Module Interface Specification for Double Pendulum

Zhi Zhang

November 21, 2019

1 Revision History

Date	Version	Notes
Nov.13	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/SRS/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for Double Pendulum, a software which determines the motion of a double pendulum given the initial conditions from user inputs.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/best-zhang-zhi/CAS741Project.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Double Pendulum.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
string	$char^n$	a sequence of alphanumeric and special characters
list	$real^n$	a list of real numbers

The specification of Double Pendulum uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Double Pendulum uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	User Input Module Angular Acceleration Module Runge Kutta Module Output Module
Software Decision Module	GUI Module Graph Module

Table 1: Module Hierarchy

6 MIS of User Input Module

The user input module is responsible for acquiring data from users. The module verifies whether or not the input data meets the constraints, and output the verified data to the Angular Acceleration Module.

6.1 Module

Input

6.2 Uses

N/A

6.3 Syntax

6.3.1 Exported Constants

- m_1: real
- m_2: real
- L_1: real
- L_2: real
- θ_1: real
- θ_2: real
- g: real

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
GetMass1	real	real	IN_NEGATIVE_MASS
GetMass2	real	real	IN_NEGATIVE_MASS
GetLength1	real	real	IN_NEGATIVE_LENGTH
GetLength2	real	real	IN_NEGATIVE_LENGTH
GetAngle1	real	real	-
GetAngle2	real	real	-
GetGravityConstant	real	real	IN_NEGATIVE_GRAVITY

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

keyboard: users type in the required data using keyboard [I'm not sure if I should include this —SS]

6.4.3 Assumptions

- Users only input numerical numbers

6.4.4 Access Routine Semantics

GetMass1(m_1):

- transition: N/A
- output: $out := self$
- exception: $exc := m_1 \leq 0 \Rightarrow IN\_NEGATIVE\_MASS$

GetMass2(m_2):

- transition: N/A
- output: $out := self$
- exception: $exc := m_2 \leq 0 \Rightarrow IN\_NEGATIVE\_MASS$

GetLength1(L_1):

- transition: N/A
- output: $out := self$
- exception: $exc := L_1 \leq 0 \Rightarrow IN\_NEGATIVE\_LENGTH$

GetLength2(L_2):

- transition: N/A
- output: $out := self$
- exception: $exc := L_2 \leq 0 \Rightarrow IN\_NEGATIVE\_LENGTH$

GetAngle1(θ_1):

- transition: N/A
- output: $out := self$
- exception: N/A

GetAngle2(θ_2):

- transition: N/A
- output: $out := self$
- exception: N/A

GetGravityConstant(g):

- transition: N/A
- output: $out := self$
- exception: $exc := g \leq 0 \Rightarrow IN\_NEGATIVE\_GRAVITY$

6.4.5 Local Functions

N/A

7 MIS of Angular Acceleration Module

The angular acceleration module uses data from user input module 6, and outputs the equations of the angular acceleration.

7.1 Module

Acceleration

7.2 Uses

Input

7.3 Syntax

7.3.1 Exported Constants

- θ_1''
- θ_2''

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
getAcc1	Input	string	-
getAcc2	Input	string	-

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

getAcc1(m_1, m_2, L_1. L_2, θ_1, θ_2, g):

- transition: N/A
- output:

$$\theta_1'' = \frac{-g(2m_1 + m_2)sin\theta_1 - m_2 g sin(\theta_1 - 2\theta_2) - 2sin(\theta_1 - \theta_2)m_2(\theta_2'^2 L_2 + \theta_2'^2 L_1 cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))}$$

- exception: N/A

getAcc2(m_1, m_2, L_1. L_2, θ_1, θ_2, g):

- transition: N/A
- output:

$$\theta_2'' = \frac{2sin(\theta_1 - \theta_2)(\theta_1' L_1(m1 + m2) + g(m_1 + m_2)cos(\theta_1) + (\theta_2')^2 L_2 m_2 cos(\theta_1 - \theta_2)}{L_2(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))}$$

- exception: N/A

7.4.5 Local Functions

N/A

8 MIS of Runge Kutta Module

Multi-variable Rung-kutta algorithm will be used to approximate the angular velocity of the two mass. Neumann This module outputs two lists of point velocity for each mass.

8.1 Module

RungeKutta

8.2 Uses

Acceleration

8.3 Syntax

8.3.1 Exported Constants

- $\theta_1{}'$
- $\theta_2{}'$

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
getVelocity1	string	list	-
getVelocity2	string	list	-

8.4 Semantics

8.4.1 State Variables

- $a_n = \theta_1{}''(0.1, \theta_1{}')$
- $b_n = \theta_1{}''(0.105, \theta_1{}' + 0.005a_n)$
- $c_n = \theta_1{}''(0.105, \theta_1{}' + 0.005b_n)$
- $d_n = \theta_1{}''(0.11, \theta_1{}' + 0.01b_n)$

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

getVelocity1():

- transition: N/A
- output: $\forall n : \mathbb{Z} | n \in [0..10000] : \theta_1(n+1) = \theta_1(n) + 0.01/6(a_n + 2b_n + 2c_n + d_n)$
- exception: N/A

getVelocity2():

- transition: N/A
- output: $\forall n : \mathbb{Z} | n \in [0..10000] : \theta_2(n+1) = \theta_2(n) + 0.01/6(a_n + 2b_n + 2c_n + d_n)$
- exception: N/A

8.4.5 Local Functions

N/A

9 MIS of Output Module

The output module takes the point velocity of two masses, and outputs the point position of them in list form.

9.1 Module

Output

9.2 Uses

RungeKutta

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Constants

- θ_1
- θ_2

9.3.3 Exported Access Programs

Name	In	Out	Exceptions
getPosition1	list	list	-
getPosition2	list	list	-

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

getPosition1($(\theta_1)_n, (\theta_1)_0$):

- transition: N/A
- output: $\forall n : \mathbb{Z} | n \in [1..9999] : \theta_1(n+1) = \theta_1(n-1) + \theta_1(n)$
- exception: N/A

9.4.5 Local Functions

N/A

10 MIS of Interval Data Structure Module

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

10.1 Module

[Short name for the module —SS]

10.2 Uses

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

10.4 Semantics

10.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

10.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

10.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

10.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

10.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

11 MIS of Equation Data Structure Module

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

11.1 Module

[Short name for the module —SS]

11.2 Uses

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

11.4 Semantics

11.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

11.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

11.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

11.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

11.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

12 MIS of Output Data Structure Module

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

12.1 Module

[Short name for the module —SS]

12.2 Uses

12.3 Syntax

12.3.1 Exported Constants

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

12.4 Semantics

12.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

12.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

12.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

12.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

12.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

Erik Neumann. Runge kutta. URL https://www.myphysicslab.com/explain/runge-kutta-en.html.

13 Appendix

[Extra information if required —SS]