figure 6.6 where the fitted curves are compare, however, similar to the validation results, the slight differences can be caused by the non-uniformity and non-symmetry of the shielding as explained above. The validation results of the individual left and right curves show that with the passive chamber compensation the accuracy of the soft actuator module control is greatly increased. It also shows that a larger bending relates to a larger error. The error in the right bending (figure 6.7b) shows a error which deviates from the expected error. A possible explanation for this could be the non-uniformity and non-symmetric construction of the shielding, as explained above in the first paragraph of this chapter.

The use of curve fitting also brings error into the experiments, because the fit is not exact, there is a deviation from the actual measuring results. An increase in accuracy can be achieved by increasing the number of measurement points and by experimenting with different curve fittings.

If the physical soft actuator module, including the shielding, could be made uniformly and symmetrically, every pneumatic chamber's bending could be considered identical requiring only measurements on one pneumatic chamber..

Furthermore, there are multiple causes of uncertainty in the measurements. First, human factor while manually determining the angles and link- and radius lengths in ImageJ. Second, the images taken are slightly distorted due to the camera's field-of-view, and consequently a non-flat image. Last, the exact location of the internal pneumatic chambers can't be directly observed. The markers used will also introduce a uncertainty into the measurements.

The repeatability of the calibration measurements can be assessed based on the differences between the multiple measurements. This difference is larger for the left bending compared to the right bending. The error between the minimum and maximum value is considered to be significant, resulting in noticeable angle differences. When concatenating multiple modules the overall error will increase linearly with the number of modules. For this case and the application of the soft actuator module the repeatability is considered to be low.



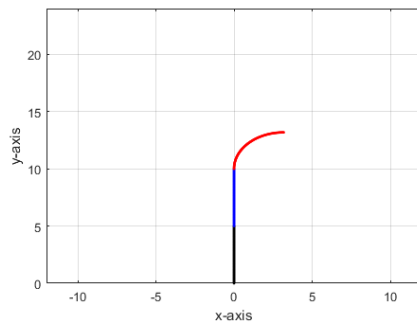
Observation on the prediction made in section 4.3 that the soft actuator needs more pressure to deviate from this natural bending is observed to be the case. With the  $\theta = 90^\circ$  bending with a  $\phi = 45^\circ$  rotation shown in figure 6.9 a pressure of around 0.53 bar was needed. But when bend in a so-called natural direction, explained in section 4.3, the soft actuator module needed around 0.39 bar.

Furthermore, as shown with the 3D proof of concept, with the control of multiple chambers a desired rotation can be achieved. However, more research into the characteristics and calibration should be done to correctly control the 3D motion. Once extended into 3D, the single module serves as a solid basis for a multi module with the kinematics described in section 3 or with the application described below.

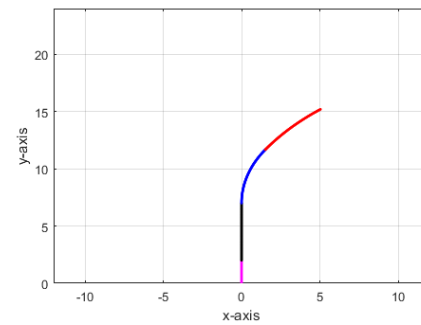
As an application for this study, based on the forward kinematics and control algorithm of a single module, multiple segment endoscope was guided through a predefined path. Within this study it has been given the name Insertion Dependent Angle Memory Control (IDAMC). IDAMC works as follows, the (human) operator will, with visual feedback from the camera in the tip of the most distal module, control the most distal module of a multi module robot. The operator is able to control the angle  $\theta$  and  $\phi$ . The operator is also able to insert the endoscope further into the the patient. In this case, the angles  $\theta$  and  $\phi$  from the most distal module will translate to the adjacent module closer based on the insertion. This way the angle stays at a certain location in the human body, making it possible to transverse the endoscope around complex curvatures.

A 2D ( $\phi = 0$ ) MATLAB demo, MATLAB code can be found in appendix D, has been made to show the above. It remembers the set angle  $\theta$  in an array based on the insertion parameter  $q$  in cm. It has a resolution of 0.5cm, which can be increased at the costs of a larger memory array. Figure 7.1a - 7.1c shows a three module endoscope that is mounted on a non-acutated base at multiple insertions. If the base of the endoscope is inserted more or less, the angle  $\theta$  is translated over two sections in such a way that the effective angles equal the set angles. Figure 7.1d shows the overlap for the insertions  $q = 0$  to  $q = 10$  with a interval of 1. For these simulations a module length of 5 cm is taken and assumed that there is a constant curvature bending throughout the full module length.

