Module Interface Specification for Double Pendulum

Zhi Zhang

November 29, 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	Input Parameters Module Output Format Module Acceleration Equations Module Velocity ODEs Module Control Module
Software Decision Module	Sequence Data Structure Module ODE Solver Module Plotting Module

Table 1: Module Hierarchy

6 MIS of Control Module

The control module provides the main program.

6.1 Module

main

6.2 Uses

Param (Section 7), Acceleration (Section 8), Velocity (Section 9), Solver (Section 10), Plot (Section 11), Output (Section 12)

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

- Users only input numerical numbers

6.4.4 Access Routine Semantics

main():

- transition: Modify the state of Param module and the environment variables for the Plot and Output modules by following steps

```
Get(filenameIn: String) and (filenameOut: string) from user

load_params(filenameIn)
#Find angular velocity function ($\theta_1, \theta_2$)
${\theta_1}'$:= solve(Acceleration_Top,0.0, 0.0,$t$) [Where is the value of $t$ given? —SS]
${\theta_2}'$:= solve(Acceleration_Bottom,0.0, 0.0,$t$)
#Find angular position function ($\theta_1, \theta_2$)
$\theta_1$:= solve(Velocity_Top,0.0, 0.0,$t$)
$\theta_2$:= solve(Velocity_Bottom,0.0, 0.0,$t$)
```

[You have a second order ODE, but looking at your document, it appears that you haven't really thought about how to solve it. The SWHS example that you based your document on is for a first order ODE. You'll need to make changes. Not big changes, but changes. For instance, your solver modules shows the equation for solving a first order ODE. I think the easiest thing for you to do is to focus on what you actually need. You need theta as a function of time. You don't really need the angular velocity. Therefore, your solver should take the definition of theta" and return theta. You can leave the angular velocity out of it altogether. Your solver module then just has to say mathematically what it means to solve a second order ODE. —SS]

```
#Output calculated values to a file and to a plot.

plot($\theta_1$, $\theta_2$)

output(filenameOut, $\theta_1$, $\theta_2$)
```

6.4.5 Local Functions

N/A

7 MIS of Input Parameter Module

The secrets of this module are the data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

7.1 Module

Param

7.2 Uses

N/A

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
load_params	string	-	FileError
verify_params	-	-	NEGATIVE_MASS, NEGATIVE_LENGTH, NEGATIVE_GRAVITY, NEGATIVE_TIME
m_1	-	$\mathbb{R}$	-
m_2	-	$\mathbb{R}$	-
L_1	-	$\mathbb{R}$	-
L_2	-	$\mathbb{R}$	-
θ_1	-	$\mathbb{R}$	-
θ_2	-	$\mathbb{R}$	-
g	-	$\mathbb{R}$	-
t	-	$\mathbb{R}$	-

7.4 Semantics

7.4.1 State Variables

m_1: $\mathbb{R}$

m_2: $\mathbb{R}$

L_1: $\mathbb{R}$
L_2: $\mathbb{R}$
θ_1: $\mathbb{R}$
θ_2: $\mathbb{R}$
g : $\mathbb{R}$
t: $\mathbb{R}$

7.4.2 Environment Variables

inputFile: sequence of string *#f[i] is the ith string in the text file f*

7.4.3 Assumptions

- load_params will be called before the values of any state variables will be accessed.
- The file contains the string equivalents of the numeric values for each input parameter in order, each on a new line. The order is the same as in the table in R1 of the SRS. Any comments in the input file should be denoted with a '#' symbol.

7.4.4 Access Routine Semantics

Param.m_1:

- transition: N/A
- output: $out := m_1$
- exception: none

Param.m_2:

- transition: N/A
- output: $out := m_2$
- exception: none

Param.L_1:

- transition: N/A
- output: $out := L_1$
- exception: none

Param.L_2:

- transition: N/A
- output: $out := L_2$
- exception: none

Param.θ_1:

- transition: N/A
- output: $out := \theta_1$
- exception: N/A

Param.θ_2:

- transition: N/A
- output: $out := \theta_2$
- exception: N/A

Param.g:

- transition: N/A
- output: $out := g$
- exception: none

Param.t:

- transition: N/A
- output: $out := t$
- exception: none

load_params(s):

- transition: The filename s is first associated with the file f. inputFile is used to modify the state variables using the following procedural specification:
    1. Read data sequentially from inputFile to populate the state variables from R1 (m_1 to t).
    2. verify_params()
- output: N/A
- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError

verify_params():

