Project Report*

*As a fulfilment to the course: "Project course name", D7039E & E7032E. Lecturer: Jan van Deventer.

Martin Blaszczyk, Edward Cedegård, Niklas Dahlquist, Edward Källstedt, Albin Martinsson, Måns Norell
Computer Science, Electrical and Space Engineering Dept.
Luleå University of Technology
Luleå, Sweden
{marbla-6, edwced-4, nikdah-6, edwkll-7, mnsnor-5, albmar-6}@student.ltu.se

***Abstract*—**

I. Introduction

A. Background and Motivation

Arrowhead is an initiative from Luleå University of Technology to create a unifying framework which can enable embedded devices to intergrate and interoperate services in an open-network environment. This framework and its approach will strongly conribute to the reduction of design and engineering efforts in the industry.

As part of the Engineering Project course for Master's students in Computer Science, Control and Electrical Engineering at Luleå University of Technology the goal is to implement and demonstrate how the Arrowhead Framework can be used in a factory setting using autonomous ground robots. This project report aims to give an overview of the thought process, workflows and results of the project.

The proposed solution is a robot which by implementing machine vision algorithms can navigate the factory floor while also being able to pick up objects using it's arm and gripper. It utilizes the Arrowhead framework to integrate with the other parts of the model factory.

B. Contributions

Our proposed solution is an example of how the Arrowhead framework is utilized in a scaled factory setting together with a ground robot. The design is small and modular making it easy for future improvements and easy evaluation of the Arrowhead framework alongside third-party open-source solutions. Secondly we propose a solution consisting of colored lines and QR-codes and how the robot can follow a path using a RGB-camera instead of the conventional sensors such as IR-sensors. With this camera solution the possibilities of navigating the factory floor increase compared to the basic conventional following solutions.

Structure

The overall structure of the report...

II. Arrowhead

Arrowhead

What is it?

Extras

More in depth about the strucutre of the Arrowhead implementation. Flow diagrams etc.

III. Communication

Communication

Unlike most servo motors Dynamixel uses digital packet communication instead of pulse-width modulation (PWM) signals. By grouping data into different packages and transmitting it as a bundle. This in turn opens up for many possibilities since the motors responds to commands, have built in control functions together with memory allocation on the motor itself. Dynamixel recommends two different setups for communication, either by using a Tri-state-buffer to construct your own communication bridge based on Universal asynchronous receiver-transmitter setup (UART). Or by using their own solution called U2D2. In essense, the U2D2 uses the same protocol but also enables for USB communication between the controller and Dynamixels [1].

A. UART full duplex to half duplex

UART communication scheme works by connecting the transmitter pin (Tx) of the Jetson Nvidia Nano microcontroller (MCU) directly to the receiver pin (Rx) of the motors and vice versa. UART communication can be setup into three different ways of communication. Full duplex, half duplex and simplex. Full duplex allows for data communication both ways simultaneously. Half duplex only allows for one way communication at any given time, by either sending or receiving. Simplex communication is static, meaning communication is always one direction.
Full duplex and half duplex are not directly compatible. If the MCU and Dynamixel motors are connected directly, since the MCU sends and receive data simultaneously it will occupy the transmission line indefinitely and the Dynamixel won't be able to tell when the messages stop. This phenomena is known as chatter. The solution for this is to implement a tri-state-buffer circuit converting the full system to half duplex.

B. Tri-state-buffer

A tri-state-buffer dictates when the MCU is allowed to communicate with the motors. This explained by figure **??**. By controlling the direction pin (GPIO pin), we can direct the flow

of data. If the direction pin is pulled high we simultaneously allow for transmission from the MCU (Tx) to the motors and disable the MCU receiver pin (Rx). A crucial part is timing, the direction pin must be pulled high long enough for the data to transfer through the buffer otherwise information is lost. A logic analyzer will give this information and is recommended to use in conjugation with the tri-state-buffer for tuning.

C. Data package - structure of message

The protocol allows for a size of 8 bits per message between the MCU and motors. The two types message structures is represented by the figures below III-C1 and III-C2. For more information how to interpret the structure use dynamixels own documentation ().