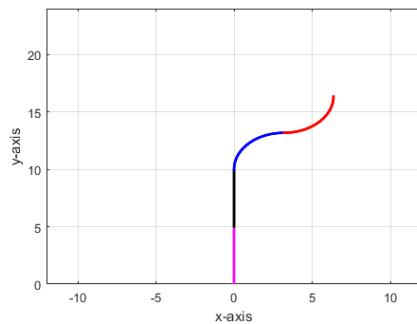
Improving on this design is to be able to more accurately follow the set angles. This can be done by decreasing the module length, and to have the same endoscope length increase the number of modules with the same factor.



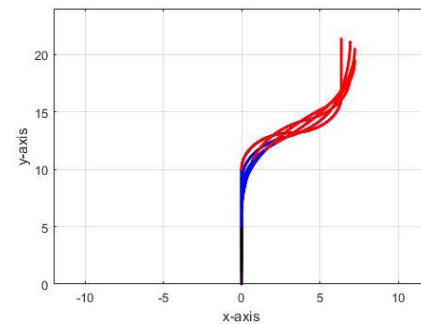
(a) Insertion of  $q = 0$  cm and  $\theta = 90^\circ$



(b) Insertion of  $q = 2$  cm. Angle  $\theta$  has been partially translated into the second section (blue module).



(c) Insertion of  $q = 5$  cm (one module length). Angle  $\theta$  has been fully translated into the second section (blue module), and a new angle has been set of  $\theta = -90^\circ$  (red module).



(d) Overlap of the insertions  $q = 0$  cm to  $q = 10$  cm with 1 cm intervals

**Figure 7.1:** 2D simulation of path following application with three modules (red, blue and black) and insertion base rod (magenta line)

---

## 8 Conclusions and Recommendations

### 8.1 Conclusions

The main research question in this study was “How to control a soft actuator module for endoscopic application”. Moreover to controlling the soft actuator, possible applications and future perspectives of the module was also assessed.

The main research question has been successfully answered by creating a control system that is based on changing the pneumatic chamber lengths by applying an air pressure. Required for this is a understanding of the relation between the applied air pressure and the length of the pneumatic chamber. This relation has been experimentally determined, and validated with a maximum error of  $9.61^\circ$  and  $9.35^\circ$  for the left and right bending respectively. An additional characterization was added to this control system, referred to as passive chamber compensation, and was able to reduce the error to a maximum of  $4.74^\circ$  and  $4.31^\circ$  respectively.

A proof of concept of 3D motion has been given to show that by actuating multiple modules a effective bending and rotation can be achieved. However, a more complex calibration is needed to use 3D motion in the control system.

With the kinematics of chapter 3 this single module serves as a solid foundation for a multi module solution and can be used as one of multiple identical modules in a multi module robot to allow more complex maneuvers. A possible endoscopic application is given as Insertion Dependent Angle Memory Control (IDAMC). Furthermore, with the use of inverse kinematics given in section 3.2.3, based on the pseudo inverse Jacobian, it is theoretically possible to set an end effector location and obtain the joint angles. This also allows velocity control in the  $x$ ,  $y$  and  $z$  direction.

### 8.2 Recommendations

This section is dedicated to give recommendations for future research and aspects of this study what could be improved.

A major improvement for this study would be the production of a uniform and symmetric soft actuator module, mainly the shielding. This could be done by improving the current production process or developing a new shielding. One could also incorporate a construction in the actually soft actuator that would constrain the sideways expansion, e.g. rigid rings inside soft actuator.

Furthermore, while a better calibration method could be developed. This can be done by improving on the current design using image based calibration with the use of physical markers or developing a new design based on e.g. lasers or other positional sensing systems. The resolution could also be increased by choosing smaller intervals in the applied air pressure.

In addition, to create the curve based on this calibration different curve fittings could be used. Comparing these curves and experimenting with them could give a better insight in what method would work best.

Currently the control algorithm is validated using a manual valve. The output pressure, calculated by the control algorithm, is applied manually using this valve. The voltage controlled pressure regulator was ordered at the beginning of this study, but was not delivered in time. The control system however is described in section 4.5.

The Insertion Dependent Angle Memory Control could be used together with MR imaging and machine learning to let an algorithm set the angles based on the insertion. This can be done to assist the operator during surgery or suggest paths (e.g. implementing path planning) to the operator.

