System Verification and Validation Plan for CVT Simulator

Team #17, Baja Dynamics
Grace McKenna
Travis Wing
Cameron Dunn
Kai Arseneau

March 20, 2025

Revision History

Date	Version	Notes
October 11th, 2024	0	First version of VnV plan
March 20th, 2024	1.0	Added traceability to tests

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations, and Acronyms

acronym	definition
CD	Continuous Development
CI	Continuous Integration
CVT	Continuous Variable Transmission
GPS	Global Positioning System
IMU	Inertial Measurement Unit
GUI	Graphical User Interface
IM	Instance Model
MG	Module Guide
MIS	Module Interface Specification
MSE	Mean Squared Error
NFR	Nonfunctional Requirement
PR	Pull Request
R	Functional Requirement
RPM	Revolutions Per Minute
SRS	Software Requirements Specification
VnV	Verification and Validation

Table 1: Verification and Validation Acronyms

This document serves as a guide for how we plan to verify and validate the CVT Simulator system. Within is a detailed plan of all the tests that will be conducted to ensure that the system is functioning as intended. The document is divided into several sections going from the general plan to the specific tests that will be conducted. On the verification side, the document lists all the tests that will be conducted to verify the performance of the system. The tests are divided into functional and non-functional requirements and then separate system and unit test for each. There are also multiple checklists in place to verify that all the requirements both within and outside the document are met. On the validation side, the document outlines all the plans in place to validate the results of the system. This includes how we plan to compare our results against real world data and how we plan to validate the usability of the system. Overall, this document comprehensively outlines all the plans in place to verify the system and validate its results.

2 General Information

This section serves as an overall summary of the verification and validation plan. It includes a summary of the project and the overall objectives of the VnV plan. The relevant documentation and other specifications are also listed. In summary, this section provides a high level overview of the VnV plan.

2.1 Summary

The software that will be tested is the CVT Simulator, this is a system that simulates the Continuously Variable Transmission (CVT) system of a Baja vehicle. The system has three components, a user interface which will allow the user to input the parameters of the CVT system and visualize the output of the simulation. A mathematical component that will take the input parameters and simulate the CVT system. There is also a 3D model component that will display the some models of the CVT system. Each of these components covered will be verified and validated to ensure that the system is functioning as intended.

2.2 Objectives

The objectives of testing our CVT system are to ensure the accuracy and reliability of our outputs. This includes evaluating that our mathematical model provides correct outputs based on provided input parameters specifically looking at the calculations of RPM, velocity, acceleration, distance, torque of the engine, clamping

forces, CVT ratio, Torque and Belt slippage. We aim to emphasize the description over specification to make the CVT's system's behavior understandable by demonstrating how input changes influence the outputs. We will verify that our 3D model and user interface allow users to easily input parameters to the system and interpret meaningful results. Our validation tests will focus on graphical models based on current available data to describe how the outputs should behave. We will aim to quantify these results by analyzing the difference between expected versus actual output. By referencing our outputs to real world data we can validate the accuracy of our mathematical model. We will conduct usability testing to ensure that the interface is intuitive and allows users to easily input parameters, visualize results and understand the performance of the CVT.
For our project attempting to optimize our systems outputs will be considered out of scope. Additionally, we will not verify any third party libraries used in our software as we assume that this has been previously validated.

2.3 Challenge Level and Extras

The challenge level of this project is general.

The extras for this project are the following:
Validation Report

- The system will undergo validation testing to verify that accuracy of the simulation.
- The validation report will be included in the final documentation.

Usability Testing

- The system will undergo usability testing by members of the McMaster Baja team to ensure that the user interface is intuitive and easy to use.
- We will compile the results of the usability testing into a report that will be included in the final documentation.

2.4 Relevant Documentation

Software Requirements Specification Dunn et al. (2024)
This document is relevant because it it list all the requirements that the system

should have and specifies how they should be achieved. Requirements serve an important role in the verification process as the metric for determining if the system is functioning as intended. When we do the verification of the system we will compare the system's capabilities and outputs to the requirements from the SRS document. This will tell us what parts of the system are working as intended and which need to be changed or updated. For validation, the specification serves and equivalent role as the metric for determining if the system is implemented correctly. The specifications will be compared with the implementations of the system to see if they correspond. If they do not, then the system will need to be updated so the the implementations match with the specifications. Overall, the SRS document is important for verification and validation as it provides the metrics for determining their success.

Module Guide
The Module Guide will serve as a reference for the components of the system. Each module listed in the MG will have a corresponding test case in the VnV plan.

Module Interface Specification
The Module Interface Specification serves as a reference for the functions and methods of the system. Each function and method listed in the MIS will have a corresponding test case in the VnV plan.

Verification and Validation Report
The Verification and Validation Report will be referenced to ensure that all the components of the system have been verified and validated.

3 Plan

This section outlines the overall plans for the verification and validation of the system. This includes our approaches, techniques and tools that will be used. Our approaches our outlined and each person's role is specified. There is also a plan listed for both the verification and validation along with an implementation specification. Any tools that we plan to use to automate the process are also listed in this section. Overall this section details how we plan to verify and validate the system.

3.1 Verification and Validation Team

Name	Role	Responsibilities
Kai Arseneau	Data Verification Lead	Analyzes data outputs to ensure trends align with expected performance and validates consistency across simulation runs.
Travis Wing	GUI Verification Lead	Ensures user interface functionality and usability, verifying that GUI elements accurately reflect simulation data and respond as intended.
Cameron Dunn	Experimental Data Comparison Lead	Compares simulation outputs to real-world data and scripts, verifying the simulation's accuracy and adjusting model parameters as needed.
Grace McKenna	Quality Assurance Lead	Conducts comprehensive testing of simulation processes and components, identifying and documenting issues, and ensuring overall software reliability.
Ariel Wolle	Team Lead	Supports the project by overseeing resources and approving initiatives that facilitate enhanced data acquisition or other verification efforts.
Daksh Mathur	Data Acquisition Lead	Acts as the primary contact for real-world data acquisition, providing essential data for validation and supporting data needs beyond the project group.
Benjamin Waldie	Mechanical Advisor	Collaborates with the team to validate mathematical models within the simulation, ensuring mechanical accuracy and guiding technical correctness.
Duncan Shearer and Leonardo Ansari	Primary User	Utilizes the simulation tool to tune CVTs, providing feedback on tool usability and functionality to ensure it meets real-world tuning needs.
Dr. Spencer Smith	Project Advisor / Supervisor	Reviews and provides final feedback, ensuring verification meets project standards and advising on methodology improvements.

Table 2: Verification and Validation Team for CVT Simulation Project

3.2 Data Collection

To ensure accurate validation of our CVT simulation outputs, the McMaster Baja Racing team employs a robust data collection system. This system, overseen by the Data Acquisition Lead, records GPS data (including position and velocity), Inertial Measurement Unit (IMU) data, engine speed, and wheel speed during structured testing sessions. These testing days are designed to maintain predictable and repeatable conditions, providing high-quality, reliable data.

The collected data will be distributed among the CVT team members, ensuring alignment between simulation predictions and physical testing outcomes. References to specific data metrics appear throughout this document, providing context for how each aspect of collected data supports the validation objectives outlined.

3.3 SRS Verification Plan

To ensure that the SRS document is accurate and complete, we will be conducting a series of reviews. These reviews will be conducted by the team, our supervisor, and our stakeholders. The team will use the below checklist to verify the quality of the SRS document:

- Review the SRS document for spelling, grammatical and formatting errors
- Check the applicability of each theoretical models
- Check the usefulness of each data definitions
- Check the validity of each instance models

Review Meetings

We will conduct meetings with experienced team members and alumni to review the SRS document. These meetings will focus on the theoretical models, data definitions, and instance models. We will also conduct a practicality check to ensure that all requirements are feasible and measurable. Finally, the assumptions will also be verified to ensure that they won't negatively impact the accuracy of the simulation.

Baja Team Review

Talk here about sharing the document to the team as a whole, include a checklist and receive issues on github.