1) Instruction package: Instruction package carries commands from the controller to the motor. An abundance of support is included, from changing statics parameters like torque, baud-rate or id of each individual unit or different actions, for example move to this location with this speed. The structure of the message is represented in figure III-C1 and table I describes the structure from left to right.

TABLE I
EXPLANATION OF PARAMETERS IN INSTRUCTION PACKADGE

$OXFF$	Indicates start of incoming packet
ID	Specific ID of each unit
$Length$	Length of message
$Param$	Used for additional information
$Checksum$	Calculate length of packet

Fig. 1. Structure of instruction message.

2) Status package: Status packet contains the response from the Dynamixel to the controller after receiving a instruction. The structure is almost the same as the instruction package. However, one difference is the the message confirms the status of sent instructions including an error if one occurs. The structure is represented in figure III-C2 and Dynamixels support page [2] gives more information how to read the error messages.

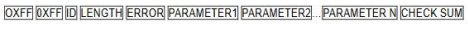

Fig. 2. Structure of return message.

D. Motor communication

Dynamixel provides a lot of support for lower level programming using instructions and reading return messages with documentation support, as stated above . However, there is also support for higher level and more user friendly programming languages like Python, C and third-party libraries by using Dynamixels own software development kit called DYNAMIXEL SDK [3].

IV. MODELING

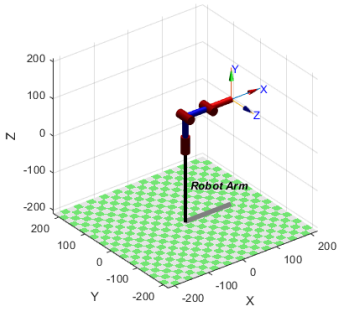

Fig. 3. Placeholder image for the robotic arm?.

MANIPULATOR KINEMATICS

Kinematics is what describes the motion of rigid bodies and points in space.

Backgroud

Rigid body transformation

In general any position can be described by translation along three axes and rotation along these same axes. These translations/rotations can be described by a matrix

$$T = \begin{bmatrix} R & d \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

where R is a 3 x 3 rotation matrix and d a 3 x 1 translation matrix. This implies that any transformation could be characterized by six parameters, three for the translation and three for the rotation. [4]

Denavit-Hartenberg Convention

A common approach for selecting the coordinate frames of reference for each joint in a robotic arm is the Denavit-Hartenberg convention. This allows the transformation matrix for each joint to be expressed as

$$A_i = Rot_z(\theta_i) \cdot Trans_z(d_i) \cdot Trans_x(a_i) \cdot Rot_z(\alpha_i) \tag{2}$$

that consists of the four basic transformations

$$Rot_z(\theta_i) = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$Trans_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$Trans_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$Rot_z(\theta_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

Where the parameters θ_i, a_i, d_i and α_i, known as DH-parameters, characterize each joint. To allow the transformation for each joint to be represented by only four parameters, compared to the six parameters that is required in general, there are some restrictions on how the coordinate frames can be chosen. To comply with the DH-convention, the following must be satisfied.

- The axis x_1 is perpendicular to the axis z_0
- The axis x_1 intersects the axis z_0

where it is assumed that two frames are given, frame 0 and frame 1, and the transformation from equation 2 transforms a coordinate from frame 1 into a coordinate in frame 0. [4]

Kinematic Chain

A robotic manipulator can be described by a set of joints with links between them where a homogeneous transformation matrix A_i that describes the transformation with respect to the previous joint exists for each joint. This means that a transformation that describes the position and orientation of joint j with respect to a joint i to a can be found by a transformation matrix [4]

$$\begin{cases} T_j^i = A_{i+1}A_{i+2}...A_{j-1}A_j, \text{ if } i < j \\ T_j^i = I, \text{ if } i = j \\ T_j^i = (T_i^j)^{-1}, \text{ if } i > j \end{cases} \quad (7)$$

Forward Kinematics

Forward kinematics is the problem of finding the position and orientation of the end effector.