## A Single Module Simulation MATLAB Code

```

1  % Jorn Jansen
2  % BSc Electrical Engineering
3  % University of Twente
4  % Enschede, the Netherlands
5  %
6  % Single Soft Actuator Module Simulation Code
7  % Part of EE BSc Thesis
8
9  %% parameters
10 l=3;           % length of arc (cm)
11 sk = 1;       % length to point on arc (cm), if l=s then
                % endpoint is taken
12 k = 80;       % resolution
13 phi = 0*2*pi/360 ; % rotational about the z axis (degrees)
14 theta = 103*2*pi/360; % rotation about the y axis (degrees)
15 kappa = theta/sk; % curvature
16 d = 0.8;      % diameter of the tube
17
18 if kappa == 0 % singularity prevention
19     kappa = 10^-32;
20 end
21
22 r = 1/kappa;  % radius of curve
23
24 %% inverse dependend mapping
25 l1 = sk*(1-kappa*d*sin(phi)); % calculation of chamber lengths
26 l2 = sk*(1-kappa*d*cos(phi));
27 l3 = sk*(1+kappa*d*sin(phi));
28 l4 = sk*(1+kappa*d*cos(phi));
29
30 Rz = [cos(-phi) -sin(-phi) 0 0; sin(-phi) cos(-phi) 0 0; 0 0 1 0; 0
        0 0 1]; % rotational matrix to keep the individual chambers
                % from rotating around the z-axis
31
32 %% physical restriction solver (old method)
33 if l1 < 3.000
34     d1 = 3.000-l1;
35     l = l + d1;
36 end
37
38 if l2 < 3.000
39     d2 = 3.000-l2;
40     l = l + d2;
41 end
42
43 if l3 < 3.000
44     d3 = 3.000-l3;
45     l = l + d3;

```

```

46 end
47
48 if l4 < 3.000
49     d4 = 3.000-l4;
50     l = l + d4;
51 end
52
53 kappa = theta/l;
54
55 if kappa == 0
56     kappa = 10^-32;
57 end
58
59 l1 = l*(1-kappa*d*sin(phi)); % recalculation chamber lengths
60 l2 = l*(1-kappa*d*cos(phi));
61 l3 = l*(1+kappa*d*sin(phi));
62 l4 = l*(1+kappa*d*cos(phi));
63
64 %% calculations
65 figure('pos',[10 10 510 360])
66 for s = 0:l/k:l % main backbone plot
67     T1 = [cos(phi)*cos(kappa*s) -sin(phi) cos(phi)*sin(kappa*s) (
68           cos(phi)*(1-cos(kappa*s)))/kappa; sin(phi)*cos(kappa*s) cos
69           (phi) sin(phi)*sin(kappa*s) (sin(phi)*(1-cos(kappa*s)))/
70           kappa; -sin(kappa*s) 0 cos(kappa*s) (sin(kappa*s))/kappa; 0
71           0 0 1];
72     V = [0; 0; 0; 1];
73     Q = T1*V;
74     plot3(Q(1), Q(2), Q(3), 'r. ');
75     grid on;
76     hold on;
77 end
78
79 for s = 0:l/k:l % pneumatic chamber 1 plot
80     T1 = [cos(phi)*cos(kappa*s) -sin(phi) cos(phi)*sin(kappa*s) (
81           cos(phi)*(1-cos(kappa*s)))/kappa; sin(phi)*cos(kappa*s) cos
82           (phi) sin(phi)*sin(kappa*s) (sin(phi)*(1-cos(kappa*s)))/
83           kappa; -sin(kappa*s) 0 cos(kappa*s) (sin(kappa*s))/kappa; 0
84           0 0 1];
85     V1 = [0; d; 0; 1];
86     Q1 = T1*Rz*V1;
87     plot3(Q1(1), Q1(2), Q1(3), 'k. ');
88     hold on;
89 end
90
91 for s = 0:l/k:l % pneumatic chamber 2 plot
92     T1 = [cos(phi)*cos(kappa*s) -sin(phi) cos(phi)*sin(kappa*s) (
93           cos(phi)*(1-cos(kappa*s)))/kappa; sin(phi)*cos(kappa*s) cos
94           (phi) sin(phi)*sin(kappa*s) (sin(phi)*(1-cos(kappa*s)))/
95           kappa; -sin(kappa*s) 0 cos(kappa*s) (sin(kappa*s))/kappa; 0
96           0 0 1];