We will share the SRS document with the entire McMaster Baja Racing team for review. This will allow us to get feedback from a wide range of people with different backgrounds and experiences. The team will use the checklist provided above as a guide for their reviews. Afterwards we will compile all of their feedback and address any issues that they have raised.

Capstone Team Testing

We will conduct unit tests on the mathematical models provided by the SRS document. These will be done on their implementations to ensure that they are functioning as intended. If any issues are found, we will reference the assumptions and requirements in the SRS document to determine the cause of the issue. Additionally, the traceability matrix will be used to ensure that the mapping between requirements and goals is logical.

3.4 Design Verification Plan

To ensure that the CVT system is functional and accurate, a design verification plan will be conducted on the system. The components of the system that will be tested are the graphical user interface, data output and visualization and the mathematical model. The design verification plan will include, reviews from fellow classmates, stakeholder reviews and specification testing based on the functional and nonfunctional requirements of the system. The system will receive Stakeholder review providing us with relevant feedback on the functionality and interface of our design. The system will undergo reviews from fellow classmates, providing our team with quality feedback from an outside perspective. A checklist has been designed below to which fellow classmates and stakeholders can use to verify the system.

- For each functional requirement there is a at least one corresponding system test.
- For each non functional requirement there is a at least one corresponding system test.
- The system tests contain tests for boundary cases for user inputs, empty inputs and edge cases.
- Each instance model (IM) within the SRS document shall correspond to a unit test.

- Each function written in our mathematical python model shall correspond to at least one unit test.
- There exists system tests for each component: user interface, mathematical model, data visualization and 3D model.

3.5 Verification and Validation Plan

To ensure that the Verification and Validation Plan is sufficient, this document will undergo a verification plan. This plan will review the contents of the document to make sure that all necessary components are included. It will be reviewed by our team, supervisor, stakeholders and classmates. Below is a checklist to use as reference for the verification of the Verification and Validation Plan.

Overall

- The document is well organized and easy to follow
- The document is free of spelling, grammatical and formatting errors
- The document is consistent with all other documentation and implementation
- The document is complete and has all components filled out
- References are included and correct

Section 1

- Tables included are relevant and necessary
- Tables include are correct and formatted properly
- Introduction sufficiently introduces the document

Section 2

- Roadmap effective summarizes the section
- Summary effectively summarizes the document
- Objectives are clear and concise
- Challenge Level correct and Extras are appropriate

- Relevant Documentation is matching and descriptive

Section 3

- Roadmap effective summarizes the section
- Team roles allocated properly and effectively
- SRS Verification Plan includes all necessary components in the checklist
- Design Verification Plan includes all necessary components in the checklist
- VnV Plan Verification Plan includes all necessary components in the checklist
- Implementation Verification Plan includes all techniques that are used
- Automated Testing, and Verification Tools includes all tools that are used
- Software Validation Plan includes all external data that is used

Section 4

- Roadmap effective summarizes the section
- All functional requirements have corresponding system tests
- All nonfunctional requirements have corresponding system tests
- Traceability table between test cases and requirements is clear

Section 5

- Roadmap effective summarizes the section
- Unit Testing Scope is correctly defined
- All functional requirements have corresponding unit tests
- All nonfunctional requirements have corresponding unit tests
- Unit tests are properly grouped into their respective modules
- Traceability table between test cases and modules is clear and includes all modules

Section 6

- Roadmap effective summarizes the section and includes all additions
- Additional information is relevant and necessary
- Included sections are informative and properly filled out

3.6 Implementation Verification Plan

For the implementation verification plan, we will be using many techniques to ensure that the system is functioning as intended. We will use the unit tests listed in the this document as well as other common testing techniques. It will be required that all unit tests are run and pass for any PRs to be merged into their parent branch. We will be doing code inspections during development by ensuring that all code is reviewed by at least one other team member before merging. Static analyzers will also be used throughout development to ensure a high quality codebase and will be integrated into our CI/CD pipeline. Before major milestones, full code walkthroughs will be conducted which is enforced by all team members needing to approve merges into main.

3.7 Automated Testing and Verification Tools

We will be using a wide variety of tools for automated testing and verification. We will use separate tools for our Unity C# frontend and python backend.

For the python backend we will be using the following tools:

flake8 - A Python linter that checks for PEP8 compliance.

black - A Python code formatter that will ensure consistent code style.

unittest - Python's built-in testing framework that will be used for unit testing.

coverage - A testing framework for Python that will be used for code coverage.

For the Unity C# frontend we will be using the following tools:

SonarLint - C# linter that checks for code quality and security vulnerabilities.

StyleCop - C# linter that checks for code style and formatting.

UTF - A testing framework for C# that will be used for unit testing.

UTR - A testing framework for Unity that will be used for unit testing and code coverage.

Additionally, we will be using GitHub Actions to automate our testing process. This will be done to ensure CI/CD and that our tests are run automatically on every push to the repository.

3.8 Software Validation Plan

We plan on using data collected from the McMaster Baja team to validate the accuracy of our simulation. Some of the values we are able to collect include:

- Primary RPM
- Secondary RPM
- CVT Ratio (from Primary and Secondary RPM)
- Wheel Speed
- GPS Speed
- GPS Position
- Acceleration

To validate this data against our simulation we plan on creating a script to compare them. This script will compute the MSE between the two datasets and also plot that error over time. Plotting the error will allow us to see in what scenarios the simulation is least accurate. This will allow us to better diagnose the issues and allocate our resources to fix them.

4 System Tests

Found below are all of the system tests that will be conducted to verify the system. These tests are split based on if they are for functional or nonfunctional requirements.

4.1 Tests for Functional Requirements

This section outlines all the system tests that will be conducted to verify the requirements of the system. Each set of tests relate to one requirement from the SRS document.

Standard Simulation Inputs

The standard simulation inputs for the CVT system are as follows.

1. Primary Pulley System:
    - Flyweight
    - Ramp Geometry
    - Spring rate
    - Spring pretension
2. Secondary Pulley System:
    - Helix Geometry
    - Torsional spring rate
    - Compressional spring rate
    - Spring pretension
3. Vehicle Characteristics:
    - Car weight
    - Driver weight
    - Traction
    - Angle of incline

Standard Simulation State

The standard simulation state for the CVT system is as follows.

- Vehicle is stationary.
- The engine's angular velocity is at ω_{idle}.
- The CVT system is in a neutral state, unshifted.
- The belt is stationary.

4.1.1 Vehicle Dynamics

This section provides tests to verify the vehicle dynamics of the CVT system. They are based on the following functional requirements from the SRS document. Listed is each requirement and how the tests verify each one.

R1: Verifies the mathematical modeling of the CVT system

R2: Verifies the acceleration of the vehicle as a function of time

R3: Verifies the velocity of the vehicle as a function of time

R4: Verifies the position of the vehicle as a function of time

Position

1. **Position test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graph should show a position that is a monotonically increasing function over time, starting from 0. There is no upper bound. A typical output graph will look similar to the one shown in Figure 1.

   Test Case Derivation: As long as the vehicle is able to overcome the initial force of the slope, the vehicle should continue to move forward indefinitely.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output position over time will be visualized as a graph which should show a smooth upward trend as time progresses. Its slope should increase over time as well. Visual inspection will follow against Figure 1 for confirmation of expected behavior.

2. **Position test-2**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: Provided experimental data along with the input values from Section 4.1, matching those of the vehicle during experimental data collection. Sample experimental data is located here: `Latitude.csv`, `Longitude.csv`.

Output: Mean squared error between the simulated position and the experimental position.

Test Case Derivation: Data from the simulation should closely match the experimental data collected from the physical car.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output of the simulation will be compared to experimental data from the physical car to ensure that the simulation is accurate. The experimental data will be configured to show the absolute distance travel since the start of the experiment.

Velocity

1. **Velocity test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graph should show a velocity that is a monotonically increasing function over time, starting from zero and remaining below $v_{\max}$, which reflects the vehicle's physical limits. A typical output graph will look similar to the one shown in Figure 2.