- transition: N/A
- out: out := none
- exception: exc :=

$\neg(m_1 > 0)$	$\Rightarrow$ NEGATIVE_MASS
$\neg(m_2 > 0)$	$\Rightarrow$ NEGATIVE_MASS
$\neg(L_1 > 0)$	$\Rightarrow$ NEGATIVE_LENGTH
$\neg(L_2 > 0)$	$\Rightarrow$ NEGATIVE_LENGTH
$\neg(g > 0)$	$\Rightarrow$ NEGATIVE_GRAVITY
$\neg(T > 0)$	$\Rightarrow$ NEGATIVE_TIME

7.4.5 Local Functions

N/A

8 MIS of Acceleration Equations Module

8.1 Module

Acceleration

8.2 Uses

Input

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Acceleration_Top	-	$\mathbb{R}$	-
Acceleration_Bottom	-	$\mathbb{R}$	-

[You probably do not want to split the acceleration into two access programs. You are solving a system of ODEs, so the ODE solver will expect both inputs (top and bottom). You are better off outputting a sequence of two values. —SS]

8.4 Semantics

8.4.1 State Variables

- θ_1''
- θ_2''

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

Acceleration_Top():

- transition:

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

- output: N/A
- exception: N/A

Acceleration_Bottom():

- transition:

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

- output: N/A
- exception: N/A

8.4.5 Local Functions

N/A

9 MIS of Velocity ODEs Module

9.1 Module

Velocity [As mentioned in the first module, I don't think you need the Velocity module. —SS]

9.2 Uses

Acceleration

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Velocity_Top	-	$\mathbb{R} \rightarrow \mathbb{R}^2$	-
Velocity_Bottom	-	$\mathbb{R} \rightarrow \mathbb{R}^2$	-

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

Velocity_Top():

- transition: N/A
- output: $out := \iint {\theta_1}'' \, d{\theta_1}' \, d{\theta_2}'$
- exception: N/A

Velocity_Bottom():

- transition: N/A
- output: $out := \iint \theta_2'' \, d\theta_1' \, d\theta_2'$
- exception: N/A

9.4.5 Local Functions

N/A

10 MIS of ODE Solver Module

#Bold font is used to indicate variables that are a sequence type

10.1 Module

Solver($n : \mathbb{N}$) *#n is the length of the sequences*

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	**In**	**Out**	**Except.**
solve	$\mathbf{f} : (\mathbb{R}^n \rightarrow \mathbb{R}^{n+1}), t_0 : \mathbb{R}, \mathbf{y}_0 : \mathbb{R}^n, t_{\text{fin}} : \mathbb{R}$	$t_1 : \mathbb{R}, \mathbf{y} : (\mathbb{R} \rightarrow \mathbb{R})^{n+1}$	-

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

#Solving $\int\int y = \mathbf{f}(t, \mathbf{y}(t))$

solve($\mathbf{f}, t_0, \mathbf{y}_0, t_{\text{fin}}$):

- output: $out := t_1, \mathbf{y}(t)$ where

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(s, \mathbf{y}(s)) ds$$

  with t_1 determined to be 0.05 second. $\mathbf{y}(t)$ is calculated from $t = t_0$ to $t = t_1$.

- exception: N/A

10.4.5 Local Functions

N/A

11 MIS of Plotting Module

11.1 Module

Plot

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
plot	$\theta_1(t) : \mathbb{R} \rightarrow \mathbb{R}, \theta_2(t) : \mathbb{R} \rightarrow \mathbb{R}$	-	-

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

win: 2D sequence of pixels displayed on the screen

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

plot(θ_1, θ_2):

- transition: Modify win to display a plot where the vertical axis is angular position and horizontal axis is time. The time should run from 0 to t.
- output: N/A
- exception: N/A

11.4.5 Local Functions

N/A

12 MIS of Output Module

12.1 Module

Output

12.2 Uses

Param(Section 7)

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
output	fname: string, $\theta_1(t) : \mathbb{R} \rightarrow \mathbb{R}, \theta_2(t) : \mathbb{R} \rightarrow \mathbb{R}$	-	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

file: A text file

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

output(fname, θ_1, $\theta_2 . t$):

- transition: Write to environment variable named fname the following: the input parameters from Param, and the calculated values θ_1, $\theta_2$from times 0 to t. The functions will be output as sequences in this file. The spacing between points in the sequence should be selected so that the motion pf the pendulums is captured in the data.
- output: N/A

- exception: N/A

12.4.5 Local Functions

N/A

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`.

13 Appendix

Table 2: Possible Exceptions

Message ID	**Error Message**
FileError	Error: The inputed file is invalid
NEGATIVE_MASS	Error: Mass of the pendulum must be > 0
NEGATIVE_LENGTH	Error: Length of rods must be > 0
NEGATIVE_GRAVITY	Error: Gravity must be > 0
NEGATIVE_TIME	Error: Time for the simulation must be > 0