```

```

85     V2 = [d; 0; 0; 1];
86     Q2 = T1*Rz*V2;
87     plot3(Q2(1), Q2(2), Q2(3), 'k. ');
88     hold on;
89 end
90
91 for s = 0:1/k:1           % pneumatic chamber 3 plot
92     T1 = [cos(phi)*cos(kappa*s) -sin(phi) cos(phi)*sin(kappa*s) (
           cos(phi)*(1-cos(kappa*s)))/kappa; sin(phi)*cos(kappa*s) cos
           (phi) sin(phi)*sin(kappa*s) (sin(phi)*(1-cos(kappa*s)))/
           kappa; -sin(kappa*s) 0 cos(kappa*s) (sin(kappa*s))/kappa; 0
           0 0 1];
93     V3 = [0; -d; 0; 1];
94     Q3 = T1*Rz*V3;
95     plot3(Q3(1), Q3(2), Q3(3), 'k. ');
96     hold on;
97 end
98
99 for s = 0:1/k:1           % pneumatic chamber 4 plot
100    T1 = [cos(phi)*cos(kappa*s) -sin(phi) cos(phi)*sin(kappa*s) (
           cos(phi)*(1-cos(kappa*s)))/kappa; sin(phi)*cos(kappa*s) cos
           (phi) sin(phi)*sin(kappa*s) (sin(phi)*(1-cos(kappa*s)))/
           kappa; -sin(kappa*s) 0 cos(kappa*s) (sin(kappa*s))/kappa; 0
           0 0 1];
101    V4 = [-d; 0; 0; 1];
102    Q4 = T1*Rz*V4;
103    plot3(Q4(1), Q4(2), Q4(3), 'k. ');
104    hold on;
105 end
106
107 text(0,d,0, ' 1');       % label chambers
108 text(d,0,0, ' 2');
109 text(0,-d,0, ' 3');
110 text(-d,0,0, ' 4');
111
112 xlabel('x-axis');       % label axis
113 ylabel('y-axis');
114 zlabel('z-axis');
115
116 title(['3D Bending Soft Actuator Module @ \theta = ' num2str(theta
           *360/(2*pi)) '\circ / \phi = ' num2str(phi*360/(2*pi)) '\circ '
           ]);
117
118 xlim([-4 4]);
119 ylim([-4 4]);
120 zlim([0 4]);

```

## B Multi Module Simulation MATLAB Code

```

1 % Jorn Jansen
2 % BSc Electrical Engineering
3 % University of Twente
4 % Enschede, the Netherlands
5 %
6 % Single Soft Actuator Module Simulation Code
7 % Part of EE BSc Thesis
8
9 %% general parameters
10 k = 20;           % resolution of plot
11 T = 1;           % simulation sample time
12
13 %% parameters link 1
14 l1 = 3;           % length of arc (cm)
15 sk1 = l1;        % length to point on arc (cm), if l=s then
                    % endpoint is taken
16 phi1 = -3/4*pi;  % rotational about the z axis (degrees)
17 theta1 = 1/2*pi; % rotation about the y axis (degrees) (ONLY
                    % POSITIVE ANGLES)
18
19 kappal = theta1/sk1; % curvature
20
21 if kappal == 0     % prevent singularity
22     kappal = 1e-32;
23 end
24 r1 = 1/kappal;    % radius of curve
25
26 %% parameters link 2
27 l2 = 3;           % length of arc (cm)
28 sk2 = l2;        % length to point on arc (cm), if l=s then
                    % endpoint is taken
29 phi2 = 1/2*pi;    % rotational about the z axis (degrees)
30 theta2 = 1/2*pi; % rotation about the y axis (degrees)
31
32 kappa2 = theta2/sk2; % curvature
33
34 if kappa2 == 0    % prevent singularity
35     kappa2 = 1e-32;
36 end
37 r2 = 1/kappa2;   % radius of curve
38
39 %% Non-bending parts modelling between module 1 and 2
40
41 NB1 = [1 0 0 0; 0 1 0 0; 0 0 1 1.5; 0 0 0 1]; % matrix Non-
                    % Bending part #1
42 NB2 = [1 0 0 0; 0 1 0 0; 0 0 1 0.5; 0 0 0 1]; % matrix Non-
                    % Bending part #2
43