   Test Case Derivation: Based on assumption 11 that the vehicle should not move backward under normal conditions. Derived from the vehicle's geometry and the CVT configuration, limiting the velocity to a defined top speed based on the system's constraints.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output velocity over time will be visualized as a graph which should show a smooth upward trend that begins at zero and approaches $v_{\max}$ as time progresses. Visual inspection will follow against Figure 2 for confirmation of expected behavior.

2. **Velocity test-2**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: Provided experimental data along with the input values from Section 4.1, matching those of the vehicle during experimental data collection. Sample experimental data is located here: `Speed.csv`.

Output: Mean squared error between the simulated velocity and the experimental velocity.

Test Case Derivation: Data from the simulation should closely match the experimental data collected from the physical car.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output of the simulation will be compared to experimental data from the physical car to ensure that the simulation is accurate.

Acceleration

1. **Acceleration test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graph should show an acceleration that is a smooth monotonically decreasing function over time, starting from $a_{\max}$ and trending to 0, which reflects the engine's limits along with the maximum and minimum transmission ratio. A typical output graph will look similar to the one shown in Figure 3.

   Test Case Derivation: Based on the engine's output limits, going no lower than ω_{idle} and no higher than $\omega_{\max}$. Based on air resistance, which increases with the square of velocity, alongside the diminishing torque output of the engine.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output will be visualized as a graph. Visual inspection will follow against Figure 3 for confirmation of expected behavior.

2. **Acceleration test-2**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: Provided experimental data along with the input values from Section 4.1, matching those of the vehicle during experimental data collection. Sample experimental data is located here: `Accel X.csv`, `Accel Y.csv`, `Accel Z.csv`.

Output: Mean squared error between the simulated acceleration and the experimental acceleration.

Test Case Derivation: Data from the simulation should closely match the experimental data collected from the physical car.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output of the simulation will be compared to experimental data from the physical car to ensure that the simulation is accurate. The experimental data will be configured to display the acceleration in the direction of travel of the vehicle.

4.1.2 CVT Dynamics

This section provides tests to verify the CVT dynamics of the system. They are based on the following functional requirements from the SRS document. Listed is each requirement and how the tests verify each one.

R1: Verifies the mathematical modeling of the CVT system

R5: Verifies the calculations of the primary clamping force

R6: Verifies the calculations of the secondary clamping force

R7: Verifies the calculations of the sheave acceleration

R8: Verifies the calculations of the belt acceleration

R9: Verifies the calculations of RPM and engine torque

Clamping forces

1. **Clamping forces test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graphs should show a clamping force against engine angular velocity beginning an initial minimum value, which should then rise to a peak before falling during the shifting phase. Suboptimal parameters may lead to a clamping force that is too low or too high, but a similar shape

nonetheless. A typical output graph will look similar to the one shown in Figure 4.

Test Case Derivation: This behavior is resulting from ideal tuning parameters, and may vary depending on inputs. Ramp and helix geometry will both have a major impact on the shape of these graphs, while weight and spring constants will affect the magnitude.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output will be visualized as a graph. Visual inspection will follow against Figure 4 for confirmation of expected behavior.

Shift

1. **Shift test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graphs should show the shift distance against the engine's angular velocity. An ideal graph will look very similar to the leading edge of a square wave. A typical output graph will look similar to the one shown in Figure 5.

   Test Case Derivation: This behavior is resulting from the CVT's clamping forces overcoming the resistive forces just enough such that the CVT begins shifting. Once this has begun, ramp geometry is expected to cause shifting rather than the engine increasing in speed, maintaining a constant power output. With suboptimal tuning parameters passed in, one might observe a slanted shifting portion.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output will be visualized as a graph. Visual inspection will follow against Figure 5 for confirmation of expected behavior.

2. **Shift test-2**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: Provided experimental data along with the input values from Section 4.1, matching those of the vehicle during experimental data collection.

Sample experimental data is located here: Primary RPM.csv, Secondary RPM.csv

Output: Mean squared error between the simulated shift against engine velocity compared to experimental data. Differences of shape and magnitude will be considered.

Test Case Derivation: Data from the simulation should closely match the experimental data collected from the physical car. Key points of interest include the shift point and the slope during the shifting phase.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output of the simulation will be compared to experimental data from the physical car to ensure that the simulation is accurate. The experimental data will be configured to show the primary RPM divided by the secondary RPM on the y-axis, and the primary RPM on the x-axis.

3. **Shift test-3**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graph show the engine's angular velocity against the vehicles' velocity. The output should show a smooth trend upwards starting at ω_{idle} until reaching the shift point, in which it then moves horizontally until finally reaching full shift and bringing the engine's angular velocity to ω_{max}. A typical output graph will look similar to the one shown in Figure 6.

   Test Case Derivation: This behavior results from the CVT's shifting behavior as a whole. An in-depth explanation of the phases can be found beneath Figure 6.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output will be visualized as a graph. Visual inspection will follow against Figure 6 for confirmation of expected behavior.

4. **Shift test-4**

   Control: Manual

   Initial State: The state values from Section 4.1.

Input: Provided experimental data along with the input values from Section 4.1, matching those of the vehicle during experimental data collection. Sample experimental data is located here: `Primary RPM.csv`, `Secondary RPM.csv`

Output: Mean squared error between the simulated shift curve (see Figure 6) and the experimental data. Differences of key points, shape and magnitude will be considered.

Test Case Derivation: Data from the simulation should closely match the experimental data collected from the physical car. Key points include the shift point, slope during shifting phase, engagement point and the final shift point.

How test will be performed: The script will run with the provided inputs. After execution is complete, the output of the simulation will be compared to experimental data from the physical car to ensure that the simulation is accurate. The experimental data will be configured to show the primary RPM on the y-axis and the secondary RPM on the x-axis.

Engine Dynamics

1. **Engine test-1**

   Control: Manual

   Initial State: The state values from Section 4.1.

   Input: The input values from Section 4.1.

   Output: The resulting graph should show a decreasing relationship with torque and engine angular velocity, while having a peak in power at $\omega_{\text{peak\_power}}$. A typical output graph will look similar to the one shown in Figure 7.

   Test Case Derivation: These are given values as specs of the engine at full load. The engine should not significantly leave the bounds of these angular velocities.

   How test will be performed: The script will run with the provided inputs. After execution is complete, the output will be visualized as a graph. Visual inspection will follow against Figure 7 for confirmation of expected behavior.

4.1.3 User Interface

This section provides tests to verify the User Interface of the system. They are based on the following functional requirements from the SRS document. Listed is each requirement and how the tests verify each one.

R10: Verifies the input parameters of the CVT can be adjusted

R11: Verifies the other input parameters can be adjusted

R12: Verifies the system provides a UI to change the input parameters

R13: Verifies the system provides output in the form of graphs

R14: Verifies the system lets users compare simulation data

R15: Verifies the system lets users export simulation data

1. **User Interface test-1**

   Control: Manual

   Initial State: Default input parameters are loaded

   Input: Change the input parameters with new values

   Output: Input parameters have been updated

   Test Case Derivation: Based on R10 and R11, the user should be able to adjust each of the input parameters (within the limitations of the input itself). This includes CVT parameters such as weight, ramp geometry, spring rate and spring pretension as well as other parameters such as vehicle weight, driver weight, traction and angle of incline.

   How test will be performed: Upon a fresh launch of the application, manually change each of the input parameters from the default values to a new value.

2. **User Interface test-2**

   Control: Automatic

   Initial State: Application has not been launched

   Input: Launch application

   Output: User interface for changing the input parameters is displayed

   Test Case Derivation: Based on R12 validate that the system provides the user with a UI to change the input parameters.

   How test will be performed: Upon a fresh launch of the software, the system will assert whether the input parameter page is displayed or not.

3. **User Interface test-3**

   Control: Automatic

   Initial State: Simulation preloaded with input parameters

   Input: Run simulation

   Output: Graphical representation of simulation data is displayed to the user

   Test Case Derivation: Based on R13, validate that the simulation produces output in the form of graphs.

   How test will be performed: The test will run the simulation with a set of pre-made input parameters, then it will verify that whether a graph has been outputted.