Since a manipulator can be seen as a kinematic chain the problem of finding the position and orientation of the end effector can be solved by finding the transformation matrices in equation 7.

Inverse Kinematics

The problem of finding the joint states required for achieving a desired pose is called inverse kinematics. For some simple kinematic chains an analytical solution exists, but in general a numerical approach might be required.

Workspace

A robotic manipulator will not be able to reach all points in space since it has a fixed size. It will not even be able to reach all mathematically reachable points since each joint (usually) have some restrictions on how it can rotate/translate. The physically reachable space is defined as the workspace of the manipulator.

Implementation

Denavit-Hartenberg Convention

In table II the DH-parameters that was used for the robotic manipulator, seen in figure 3, are listed.

TABLE II
THE DENAVIT-HARTENBERG PARAMETERS USED FOR THE ROBOTIC MANIPULATOR.

i	θ_i [rad]	d_i [mm]	a_i [mm]	α_i [rad]
1	θ_1	75	0	$\pi/2$
2	θ_2	0	67.5	0
3	θ_3	0	67.5	0
4	θ_4	0	65	0

Forward Kinematics

The transformation matrices from equation 2 for each joint can be multiplied together

$$T_{end} = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \quad (8)$$

where A_1 - A_4 are calculated from equation 2 to find the pose T_{end} of the end effector and thereby solving the forward kinematic problem.

Inverse Kinematics

To solve the inverse kinematics for the position of a three joint manipulator a geometric method can be used to find an analytical solution. The required angles to achieve a desired position (x_d, y_d, z_d) in an elbow-up configuration are [5]

$$\begin{cases} \theta_1 = Atan(x, y) \\ \theta_3 = Atan(D, +\sqrt{1 - D^2}) \\ \theta_2 = Atan(\sqrt{x_d^2 + y_d^2}, z_d - d_1) \\ -Atan(a_2 + a_3 \cdot D, a_3 \cdot \sqrt{1 - D^2}) \end{cases} \quad (9)$$

where $Atan(x, y)$ is the two argument arctangent function [6] and

$$D = \frac{x_d^2 + y_d^2 + (z_d - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3} \quad (10)$$

and a_2, a_3, and d_1 are the corresponding DH-parameters.

Workspace

The robotic manipulator seen in figure 3 has the following limitations for the first three joints

$$\begin{cases} -100^\circ < \theta_1 < 100^\circ \\ -100^\circ < \theta_2 < 100^\circ \\ -100^\circ < \theta_3 < 100^\circ \end{cases} \quad (11)$$

and a simulated 2D slice of the workspace can be seen in figure 4. The full workspace can be formed by rotating this 2D slice around the z-axis according to the limitations on θ_1 from equation 11.

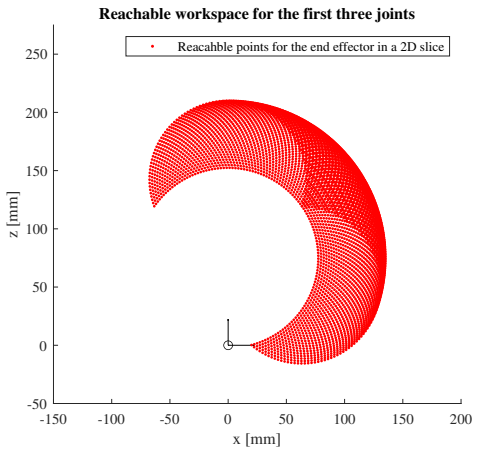

Fig. 4. A simulated 2D slice of the workspace for the three jointed manipulator.