```



```

44 %% plots
45 figure('pos',[10 10 510 360])
46 for s = 0:1/k:l1          % plot first section
47     T1 = [cos(phi1)*cos(kappal*s) -sin(phi1) cos(phi1)*sin(kappal*s)
           ] (cos(phi1)*(1-cos(kappal*s)))/kappal; sin(phi1)*cos(
           kappal*s) cos(phi1) sin(phi1)*sin(kappal*s) (sin(phi1)*(1-
           cos(kappal*s)))/kappal; -sin(kappal*s) 0 cos(kappal*s) (sin
           (kappal*s))/kappal; 0 0 0 1];
48     V = [0; 0; 0; 1];          % origin
49
50     if s == 0
51         Q11 = T1*V;          % begin first section
52     end
53     if s == l1
54         Q12 = T1*V;          % end second section
55     end
56
57     Q1 = T1*V;
58     plot3(Q1(1), Q1(2), Q1(3), 'k. ');
59     grid on;
60     hold on;
61 end
62
63 for s2 = 0:1/k:l2          % plot second section
64     T2 = [cos(phi2)*cos(kappa2*s2) -sin(phi2) cos(phi2)*sin(kappa2*
           s2) (cos(phi2)*(1-cos(kappa2*s2)))/kappa2; sin(phi2)*cos(
           kappa2*s2) cos(phi2) sin(phi2)*sin(kappa2*s2) (sin(phi2)
           *(1-cos(kappa2*s2)))/kappa2; -sin(kappa2*s2) 0 cos(kappa2*
           s2) (sin(kappa2*s2))/kappa2; 0 0 0 1];
65
66     if s2 == 0
67         Q21 = T1*NB1*T2*V; % begin second section
68     end
69     if s2 == l2
70         Q22 = T1*NB1*T2*V; % end second section
71     end
72
73     Q2 = T1*NB1*T2*V;
74     plot3(Q2(1), Q2(2), Q2(3), 'b. ');
75     hold on
76 end
77
78 NBF1 = plot3([Q12(1) Q21(1)], [Q12(2) Q21(2)], [Q12(3) Q21(3)], 'r'
           ); % plot non-bending part #1
79
80 hold on
81
82 Q3 = T1*NB1*T2*NB2*V; % calculation non-bending part #2
83
84 NBF2 = plot3([Q22(1) Q3(1)], [Q22(2) Q3(2)], [Q22(3) Q3(3)], 'r');
           % plot non-bending part #2

```

```
85
86 set ([NBF1 NBF2], 'LineWidth', 2);
87
88 hold on;
89
90 plot3([0 Q3(1)], [0 Q3(2)], [0 0], 'k'); % plot "shadow" on x-y
    plane
91
92 xlabel('x-axis');
93 ylabel('y-axis');
94 zlabel('z-axis');
95
96 xlim([-8 8]);
97 ylim([-8 8]);
98 zlim([0 8]);
99
100 title(['Multi Module bending @ \theta_1 = ' num2str(theta1*360/(2*
    pi)) '\circ / \phi_1 = ' num2str(phi1*360/(2*pi)) '\circ / \
    theta_2 = ' num2str(theta2*360/(2*pi)) '\circ / \phi_2 = '
    num2str(phi2*360/(2*pi)) '\circ']);
```

## C 2D Control System MATLAB Code

```

1  % Jorn Jansen
2  % BSc Electrical Engineering
3  % University of Twente
4  % Enschede, the Netherlands
5  %
6  % Single Soft Actuator Module Simulation Code
7  % Part of EE BSc Thesis
8
9  %% parameters
10 l=3;           % length of arc (cm)
11 sk = 1;       % length to point on arc (cm), if l=s then
                % endpoint is taken
12 k = 80;       % resolution
13 phi = 0*2*pi/360 ; % rotational about the z axis (degrees)
14 theta = 90*2*pi/360; % rotation about the y axis (degrees)
15 kappa = theta/sk; % curvature
16 d = 0.8;      % diameter of the tube
17
18 kappa = kappaconstraint(kappa);
19
20 r = 1/kappa;  % radius of curve
21
22 mcl = 3.0;    % minimum chamber length
23 Lold = 0;     % initial value Lold
24
25 Rz = [cos(-phi) -sin(-phi) 0 0; sin(-phi) cos(-phi) 0 0; 0 0 1 0; 0
        0 0 1];
26
27 %% inverse dependend mapping
28 L = chamberlength(l, kappa, d, phi);
29
30 %% physical restriction solver
31 done = 0;
32 count = 0;
33
34 %% inverse dependend mapping
35 L = chamberlength(l, kappa, d, phi);
36 linit = chamberconstraint(L, mcl, l);
37
38 %% physical restriction solver
39
40 while done == 0 % run until minimum length is within 1% of the
                % previous minimum length
41
42     l = chamberconstraint(L, mcl, l); % calculate backbone length
                % with compensation
43
44     kappa = theta/l;