4. **User Interface test-4**

   Control: Manual

   Initial State: Simulation has completed

   Input: Select export data option

   Output: simulation data exported successfully

   Test Case Derivation: Based on the R15 validate user can export simulation data after simulation has completed.

   How test will be performed: After simulation has completed successfully export the data and verify that it was exported in the correct format to the right location.

4.1.4 Compatibility

This section provides tests to verify the Compatibility of the system. They are based on the following functional requirements from the SRS document. Listed is each requirement and how the tests verify each one.

R16: Verifies the application works on different types of computers

R17: Verifies the application works on different operating systems

1. **Compatibility test-1**

   Control: Manual

   Initial State: Application is not installed

Input: Install application files

Output: Applications installs and functions as expected

Test Case Derivation: Based on R16 and R17, validate that the application can be installed and ran on different operating systems and types of computers.

How test will be performed: This test will be performed on two types of computers, a personal laptop and a desktop. On each of those types of computers it will be run on each of the operating systems, Windows, Linux and macOS to verify compatability with each OS and type of computer.

4.2 Tests for Nonfunctional Requirements

The nonfunctional requirements we will be testing for are accuracy, usability, maintainability, verifiability, understandability and reusability. Referencing the Nonfunctional requirements within the SRS document: NFR1: Accuracy, NFR4 Verifiability will reference the above tests given for the functional requirements. The Nonfunctional requirements related to usability and understandability (NFR2 and NFR 5 respectively) will reference the usability/understandability survey linked in the appendix.

4.2.1 Accuracy

The below test are to verify the accuracy of the system. They are based on NFR1 from the SRS document.

1. **Accuracy test-1**

   Type: Manual

   Initial State: The user has successfully installed the system on their device and completed the tests above for the functional requirements.

   Input/Condition: The outputs of the functional tests: Position test-2, Velocity test-2, Acceleration test-2, Shift test-2 and Shift test-4.

   Output/Result: Based on the outputted MSE's from the functional requirement tests listed we will look at the various MSE's to estimate the accuracy. Each MSE is within 0.2 of 0.

   How test will be performed: We will compare each MSE to the value of 0 as the closer the value to 0 the more accurate the prediction. An MSE with a range of 0-0.2 will indicate our system is accurate.

4.2.2 Usability

The below test are to verify the usability of the system. They are based on NFR2 from the SRS document.

1. **Usability test-1**

   Type: Manual

   Initial State:

   Input/Condition: Users within the Primary User role as well as Baja team members are asked to rate how simple the navigation process of the main interface. They are asked to rate this on a scale of (1-5) 1 being extremely difficult and 5 being extremely easy with the other options being 4: somewhat easy, 3: neutral and 2: somewhat difficult.

   Output/Result: The average output rating from all users is greater than or equal to a 4(somewhat easy or above expectations).

   How test will be performed: Each user in the test group will be provided with a survey which provides a series of questions and a scale for each option where 1 represents Poor, 2 represents below expectation, 3 represents satisfactory, 4 represents above average and 5 represents excellent. The average rating will then be calculated and must be above or equal to 4 representing the system usability is above expectations. See the usability/understandability survey in section 7.2.

2. **Usability test-2**

   Type: Manual

   Initial State: The user has successfully installed the system on their device.

   Input/Condition: Users within the Primary User role as well as Baja team members are asked to rate the features inputting parameters, adjusting parameters, viewing data outputs and saving and exporting data on how easy it was to use each feature. They are asked to rate this on a scale of (1-5) 1 being extremely difficult and 5 being extremely easy with the other options being 4: somewhat easy, 3: neutral and 2: somewhat difficult.

   Output/Result: The average output rating from all users for each listed feature is greater than or equal to a 4(somewhat easy or above expectations).

   How test will be performed: Each user in the test group will be provided with a survey which provides a series of questions and a scale for each option where

1 represents Poor, 2 represents below expectation, 3 represents satisfactory, 4 represents above average and 5 represents excellent. The average rating will then be calculated and must be above or equal to 4 representing the system usability is above expectations. See the usability/understandability survey in section 7.2.

4.2.3 Maintainability

The below test are to verify the maintainability of the system. They are based on NFR3 from the SRS document.

1. **Maintainability test-1**

   Type: Manual

   Initial State: The completed system which includes, the finalized mathematical model, GUI and 3D model.

   Input/Condition: The 2023 McMaster Baja CVT's configuration.

   Output/Result: The amount of time taken and number of lines of code modified using the 2023 CVT.

   How test will be performed: The test will be manually completed by the teams Quality Assurance Lead, where the amount of time and number of lines of code will be recorded when implementing the changes that correspond to the 2023 CVT configuration.

4.2.4 Verifiability

The below test are to verify the verifiability of the system. They are based on NFR4 from the SRS document.

1. **Verifiability test-1**

   Type: Manual

   Initial State: The tests for the functional requirements and the Nonfunctional requirement test Accuracy test1 have been completed.

   Input/Condition: The outputs of the functional tests: Position test1, Velocity test-1, Acceleration test-1, Shift test-, Shift test-1 and Shift test-3 will be run 30 times with the same input settings.

   Output/Result: The graphed behavior of each run is consistent each time.

How test will be performed: The outputs will be compared after each run to ensure that the behavior of each run is consistent for the same tunning settings.

Checklist for verfiability

- Ensure each test runs 30 times with the same input settings.
- Verify that input conditions remain unchanged between test executions.
- Identify and document any deviations between runs.
- Ensure that differences if there are any are within acceptable tolerances defined in the requirements.

4.2.5 Understandability

The below test are to verify the understandability of the system. They are based on NFR5 from the SRS document.

1. **Understandability test-1**

   Type: Manual.

   Initial State: The user has successfully installed the system on their device.

   Input/Condition: Users within the Primary User role as well as Baja team members are asked to rate how clear they found the features and functions within the system. They are asked to rate this on a scale of (1-5) 1 being extremely unclear and 5 being extremely clear with the other options being 4: somewhat clear, 3: neutral and 2: somewhat unclear.

   Output/Result: The average output rating from all users for each listed feature is greater than or equal to a 4(somewhat clear or above expectations).

   How test will be performed: Each user in the test group will be provided with a survey which provides a series of questions and a scale for each option where 1 represents Poor, 2 represents below expectation, 3 represents satisfactory, 4 represents above average and 5 represents excellent. The average rating will then be calculated and must be above or equal to 4, representing the systems' understandability is above expectations. See the usability/understandability survey in section 7.2.

2. **Understandability test-2**

   Type: Manual

   Initial State: The user has successfully installed the system on their device.

   Input/Condition: Users within the Primary User role as well as Baja team members are asked to rate their understanding of the simulation results and outputs. They are asked to rate this on a scale of (1-5) 1 being extremely unclear and 5 being extremely clear with the other options being 4: somewhat clear, 3: neutral and 2: somewhat unclear.

   Output/Result: The average output rating from all users for each listed feature is greater than or equal to a 4(somewhat clear or above expectations).

   How test will be performed: Each user in the test group will be provided with a survey which provides a series of questions and a scale for each option where 1 represents Poor, 2 represents below expectation, 3 represents satisfactory, 4 represents above average and 5 represents excellent. The average rating will then be calculated and must be above or equal to 4, representing the systems understandability is above expectations. See the usability/understandability survey in section 7.2.

4.2.6 Reusability

The below test are to verify the reusability of the system. They are based on NFR6 from the SRS document.

1. **Reusability test-1**

   Type: Manual

   Initial State: Completed system which includes, the finalized mathematical model, GUI and 3D model.

   Input/Condition: The 2023 McMaster Baja CVT's configuration.

   Output/Result: The amount of time taken and number of lines of code modified using the 2023 CVT.

   How test will be performed: The test will be manually completed by the teams Quality Assurance Lead, where the amount of time and number of lines of code will be recorded when implementing the changes to correspond to the 2023 CVT. This will indicate how long it will take for the system to be adapted to future CVT configurations.

