

# Module Interface Specification for Bridge Corrosion

Cynthia Liu

July 24, 2024

# 1 Revision History

Date	Version	Notes
Mar. 8, 2024	0	Initial Release
March 18, 2024	0.1	Edit according to feedback from peer review
April 5, 2024	1	Edit according to feedback from Dr. Smith
July 3, 2024	1.1	Add modules for chloride on deck
July 24, 2024	1.2	Match the function name to the actual one

## 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [SRS](#).

[Also add any additional symbols, abbreviations or acronyms —SS]

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>1</b>
<b>6</b>	<b>MIS of Control Module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	3
6.3	Syntax . . . . .	3
6.3.1	Exported Constants . . . . .	3
6.3.2	Exported Access Programs . . . . .	3
6.4	Semantics . . . . .	3
6.4.1	State Variables . . . . .	3
6.4.2	Environment Variables . . . . .	3
6.4.3	Assumptions . . . . .	3
6.4.4	Access Routine Semantics . . . . .	4
6.4.5	Local Functions . . . . .	4
<b>7</b>	<b>MIS of Input Module</b>	<b>5</b>
7.1	Module . . . . .	5
7.2	Uses . . . . .	5
7.3	Syntax . . . . .	5
7.3.1	Exported Constants . . . . .	5
7.3.2	Exported Access Programs . . . . .	5
7.4	Semantics . . . . .	5
7.4.1	State Variables . . . . .	5
7.4.2	Environment Variables . . . . .	5
7.4.3	Assumptions . . . . .	6
7.4.4	Access Routine Semantics . . . . .	6
7.4.5	Local Functions . . . . .	6
<b>8</b>	<b>MIS of Data Searching Module</b>	<b>7</b>
8.1	Module . . . . .	7
8.2	Uses . . . . .	7
8.3	Syntax . . . . .	7
8.3.1	Exported Constants . . . . .	7
8.3.2	Exported Access Programs . . . . .	7

8.4	Semantics . . . . .	7
8.4.1	State Variables . . . . .	7
8.4.2	Environment Variables . . . . .	7
8.4.3	Assumptions . . . . .	7
8.4.4	Access Routine Semantics . . . . .	8
8.4.5	Local Functions . . . . .	8
<b>9</b>	<b>MIS of Output Visualization Module</b>	<b>9</b>
9.1	Module . . . . .	9
9.2	Uses . . . . .	9
9.3	Syntax . . . . .	9
9.3.1	Exported Constants . . . . .	9
9.3.2	Exported Access Programs . . . . .	9
9.4	Semantics . . . . .	9
9.4.1	State Variables . . . . .	9
9.4.2	Environment Variables . . . . .	9
9.4.3	Assumptions . . . . .	9
9.4.4	Access Routine Semantics . . . . .	10
9.4.5	Local Functions . . . . .	10
<b>10</b>	<b>MIS of Data Model Reading Module</b>	<b>11</b>
10.1	Module . . . . .	11
10.2	Uses . . . . .	11
10.3	Syntax . . . . .	11
10.3.1	Exported Constants . . . . .	11
10.3.2	Exported Access Programs . . . . .	11
10.4	Semantics . . . . .	11
10.4.1	State Variables . . . . .	11
10.4.2	Environment Variables . . . . .	12
10.4.3	Assumptions . . . . .	12
10.4.4	Access Routine Semantics . . . . .	12
10.4.5	Local Functions . . . . .	12
<b>11</b>	<b>MIS of Constant Module</b>	<b>13</b>
11.1	Module . . . . .	13
11.2	Uses . . . . .	13
11.3	Syntax . . . . .	13
11.3.1	Exported Constants . . . . .	13
11.3.2	Exported Access Programs . . . . .	13
11.4	Semantics . . . . .	13
11.4.1	State Variables . . . . .	13
11.4.2	Environment Variables . . . . .	13
11.4.3	Assumptions . . . . .	14