```

```

45
46 kappa = kappaconstraint(kappa);      % check for singularity
47
48 L = chamberlength(l, kappa, d, phi);  % calculate chamber
    lengths
49
50 pa = pressureoutput(l);              % calculate pressure needed
51 mcl = lengthoutputpassive(pa);       % check passive chamber
    length
52
53 l = chamberconstraint(L, mcl, l);     % calculate new backbone
    length
54
55 kappa = theta/l;
56
57 kappa = kappaconstraint(kappa);      % check for singularity
58
59 L = chamberlength(l, kappa, d, phi);  % calculate chamber
    lengths again with the new l
60
61 Lnew = max(L);                       % determine Lnew, this is the max in a
    2D case
62
63 if ((0.9999*Lold<=Lnew) && (Lnew<=1.0001*Lold))
64     done = 1;                         % if within 1% done = 1
65 else
66     done = 0;                         % if not within 1% done = 0
67 end
68
69 Lold = Lnew;                         % Set old value as the new value
70
71 count = count + 1;                  % keep track of the amount of
    itteration needed
72 end
73
74 PA = pressureoutput(max(L)) % output pressure to command window
75
76 %% functions
77 function L = chamberlength(l, kappa, d, phi)
78
79     % calculate chamber lengths
80
81     L(1) = l*(1-kappa*d*sin(phi));
82     L(2) = l*(1-kappa*d*cos(phi));
83     L(3) = l*(1+kappa*d*sin(phi));
84     L(4) = l*(1+kappa*d*cos(phi));
85
86 end
87
88 function l = chamberconstraint(L, mcl, l)
89

```

```

90     % calculate backbone length
91
92     if min(L) < mcl
93         dl = mcl - min(L);
94         l = l + dl;
95     end
96
97 end
98
99 function kappa = kappaconstraint(kappa)
100
101     % check for singularity
102
103     if kappa == 0
104         kappa = 10^-32;
105     end
106
107 end
108
109 function ansp = pressureoutput(l)
110
111     % pressure output based on chamber length
112
113     PQ = [0.000 0.050 0.100 0.150 0.200 0.250 0.300 0.350 0.400
114           0.450 0.500];
115
116     ML1 = [3.038 3.038 3.038 3.101 3.164 3.517 4.041 5.002 5.717
117            6.262 6.982; PQ];
118
119     ML2 = [2.924 2.924 2.924 2.940 3.009 3.434 3.896 4.866 5.469
120            6.166 6.623; PQ];
121
122     ML3 = [2.951 2.951 2.951 2.998 3.017 3.614 3.847 4.966 5.690
123            6.257 6.780; PQ];
124
125     ML = mean([ML1(1,:); ML2(1,:); ML3(1,:) ],1);
126
127     pl = polyfit(PQ,ML(1,:),8); % fitted curve
128
129     syms p;
130     eqn = (pl(1)*(p)^8)+(pl(2)*(p)^7)+(pl(3)*(p)^6)+(pl(4)*(p)^5)+(
131           pl(5)*(p)^4)+(pl(6)*(p)^3)+(pl(7)*(p)^2)+(pl(8)*(p))+(pl(9)
132           ) == 1; % inverse of the fitted curve
133
134     solp = solve(eqn, p, 'real', true); %pressure output
135
136     solpa = vpa(solp);
137
138     solpadouble = double(solpa);
139
140     ansp = solpadouble((solpadouble >= 0.2) & (solpadouble <= 0.5))
141           ; % get real solution withint the range of 0.2 bar to 0.5
142           bar
143
144 end
145
146 function l = lengthoutputpassive(pa)

```

```
133
134 % calculate passive chamber length based on pressure applied to
      the
135 % actuated chamber
136
137 PQ = [0.000 0.050 0.100 0.150 0.200 0.250 0.300 0.350 0.400
      0.450 0.500];
138
139 ML1 = [3.071 3.071 3.071 3.071 3.071 3.041 3.001 3.075 3.200
      3.345 3.487; PQ];
140 ML2 = [3.007 3.007 3.018 3.021 3.009 2.986 3.055 3.136 3.291
      3.339 3.502; PQ];
141 ML3 = [2.982 3.001 3.001 3.008 2.991 2.984 3.008 3.062 3.170
      3.371 3.464; PQ];
142
143 ML = mean([ML1(1,:); ML2(1,:); ML3(1,:) ],1);
144
145 pl = polyfit(PQ,ML(1,:),8);
146 l = (pl(1)*(pa)^8)+(pl(2)*(pa)^7)+(pl(3)*(pa)^6)+(pl(4)*(pa)^5)
      +(pl(5)*(pa)^4)+(pl(6)*(pa)^3)+(pl(7)*(pa)^2)+(pl(8)*(pa))
      +(pl(9));
147
148 end
```