4.3 Traceability Between Test Cases and Requirements

Requirement	Test(s)
R1	Position test-2, Velocity test-2, Acceleration test-2, Shift test-4
R2	Acceleration test-1
R3	Velocity test-1
R4	Position test-1
R5	Clamping forces test-1
R6	Clamping forces test-1
R7	Shift test-1, Shift test-2, Shift test-3
R8	Shift test-1, Shift test-2, Shift test-3
R9	Engine test-1
R10	User Interface test-1
R11	User Interface test-1
R12	User Interface test-2
R13	User Interface test-3
R14	User Interface test-4
R15	Compatibility test-1
R16	Compatibility test-1

Table 3: System Functional Requirements and Corresponding Tests

Requirement	Test(s)
NFR1	Accuracy test-1
NFR2	Usability test-1, Usability test-2
NFR3	Maintainability test-1
NFR4	Verifiability test-1
NFR5	Understandability test-1, Understandability test-2
NFR6	Reusability test-1

Table 4: System Nonfunctional Requirements and Corresponding Tests

5 Unit Test Description

This section provides an overview of the unit tests implemented for individual modules. The tests ensure correctness and reliability of key functionalities.

5.1 Unit Testing Scope

All unit tests are automated and executed using Python's built-in `unittest` framework. The tests ensure that key components behave as expected under various conditions. Test coverage reports are generated using the `coverage.py` tool.

5.2 Tests for Functional Requirements

The unit tests verify critical components, including argument parsing, conversion functions, simulation results processing, system state management, and theoretical models computations. These test are split into two directories are can be located in the simulations directory and the utils directory. The simulations directory contains 6 python files each file named descriptively to reflect the functionality they test: `test_belt_simulator.py`, `test_car_simulation.py`, `test_engine_simulation.py`, `test_load_simulation.py`, `test_primary_pulley.py`, `test_secondary_pulley.py`. Additionally, utils directory contains 5 python files each file named to reflect the functionality they test: `test_argument_parser.py`, `test_conversions.py`, `test_simulation_result.py`, `test_system_state.py`, `test_theoretical_models.py`.

The following is the file breakdown for each test suite:

- test_belt_simulator.py - This test suite will verify that the BeltSimulator's methods correctly compute centrifugal force, clamping-derived radial forces, net radial force, slack tension, and maximum transferable torque according to theoretical models.
- test_car_simulator.py - This test suite will verify that CarSimulator calculates acceleration correctly as force divided by mass.
- test_engine_simulation.py - This test suite will verify that EngineSimulator correctly initializes and computes torque, power, and angular acceleration using a mock torque curve.
- test_load_simulation.py - This test suite will verify that the LoadSimulator correctly computes vehicle load forces (incline, drag, total, gearbox) and acceleration given specific parameters.
- test_primary_pulley.py - This test suite will verify that the PrimaryPulley class correctly computes flyweight force, spring compression force, net force, and properly handles shift distance limits.

- test_secondary_pulley.py - This test suite will verify that the SecondaryPulley correctly computes helix force, spring compression force, spring torsion torque, and net force based on theoretical models and geometric parameters.
- test_argument_parser.py – This test suite will verify arguments are correctly parsed and will check that command line inputs are turned into an object for both default and custom scenarios.
- test_conversions.py - This test suite will verify that conversion functions for angular velocity, angle measurements, circumference, and inch-to-meter conversions return correct values.
- test_simulation_result.py - This test suite will verify that simulation result correctly parses ODE solutions into system state objects, writes the results to a CSV file, and calls the plot method as expected.
- test_system_state.py - This test suite will verify that the SystemState class initializes correctly and accurately converts to and from array representations.
- test_theoretical_models.py - This test suite verifies that Theoretical Models correctly compute various physics and geometry calculations—including Hooke's law, centrifugal force, air resistance, torque, gearing, friction, capstan equation, Newton's second law, and CVT and wrap angle parameters

5.3 Tests for Nonfunctional Requirements

There are no unit tests for Nonfunctional Requirements.

5.4 Traceability Between Test Cases and Modules

6 Trace to Modules

Module	Test(s)
M1	
M2	Engine test-1, Unit Tests: `test_engine_simulation.py`
M3	Velocity test-1, Acceleration test-1, Position test-1, Unit Tests: `test_car_simulator.py`
M4	Clamping forces test-1, Unit Tests: `test_primary_pulley.py`
M5	Clamping forces test-1, Unit Tests: `test_secondary_pulley.py`
M6	
M7	Tested by SciPy
M8	Shift test-1, Shift test-2, Shift test-3, Unit Tests: `test_belt_simulator.py`
M9	Unit tests for CSV Reader
M10	
M11	
M12	
M13	
M14	User Interface test-1, 2, 3, 4
M15	
M16	Unit tests for Python Runner

Table 5: Software Modules and Corresponding Tests

References

Olav Aaen. *Clutch Tuning Handbook*. Aaen Performance, Wisconsin, USA, 2007. URL https://archive.org/details/aaen-clutch-tuning-handbook-2007/page/2/mode/2up.

Baja SAE. Sae baja restricted kohler ch440 engine performance data. Available from the Baja SAE website under Document Resources, 2022. URL https://www.bajasae.net/cdsweb/gen/DownloadDocument.aspx?DocumentID=b250342f-ea19-46c2-95ea-6deb728ddb7a. Accessed: 2024-11-03.

Cameron Dunn, Grace McKenna, Kai Arseneau, and Travis Wing. System requirements specification. https://github.com/gr812b/CVT-Simulator/blob/develop/docs/SRS/SRS.pdf, 2024.

7 Appendix

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

constant	value
$v_{\max}$	16.67 m/s
ω_{idle}	180 rad/s
$\omega_{\max}$	420 rad/s
$\omega_{\text{peak\_power}}$	6987.24 W

Table 6: Verification and Validation Constants

7.2 Usability Survey Questions

Usability/Understandability survey: https://forms.office.com/r/RkeDW31ZTS

7.3 Expected Graphs

This section includes all referenced graphs for various tests. Please note that all values may not be accurate, and it is the shape that is being tested for.