11.4.4	Access Routine Semantics . . . . .	14
11.4.5	Local Functions . . . . .	14
<b>12</b>	<b>MIS of Deicing Salts Calculation Module</b>	<b>15</b>
12.1	Module . . . . .	15
12.2	Uses . . . . .	15
12.3	Syntax . . . . .	15
12.3.1	Exported Constants . . . . .	15
12.3.2	Exported Access Programs . . . . .	15
12.4	Semantics . . . . .	15
12.4.1	State Variables . . . . .	15
12.4.2	Environment Variables . . . . .	15
12.4.3	Assumptions . . . . .	15
12.4.4	Access Routine Semantics . . . . .	15
12.4.5	Local Functions . . . . .	16
<b>13</b>	<b>MIS of Melted Water Thickness Module</b>	<b>17</b>
13.1	Module . . . . .	17
13.2	Uses . . . . .	17
13.3	Syntax . . . . .	17
13.3.1	Exported Constants . . . . .	17
13.3.2	Exported Access Programs . . . . .	17
13.4	Semantics . . . . .	17
13.4.1	State Variables . . . . .	17
13.4.2	Environment Variables . . . . .	17
13.4.3	Assumptions . . . . .	17
13.4.4	Access Routine Semantics . . . . .	17
13.4.5	Local Functions . . . . .	18
<b>14</b>	<b>MIS of Single Water SAS Calculation Module</b>	<b>19</b>
14.1	Module . . . . .	19
14.2	Uses . . . . .	19
14.3	Syntax . . . . .	19
14.3.1	Exported Constants . . . . .	19
14.3.2	Exported Access Programs . . . . .	19
14.4	Semantics . . . . .	19
14.4.1	State Variables . . . . .	19
14.4.2	Environment Variables . . . . .	19
14.4.3	Assumptions . . . . .	19
14.4.4	Access Routine Semantics . . . . .	19
14.4.5	Local Functions . . . . .	20
14.4.6	Local Functions . . . . .	20

<b>15 MIS of Single Chloride Ions SAS Calculation Module</b>	<b>21</b>
15.1 Module . . . . .	21
15.2 Uses . . . . .	21
15.3 Syntax . . . . .	21
15.3.1 Exported Constants . . . . .	21
15.3.2 Exported Access Programs . . . . .	21
15.4 Semantics . . . . .	21
15.4.1 State Variables . . . . .	21
15.4.2 Environment Variables . . . . .	21
15.4.3 Assumptions . . . . .	21
15.4.4 Access Routine Semantics . . . . .	22
15.4.5 Local Functions . . . . .	22
<b>16 MIS of All Chloride Ions SAS Calculation Module</b>	<b>23</b>
16.1 Module . . . . .	23
16.2 Uses . . . . .	23
16.3 Syntax . . . . .	23
16.3.1 Exported Constants . . . . .	23
16.3.2 Exported Access Programs . . . . .	23
16.4 Semantics . . . . .	23
16.4.1 State Variables . . . . .	23
16.4.2 Environment Variables . . . . .	23
16.4.3 Assumptions . . . . .	23
16.4.4 Access Routine Semantics . . . . .	24
16.4.5 Local Functions . . . . .	24
<b>17 MIS of Chloride on Pier Calculation Module</b>	<b>25</b>
17.1 Module . . . . .	25
17.2 Uses . . . . .	25
17.3 Syntax . . . . .	25
17.3.1 Exported Constants . . . . .	25
17.3.2 Exported Access Programs . . . . .	25
17.4 Semantics . . . . .	25
17.4.1 State Variables . . . . .	25
17.4.2 Environment Variables . . . . .	25
17.4.3 Assumptions . . . . .	25
17.4.4 Access Routine Semantics . . . . .	25
17.4.5 Local Functions . . . . .	26
<b>18 MIS of Chloride on Deck Calculation Module</b>	<b>27</b>
18.1 Module . . . . .	27
18.2 Uses . . . . .	27
18.3 Syntax . . . . .	27