## D 2D Insertion Dependent Angle Memory Control MATLAB Code

```

1 % Jorn Jansen
2 % BSc Electrical Engineering
3 % University of Twente
4 % Enschede, the Netherlands
5 %
6 % Single Soft Actuator Module Simulation Code
7 % Part of EE BSc Thesis
8
9 %% general parameters
10 k = 20;           % resolution
11 d = 1;           % diameter of the tube to pressure chamber
12
13 %% base section parameters
14 O = [0; 0; 0; 1]; % location base
15 q = 0;
16
17 %% section 1 parameters
18 l1 = 5;           % length of arc (cm)
19 sk1 = l1;         % length to point on arc (cm), if l=s then
    endpoint is taken
20 phi1 = 0;        % rotational about the z axis (degrees)
21 theta1 = 0;      % rotation about the y axis (degrees)
22 kappal = theta1/sk1; % curvature
23
24 if kappal == 0    %prevent singularity
25     kappal = 10^-32;
26 end
27
28 r1 = 1/kappal;   % radius of curve
29
30 l11 = sk1*(1-kappal*d*sin(phi1)); % lengths of the pressure
    chambers,
31 l12 = sk1*(1-kappal*d*cos(phi1)); % these can be related to the
    pressure
32 l13 = sk1*(1+kappal*d*sin(phi1)); % applied.
33 l14 = sk1*(1+kappal*d*cos(phi1));
34
35 %% section 2 parameters
36 l2 = l1;         % length of arc (cm)
37 sk2 = l2;         % length to point on arc (cm), if l=s then
    endpoint is taken
38 phi2 = 0;        % rotational about the z axis (degrees)
39 theta2 = 0;      % rotation about the y axis (degrees)
40 kappa2 = theta2/sk2; % curvature
41
42 if kappa2 == 0    %prevent singularity

```

```

43     kappa2 = 10^-32;
44 end
45
46     r2 = 1/kappa2;           % radius of curve
47
48     l21 = sk2*(1-kappa2*d*sin(phi2)); % lengths of the pressure
    chambers,
49     l22 = sk2*(1-kappa2*d*cos(phi2)); % these can be related to the
    pressure
50     l23 = sk2*(1+kappa2*d*sin(phi2)); % applied.
51     l24 = sk2*(1+kappa2*d*cos(phi2));
52
53 %% section 3 parameters
54     l3 = l1;                 % length of arc (cm)
55     sk3 = l3;               % length to point on arc (cm), if l=s then
    endpoint is taken
56     phi3 = 0;              % rotational about the z axis (degrees)
57     theta3 = 0;            % rotation about the y axis (degrees)
58     kappa3 = theta3/sk3;    % curvature
59
60     if kappa3 == 0          %prevent singularity
61         kappa3 = 10^-32;
62     end
63
64     r3 = 1/kappa3;         % radius of curve
65
66     l31 = sk3*(1-kappa3*d*sin(phi3)); % lengths of the pressure
    chambers,
67     l32 = sk3*(1-kappa3*d*cos(phi3)); % these can be related to the
    pressure
68     l33 = sk3*(1+kappa3*d*sin(phi3)); % applied.
69     l34 = sk3*(1+kappa3*d*cos(phi3));
70
71 %% memory
72     n = 2;
73     Q = [0:1/n:50];
74     THETA = [Q; zeros(1,50*n+1)]; % create theta array to
    store angles for multiple displacements
75
76     THETA(2, 10*n+1:15*n) = 1/2*pi/(l1*n); % set test angle theta
77     THETA(2, 15*n+1:20*n) = -1/2*pi/(l1*n); % set test angle theta
78
79 %% visualisation
80     figure('pos',[10 10 510 360])
81     for q = 10:-1:0 % overlap insertions q = 10 till q = 0
82         theta1 = sum(THETA(2, q*n+1:(q+l1)*n));
83         theta2 = sum(THETA(2, (q+l1)*n+1:(q+l1+l2)*n));
84         theta3 = sum(THETA(2, (q+l1+l2)*n+1:(q+l1+l2+l3)*n));
85
86     kappa1 = theta1/l1;
87     kappa2 = theta2/l2;