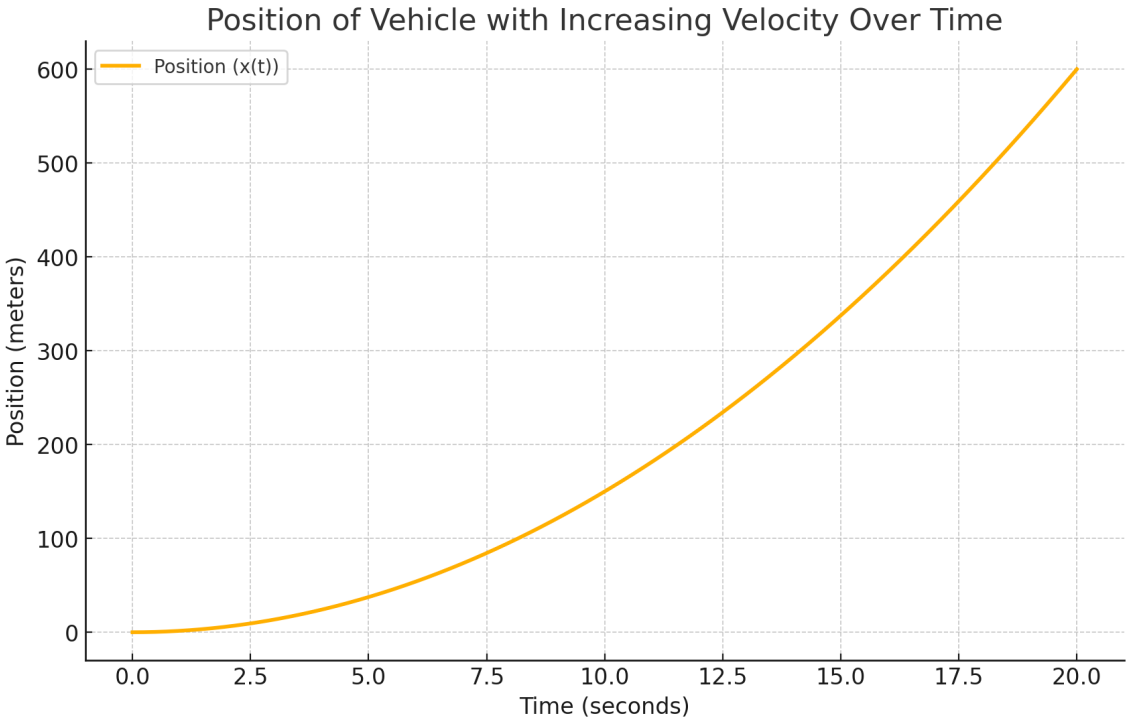

Figure 1: Position over time graph

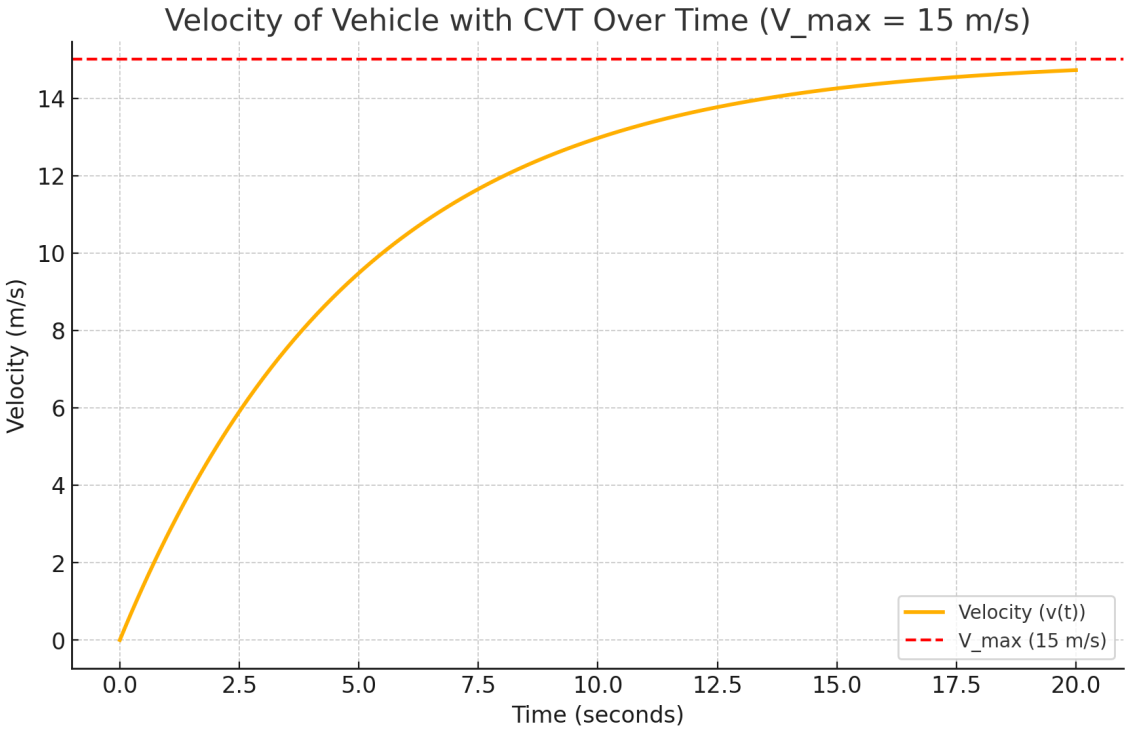

Figure 2: Velocity over time graph

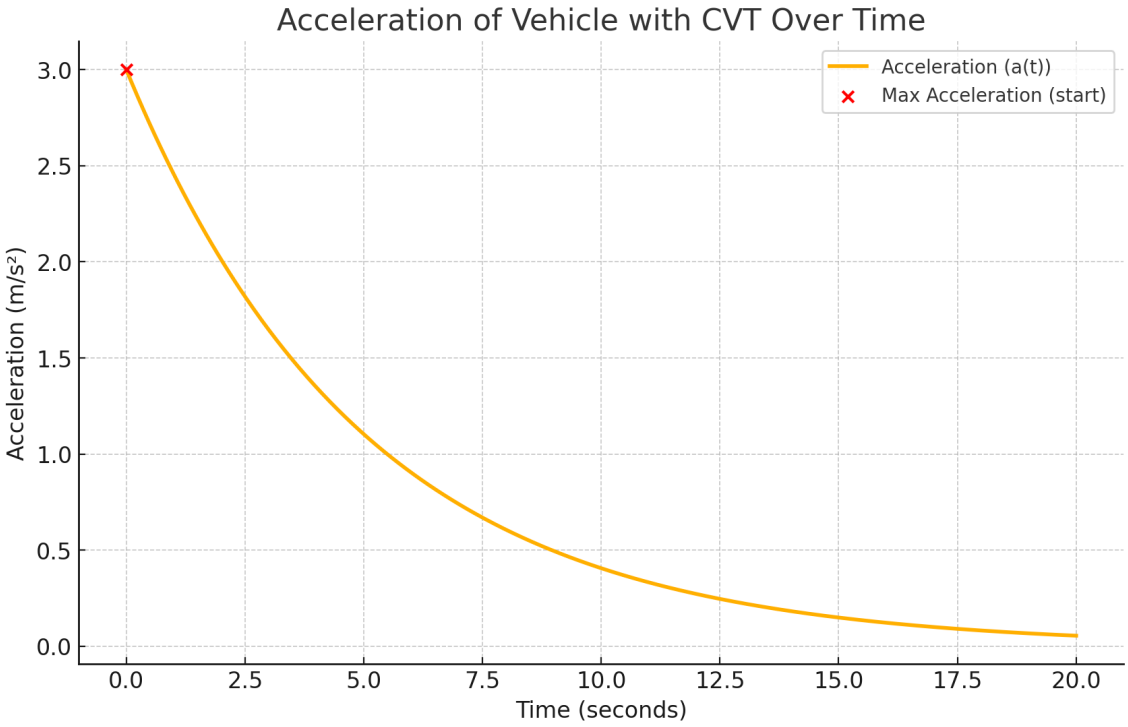

Figure 3: Acceleration over time graph

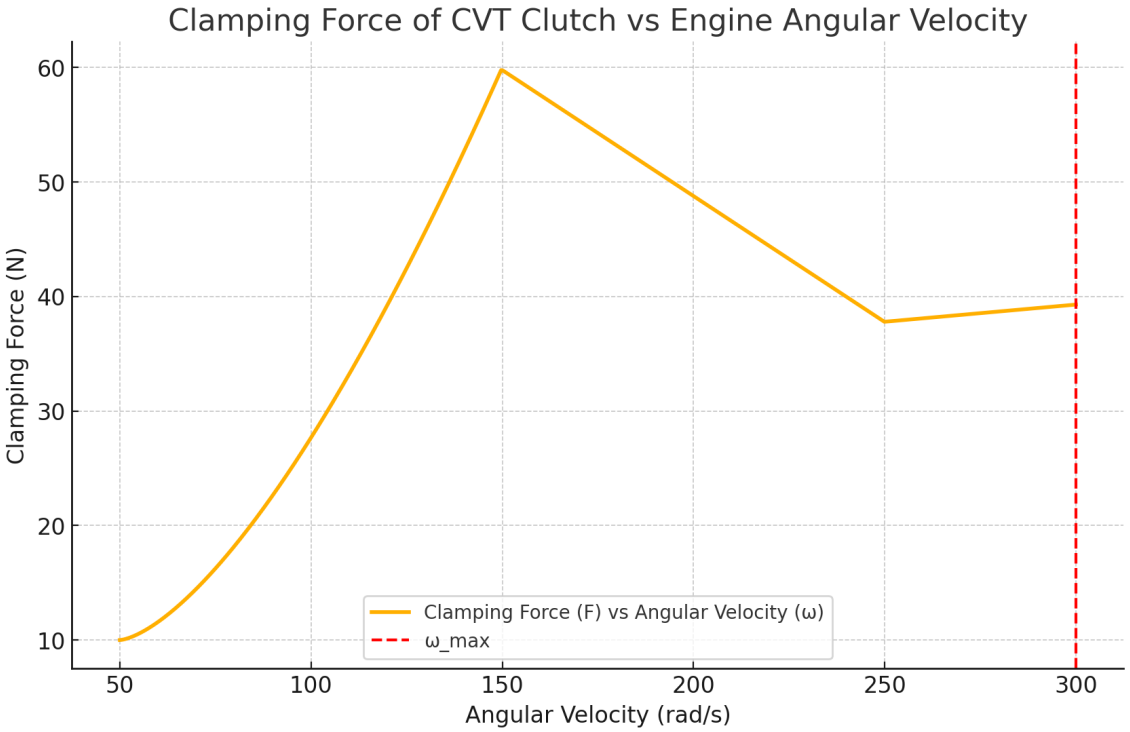

Figure 4: Clamping force against engine angular velocity graph

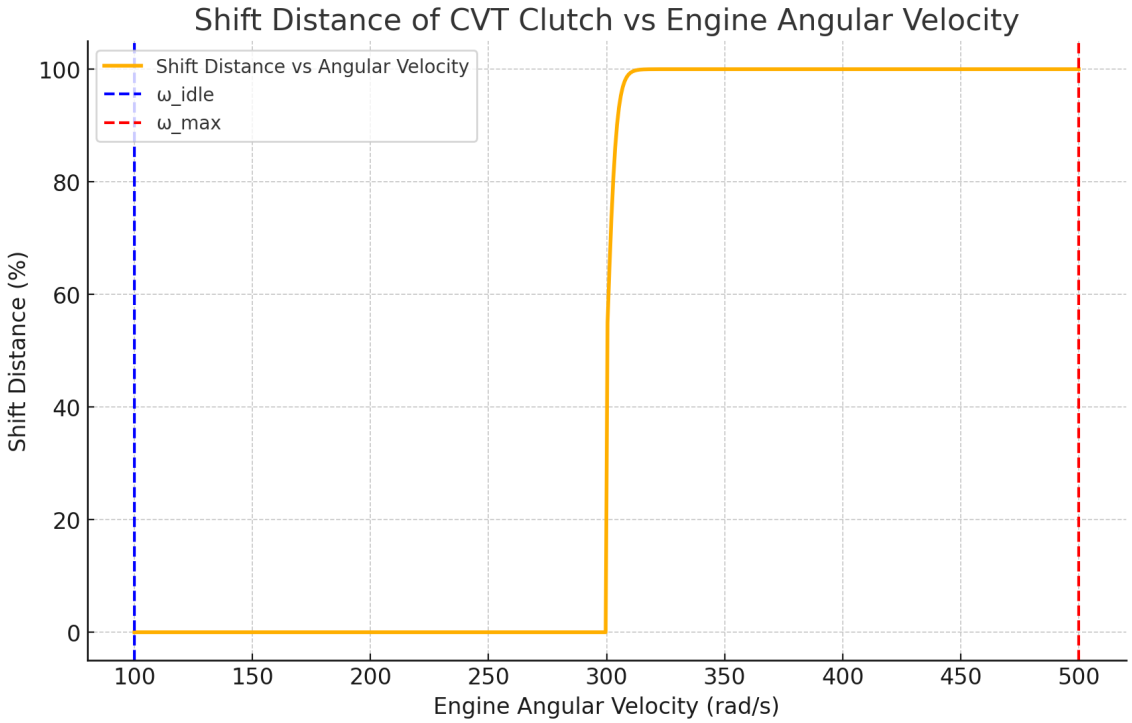

Figure 5: Shift distance against engine angular velocity graph

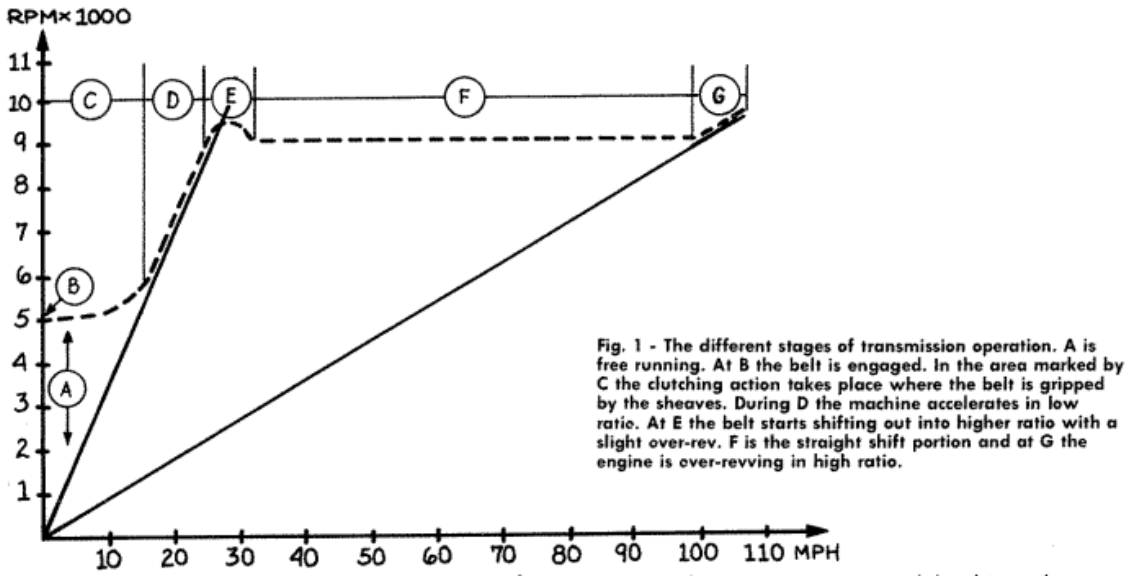

Figure 6: Shift curve graph Aaen (2007)

B / C - Clutching

Belt engagement starts with a lot of slip when the vehicle is stationary, increasing to 100% engagement when the vehicle is accelerating along the low ratio line. In between these two points there is slippage between the belt and the sheave. After the flyweight force overcomes the spring tension, the RPM has to increase and give more sheave pressure on the belt until there is no slippage and full engagement. This takes place as the vehicle is starting to move and as the engine speed and vehicle speed intersect at the low ratio line. This area is often critical for good and smooth take-off. Incorrect matching of springs and weights may lead to excessive slippage and belt wear.

The clutching phase begins when the flyweight force overcomes the pretension, and lasts until the flyweights have generated enough side force to transfer the engine torque without slipping the belt.

D - Low Ratio Acceleration

After the sheaves have gripped the belt and they are transferring the torque without slipping, the machine will accelerate along the low ratio line. The engine speed will continue to increase as if you were in low gear in your car. While the vehicle speed is increasing the belt remains in low ratio at the bottom of the driving sheaves. The