BASE

The motors will be connected to tracks on the robot and can therefore be modelled as two wheels connected with a rod, as seen in figure 5.
Deriving a mathematical model is straight forward if we can read the encoders from the motors. From the encoders we would be able to get both the length the robot has traveled and more importantly the individual speed each motor rotates with. The speed for the individual motor is given by:

$$v_m = \frac{2\pi r_{wh}/N_{enc}}{\Delta t} \tag{12}$$

Where N_{enc} is the number of encoders on the motor, r_{wh} is the radius of the driving wheel and thickness of the track and Δt is the time between the previous encoder reading and the most recent one. By doing this for both the left and right motor we can find how fast the robot is moving along the line by:

$$\overline{v} = \frac{v_L + v_R}{2}(-\cos\theta\hat{i} + \sin\theta\hat{j}) \tag{13}$$

Where the y-axis parallel to the line. In a similar fashion the angular velocity can be calculated using:

$$\dot{\theta} = \frac{v_R - v_L}{2r_b} \tag{14}$$

Where r_b is the distance from the center of the base to the wheels. From this and figure 5 it's easy to see that the change of angle then becomes:

$$\theta = sin^{-1}\left(\frac{V_L - V_R}{2r_b}t\right) \tag{15}$$

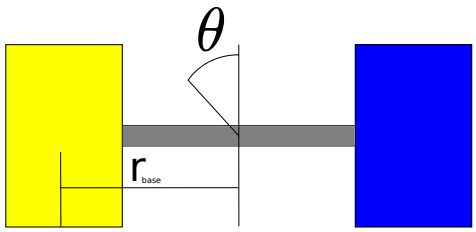

Fig. 5. Modelling of the base as two wheels connected with a rod, where θ is the angle to the line

Given the use of small-angles approximation[1], the angular velocity can be written as:

$$\theta = \frac{V_L - V_R}{2r_b}t \tag{16}$$

Using equations (12)-(14) a system can be built for simulations in matlab, using rate of change and max/min value limiters for simulations of the motors physical restrictions.

CAMERA VISION AND CALIBRATION

In this section it is outlined the theory behind the camera vision system. How points in a 3D-space are projected on a 2D-plane, calibration and distortion correction.

Camera model

Distortion

V. MACHINE VISION

LINE FOLLOWING

QR-CODES READING

DEEP LEARNING?

VI. MOTION

ROBOT ARM PATH PLANNING

If a robot manipulator is required to visit specific discrete points, waypoints, path planning is the problem of deciding how to move between these points. Planning the path is also important to ensure the movements are smooth, meaning that, since we "control" the speed of each joint, the desired positions and velocities should be continuous.

Interpolation

One method of interpolating between points is to use cubic spline interpolation. A spline [7] is a piecewise polynomial function and consequently a cubic spline is a function consisting of piecewise cubic polynomials. A cubic polynomial will ensure a smooth movement since its first derivative, the velocity, will be continuous. To ensure that each required point is visited the bounding values for each cubic function are set to zero [8].

[1]Which will be assumed since the robot will be operating on a grid with hard coded functions for turning at QR-codes

Joint Space interpolation or Cartesian Motion

The interpolation can take place in joint space or in cartesian space. Joint space interpolation means that the waypoints are expressed as joint values, angle of rotation if the joint is a revolute joint and distance if the joint is a prismatic joint. The interpolation is then calculated between these joint values. Cartesian motion means that the waypoint are expressed as cartesian coordinates and that the interpolation takes place in cartesian space.

Implementation

The robotic manipulator, shown in figure 3, has a number of waypoints that it is required to visit. The desired waypoints are expressed in joint values, and a cubic spline interpolation is used to interpolate between these joint space waypoints. The path for picking up and placing the object can be seen in figure 6 and 7 respectively.

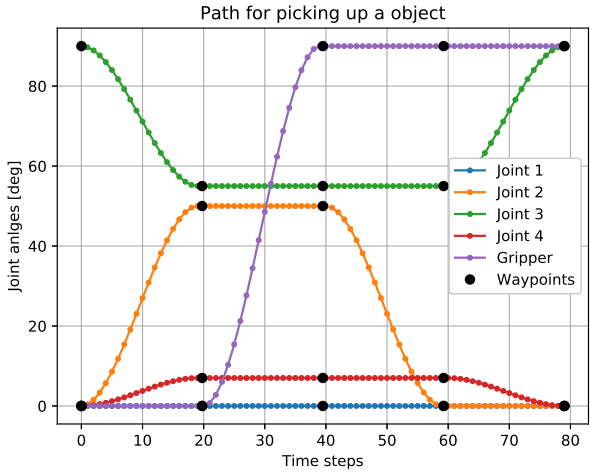

Fig. 6. Path showing the joint values required for picking up the object.