```



```

88 kappa3 = theta3/l3;
89
90 if kappa1 == 0           %prevent singularity
91     kappa1 = 10^-32;
92 end
93
94 if kappa2 == 0           %prevent singularity
95     kappa2 = 10^-32;
96 end
97
98 if kappa3 == 0           %prevent singularity
99     kappa3 = 10^-32;
100 end
101
102 T0 = [1 0 0 0; 0 1 0 0; 0 0 1 q; 0 0 0 1]; % base section
      visualisation
103 V = T0*O;
104
105 plot([O(1) V(1)], [O(3) V(3)], 'm', 'LineWidth',2); % plot base
      of endoscope
106
107 grid on;
108 hold on;
109
110 for s1 = 0:1/k:l1           % visualisation section
111     1
      T1 = [cos(phi1)*cos(kappa1*s1) -sin(phi1) cos(phi1)*sin(kappa1*
            s1) (cos(phi1)*(1-cos(kappa1*s1)))/kappa1; sin(phi1)*cos(
            kappa1*s1) cos(phi1) sin(phi1)*sin(kappa1*s1) (sin(phi1)
            *(1-cos(kappa1*s1)))/kappa1; -sin(kappa1*s1) 0 cos(kappa1*
            s1) (sin(kappa1*s1))/kappa1; 0 0 0 1];
112     Q1 = T0*T1*O;
113     plot(Q1(1), Q1(3), 'k. ');
114     hold on;
115 end
116
117 for s2 = 0:1/k:l2           % visualisation section
118     2
      T2 = [cos(phi2)*cos(kappa2*s2) -sin(phi2) cos(phi2)*sin(kappa2*
            s2) (cos(phi2)*(1-cos(kappa2*s2)))/kappa2; sin(phi2)*cos(
            kappa2*s2) cos(phi2) sin(phi2)*sin(kappa2*s2) (sin(phi2)
            *(1-cos(kappa2*s2)))/kappa2; -sin(kappa2*s2) 0 cos(kappa2*
            s2) (sin(kappa2*s2))/kappa2; 0 0 0 1];
119     Q2 = T0*T1*T2*O;
120     plot(Q2(1), Q2(3), 'b. ');
121     hold on;
122 end
123
124 for s3 = 0:1/k:l3           % visualisation section
125     3

```

```
125 T3 = [cos(phi3)*cos(kappa3*s3) -sin(phi3) cos(phi3)*sin(kappa3*
        s3) (cos(phi3)*(1-cos(kappa3*s3)))/kappa3; sin(phi3)*cos(
        kappa3*s3) cos(phi3) sin(phi3)*sin(kappa3*s3) (sin(phi3)
        *(1-cos(kappa3*s3)))/kappa3; -sin(kappa3*s3) 0 cos(kappa3*
        s3) (sin(kappa3*s3))/kappa3; 0 0 0 1];
126 Q3 = T0*T1*T2*T3*O;
127 plot(Q3(1), Q3(3), 'r. ');
128 hold on;
129 end
130
131
132 xlabel('x-axis');
133 ylabel('y-axis');
134
135 xlim([-12 12]);
136 ylim([0 24]);
137
138 end
```

---

## Bibliography

- [1] H. Abidi, G. Gerboni, M. Brancadoro, J. Frascarelli, A. Diodato, M. Cianchetti, H. Wurdemann, K. Althoefer, and A. Menciassi. Highly dexterous 2-module soft robot for intra-organ navigation in minimally invasive surgery. *Int J Med Robotics Comput Assist Surg.*, 14, 2018.
- [2] K. Althoefer. Antagonistic actuation and stiffness control in soft inflatable robots. *Nature Reviews Materials*, 3:76–77, 6 2018.
- [3] M. Cianchetti, T. Ranzani, G. Gerboni, I. D. Falco, C. Laschi, and A. Menciassi. Stiff-flop surgical manipulator: mechanical design and experimental characterization of the single module. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3576–3581, 11 2013.
- [4] M. W. Hannan and I. D. Walker. Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. *Journal of Field Robotics*, 20(2):45–63, 2 2003.
- [5] B. A. Jones and I. D. Walker. Kinematics for multisection continuum robots. *IEEE Transactions on Robotics*, 22(1):43–57, 2 2006.
- [6] M. Manti, V. Cacucciolo, and M. Cianchetti. Stiffening in soft robotics: A review of the state of the art. *IEEE Robotics Automation Magazine*, 23(3):93–106, 9 2016.
- [7] N. Simaan, K. Xu, W. Wei, A. Kapoor, P. Kazanzides, R. Taylor, and P. Flint. Design and integration of a telerobotic system for minimally invasive surgery of the throat. *The International Journal of Robotics Research*, 28(9):1134–1153, 2009.
- [8] K. K. Smith and W. M. Kier. Trunks, tongues, and tentacles: Moving with skeletons of muscle. *American Scientist*, 77(1):28–35, 1 1989.
- [9] R. J. Webster III and B. A. Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 22(13):1661–1683, 2010.