belt will not start moving out on the sheave until the flyweight force has become large enough to overcome both the pressure spring and the side pressure of the belt on the driven sheave. When the engine speed has built up to create enough centrifugal force to overcome the pressure spring and the driven sheave pressures, the belt will start shifting out and, the ratio will change. This is called the "shift point" and should coincide with the power peak of the engine.

Between full engagement and the shift point, centrifugal forces are larger than the pressure spring but less than the belt pressure from the driven clutch.

E - Shift-Out-Point

As the RPM of the driving clutch increases in low ratio, the centrifugal force from the flyweights will eventually be large enough to overcome the tension on the belt from the driven clutch. When this occurs, the transmission has reached the shift out point and the belt will start to move outward on the driving clutch and inward on the driven clutch. As the belt starts to move, the drive ratio is changed.

Ideally, the shift out point should be at the power peak of the engine and the shift curve should be straight from there through high ratio. In practice, it may sometimes end up a little different. If you have a heavy machine and a small high revving engine, you may want the shift out point to be several hundred RPMs higher than the power peak. As the machine increases acceleration the engine may be pulled down in RPM by the increasing load as the transmission shifts out. By over-revving slightly in the beginning the engine will be pulled down on to the power peak, and smooth acceleration continues. If the shift out point was right on the power peak, the reduction in RPM would have pulled the transmission down off the peak and a more sluggish acceleration would have occurred as the transmission worked its RPM up again.