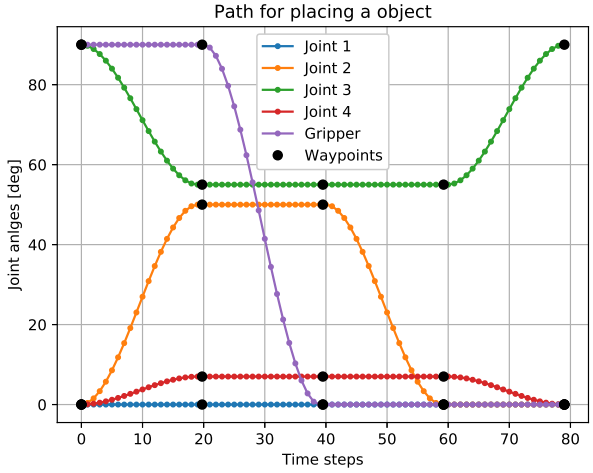

Fig. 7. Path showing the joint values required for placing the object.

BASE

VII. EXPERIMENT

PROTOTYPE COMPONENTS

Overview of what is being used in the evaluation

MOVABLE BASE EVAUATION

ARM EVALUATION

GRIPPING EVALUATION

SENSOR EVALUATION

ARROWHEAD EVALUATION

VIII. CONCLUSIONS

Conclusions of the robot and the course (?) Is our design good? Is Arrowhead good?

CONCEPTUAL DESIGN

APPENDIX

MARTIN BLASZCZYK

Martin is a 5th year Y-student with interest in Control och Mechatronics. In this course he'll take the role of the Project Leader where the main objectives are to keep focus on the goal, hold meetings and an overall oversight of the project. As for the technical part the main interest will be in machine vision together with Edward K. to use cameras or other sensors to localize external objects for the robot to grip, avoid or approach.

EDWARD CEDGÅRD

Master in electronic systems and control engineering. Edward's main task is to design the robotic arm and gripper mechanism together with Niklas. Tasks as deriving the kinematic equations, and implementation using forward and inverse kinematics. Choise of motors, armdesign, communication with motors using serial communication.

NIKLAS DAHLQUIST

Niklas is studying his fifth year at the Engineering physics and electrical engineering student program.

His main focus will be to work with Edward Cedgård to evaluate the gripping mechanism and if necessary design new components and model the corresponding control system to be able to lift up and hold the target object.

EDWARD KÄLLSTEDT

Currently studying his fourth year in the Master Programme in Computer Science and Engineering. A fan of making things secure, fast, scalable, and well-documented. Primarily interested in low level software development. Will initially work on the machine vision implementation together with Martin. In addition to machine vision specifically this work will also consist of robot localization and collision avoidance. As the project progresses he will take on more general software problems that might arise. The first week will be spent researching different computer vision technologies.

ALBIN MARTINSSON

Albin is a 5th year computer science student specializing in industrial computersystem. In this project he will be focusing on the arrowhead integration and bein charge of the Github repository. This will entail connecting all the services toeach other and making sure they are authenticated and secure. Being in chargeof the git repository will entail merging pull requests and sorting out conflicts, making sure that the version control part of this project runs smoothly.

MÅNS NORELL

Studying for a master in electronic systems and control engineering. Måns main task is to design the base and line-following controller for moving the robot along a line. Tasks include designing the base, printing the specialized parts, simulating and testing the base. Communication between controller, motor and camera will be worked on in collaboration with those in charge of these tasks.

PROJECT STRUCTURE

To keep the project going and have an organized structure the project is divided in different parts, or subprojects. Each group member is either alone or in group responsible for each part of the project which coincides with their interests.