This shift out over-rev is of short duration, and highly individual depending on the combination of machine weight and engine power. On a super light drag machine with a big engine, you may get away with starting the shift-out slightly before the power peak as the power will pull you right through without an RPM drop. The best way to achieve the initial over-rev is to modify the shift curve of the flyweight right off the engagement point. This is usually done by extending the engagement flat into the shift curvature. The length and shape of the transition from engagement to shift curve will determine the amount and duration of the shift out over-rev.

The shift out point occurs when the flyweight force overcomes the belt pressure from the driven clutch.

F - Straight Shift

The ideal shift-curve is "straight" between the low and high ratio. This means that the engine speed is held constant at the power peak while the transmission is shifting out and the vehicle speed is increasing. With the exception of a slight shift-out over-rev as discussed in the previous section, a straight shift at the power peak will give the maximum performance. To maintain a straight shift is not an easy task. As we saw under the discussion of the belt pressure requirements, the pressure on the belt is actually decreasing as the transmission shifts out. This decrease in belt tension may be as much as 50

Since the flyweight system works against the pressure spring in the driving clutch, the loads and rates of this spring also influence straight shift and engine RPM. Flyweight curvatures have been experimented with and fine-tuned by the manufacturers over the last 15 years, and they provide a good selection of tuning components. Polaris is a good example; they provide two basic flyweight curvatures, trail and racing. The trail curvature has lower engagement and softer shift-out than the racing curvature. Each flyweight group is available with weights in 2-4 gram increments for each curvature. Each manufacturer also has a large selection of springs with different pretensions and rates for calibration purposes. Achieving the correct combination of flyweight and spring to obtain a straight shift at the correct engine speed is the task of the tuner. This will be covered in greater detail under the Selection of Components and Testing chapters.

Straight shift is obtained by matching curvature and spring rates; correct engine speed is dependent on the weight of the flyweight.

G - Over-Run

When the transmission is shifted all the way out into high ratio, engine speed must increase along the high ratio line as vehicle speed increases.

As the engine speed increases, the power curve goes off the peak on the back side and less power is available to propel the machine. Maximum speed cannot be obtained if you are off the peak; this means the machine is geared too low. There may be conditions where this is done on purpose because acceleration is more important than top speed. In most cases the machines are geared to give maximum top speed, which means they will seldom reach high ratio and over-run is not usually experienced.

In over-run, the clutches are shifted all the way out and the flyweight forces have no more influence on the engine speed. Engine speed will increase along the fixed high ratio line. Aaen (2007)

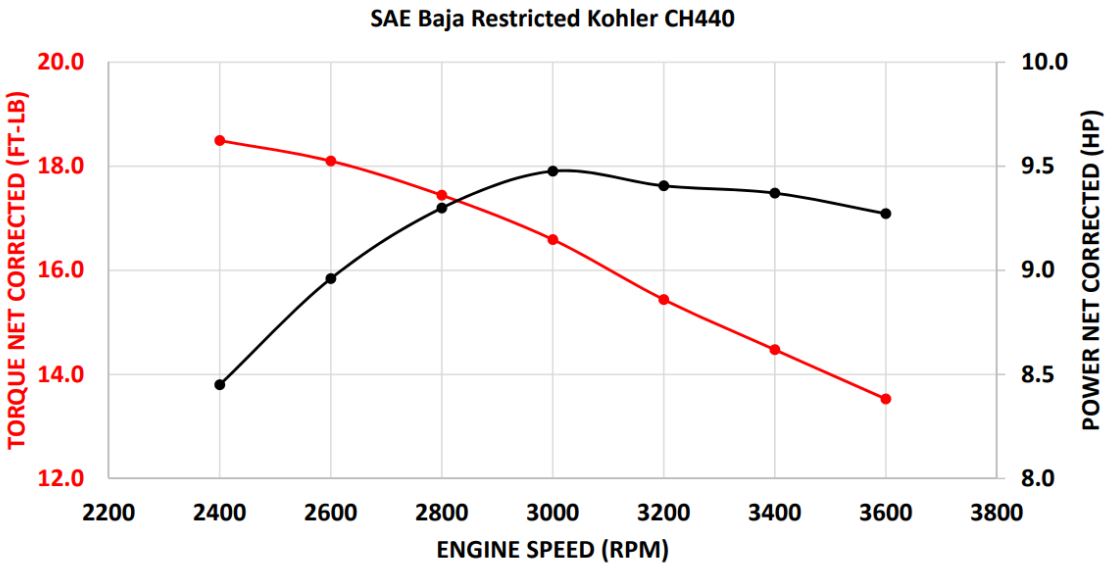

Figure 7: Engine torque and power against engine angular velocity Baja SAE (2022)

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
   When writing this deliverable there was a lot of things that went well. Firstly the team had a long sit down with our capstone supervisor Dr Smith to discuss the process of how we will be validating our simulation. We were able to come up with a great plan for how the simulation will be validated against real world data and how to incorporate that into this document and the project as a whole. The team was also able to have some great communication with each other when figuring out how to write the various kinds of tests for the system. Since there was a lot of different types of tests we were constantly bouncing ideas off each other to figure out the best way to move forward. Team communication such as that is always a good thing and this deliverable specifically had a lot of that in it. Another thing that went well was the team was able to figure out who on the McMaster Baja team will help us validate our simulation once the implementation is complete. We outlined a list of people that will help us in section 3 of this document, and we were able to get in contact with them, and they are all on board to help us validate the simulation.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   When writing this deliverable there was only one pain point that needed to be resolved. The first one was how to write the functional tests for the math component of the system. Due to the nature of our project the simulation

outputs are not something that will be able to be verified/validated numerically or with a simple test. Instead, we will need to compare the outputs of the simulation to the outputs of the physical car. Therefore, we had to change the way that we wrote the functional requirements, and instead of writing them as a simple test we wrote them as a comparison between some specific part of the simulation output and against the same data gathered from the physical car. The way this pain point was resolved was through sitting down with our capstone supervisor Dr Smith and talked the process of how we will be validating our simulation and he was the one who gave us the recommendation to write the functional requirements in this way.

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

   In order to successfully complete the verification and validation of our project the team will need to acquire various knowledge and skills regarding testing. Some areas that we will need to gather skills in will be in areas of testing such as dynamic testing, static testing, unit testing and system testing. As well the team will need to learn how to use specific tools for the various languages that we will be using. For python this will include:

   - coverage
   - unittest
   - Flake8 (linter)
   - black (code formatter)

   For C# this will include:

   - SonarLint
   - StyleCop
   - UTF (Unit Test Framework)
   - UTR (Test framework for Unity)

   As well beyond testing tools and knowledge the team will also have to get familiar with how to validate the simulation output against the physical car data. This involves working with the McMaster Baja teams data acquisition

system to gather the data from the car and then how to compare that data to the simulation output. This will involve understanding different types of data involved with the CVT that were outlined in section 4 and learning what a correct graph of those kinds of data look like.

Team skill breakdown

- Travis: Need to familiarize himself with the C# testing tools and how to use them. As well, he will need to learn how to validate the simulation output against the physical car data.
- Grace: Need to familiarize herself with the C# testing tools and how to use them. As well, she will need to learn how to validate the simulation output against the physical car data.
- Kai: Need to familiarize himself with the Python testing tools and how to use them. He is already familiar with the data acquisition system and how to gather data from the car.
- Cam: Need to familiarize himself with the Python testing tools and how to use them. He is already familiar with the data acquisition system and how to gather data from the car.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   One way to acquire the knowledge and skills needed to successfully complete the verification and validation of our project is to take watch some videos online about how to use them. There are many videos that are available that can teach you how to use the various testing tools that we will need to use. Another way to acquire the knowledge is to read the documentation for the tools libraries that we are going to use. This would involve reading the documentation for the testing tools that we will be using and learning how to use them from there. After reading some documentation you can then try to implement the tools in a small project to get a feel for how they work.

   Each team member will be using the second approach to acquire these knowledge and skills. We felt that the best way to learn testing tools is to read the documentation on how they should be used. Then once we are implementing the POC we can use that smaller project to get a feel for how the tools work

and how to use them. This will give us the base knowledge and the practicality of how to use the tools in a real project.