- Arrowhead
- Machine vision and localization of external objects
- Gripping tool
- Movable base

MEETINGS

Every week the group met on Mondays and Thursdays to catch up and support each other. This structure gave the students a great deal of responsibility to do work for each meeting while still maintaining a good structure of the project and encouraged discussions. The agenda of the monday meeting was:

- Status of work done the previous week by each member
- Preparation for the seminar
  - Discussion of the previous seminar meeting
  - How the weeks work has been coinciding with the seminar feedback
  - Questions to ask the teachers
  - Questions to ask the other group
  - Who does what during the seminar
- Other

The Thursday meeting was to collect and reflect over the feedback from the teachers and the other group from the seminar. Also a status on the work planned to be done the following week was discussed so that each member had a good understanding of what the other members were doing. Those meetings had had the following agenda

- What feedback did the teachers give
- What feedback did he other group give
- Feedback to each other withing the group
- Work to be done the following week
- Other

A. Project planning

The projects course started in August 2020 and continued until mid January 2020 with the project deadline in the middle of December. To keep a good structure and to synchronize all the parts of the project a plan was made, shown in Table III. This plan enabled the group to be flexible and maintain a good long-term structure of the project. Every month a more detailed time plan was prepared to facilitate the small changes, delays etc.

For the group members to what task were to be done, the built in function of Issues on the souce control platform Github® was utilized. A Milestone was created for each week and populated with issues. When an issue was finished it could simply be closed. If the issue was delayed it showed clearly in the Issue overview which tasks had to be prioritized.

TABLE III
OVERALL TIME PLAN

Sep	Oct	Nov	Dec
Concept generation	Evaluation	Evaluation	
Theory	Prototyping	Evaluation	Finishing up
Simulation	Evaluation	Evaluation	
Prototyping	Final Design	Evaluation	

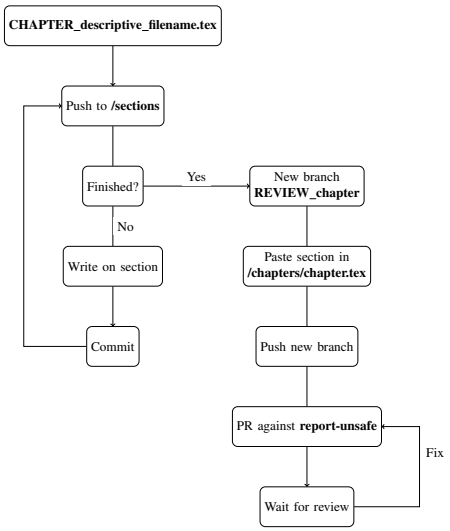

Fig. 8. Report writing flowchart

B. Source control

To keep track of the different software implementation the projects source control implements Git in one common repository [9]. The repository is where all source code and relevand 3D-files are located. This report is written in LaTeX with Git as source control. To make sure it's easy for all members to write their designated sections a workflow was designed to to minimiza merge conflicts while writing drafts as show in Fig. 8. Without this flow the group members would experience merge conflicts after every push which would make it more complex and time consuming.

REFERENCES

[1] ROBOTIS TEAM. e-manual from robotis. [Online]. Available: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

[2] ——. Protocol from robotis. [Online]. Available: https://emanual.robotis.com/docs/en/dxl/protocol1/

[3] ——. Dynamixel sdk lib. [Online]. Available: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/

[4] M. V. "Mark W. Spong", "Seth Hutchinson", *Robot Dynamics and Control*, 2nd ed. John Wiley & Sons, Inc., 2004.

[5] D. Rus. Robotics systems and science lecture 14: Forward and inverse kinematics. [Online]. Available: https://courses.csail.mit.edu/6.141/spring2011/pub/lectures/Lec14-Manipulation-II.pdf

[6] E. W. Weisstein. "inverse tangent." from mathworld–a wolfram web resource. [Online]. Available: https://mathworld.wolfram.com/InverseTangent.html

[7] ——. "spline." from mathworld–a wolfram web resource. [Online]. Available: https://mathworld.wolfram.com/Spline.html

[8] ——. "cubic spline." from mathworld–a wolfram web resource. [Online]. Available: https://mathworld.wolfram.com/CubicSpline.html

[9] Students, "D7039e," https://github.com/kottz/D7039E, 2020.