18.3.1	Exported Constants . . . . .	27
18.3.2	Exported Access Programs . . . . .	27
18.4	Semantics . . . . .	27
18.4.1	State Variables . . . . .	27
18.4.2	Environment Variables . . . . .	27
18.4.3	Assumptions . . . . .	27
18.4.4	Access Routine Semantics . . . . .	27
18.4.5	Local Functions . . . . .	28
<b>19</b>	<b>MIS of Chloride Exposure Database Generation Module</b>	<b>29</b>
19.1	Module . . . . .	29
19.2	Uses . . . . .	29
19.3	Syntax . . . . .	29
19.3.1	Exported Constants . . . . .	29
19.3.2	Exported Access Programs . . . . .	29
19.4	Semantics . . . . .	29
19.4.1	State Variables . . . . .	29
19.4.2	Environment Variables . . . . .	29
19.4.3	Assumptions . . . . .	30
19.4.4	Access Routine Semantics . . . . .	30
19.4.5	Local Functions . . . . .	30



### 3 Introduction

The following document details the Module Interface Specifications for Bridge Corrosion which investigate how climate, traffic might impact corrosion-induced damage for reinforced concrete bridges by influencing the chloride exposure.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at [here](#).

### 4 Notation

The structure of the MIS for modules comes from [HoffmanAndStrooper1995], with the addition that template modules have been adapted from [GhezziEtAl2003]. The mathematical notation comes from Chapter 3 of [HoffmanAndStrooper1995]. For instance, the symbol  $:=$  is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

The following table summarizes the primitive data types used by BC.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
empty	$\emptyset$	when the input is empty or the variable does not exist
GeoDataFrame	GeoDataFrame	pandas dataFrame object with geometry column

The specification of BC 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, BC 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	
	Input Module
	Control Module
	Data Searching Module
	Output Visualization Module
	Data Model Reading Module
	Constant Module
Behaviour-Hiding Module	Deicing Salts Calculation Module
	Melted Water Thickness Calculation Module
	Single Water SAS Calculation Module
	Single Chloride Ions SAS Calculation Module
	All Chloride Ions SAS Calculation Module
	Chloride on Pier Calculation Module
	Chloride on Deck Calculation Module
	Chloride Exposure Database Generation Module
Software Decision Module	File I/O Module
	Plotting Module

Table 1: Module Hierarchy

## 6 MIS of Control Module

This module provides the main program and the GUI of the software.

### 6.1 Module

app

### 6.2 Uses

- Input Module (Section [7](#))
- Data Searching Module (Section [8](#))
- Output Visualization Module (Section [9](#))

### 6.3 Syntax

#### 6.3.1 Exported Constants

None

#### 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

### 6.4 Semantics

#### 6.4.1 State Variables

None

#### 6.4.2 Environment Variables

None

#### 6.4.3 Assumptions

None

#### 6.4.4 Access Routine Semantics

app():

- transition: Control and execute the other modules as follow:
  - Get and verify the input from user. (Section 7)
  - Search the corresponding data in Data Searching Module if the input is valid. (Section 8)
  - Visualize the resulting data by using Output Visualization Module. (Section 9)
- output: out := None
- exception: exc := None

#### 6.4.5 Local Functions

None

## 7 MIS of Input Module

This module get the input from user and verify if it is within the physical and software constraints.

### 7.1 Module

CoordinateChecker

### 7.2 Uses

None

### 7.3 Syntax

#### 7.3.1 Exported Constants

None

#### 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
convertLongitude	long: String	long: $\mathbb{R}$	ValueError
convertLatitude	lat: String	lat: $\mathbb{R}$	ValueError
checkCoordinate	long: $\mathbb{R}$ , lat: $\mathbb{R}$	Boolean	InputOutofOntarioError

### 7.4 Semantics

#### 7.4.1 State Variables

- lon:  $\mathbb{R}$  # longitude get from user.
- lat:  $\mathbb{R}$  # latitude get from user.
- isWithinOntario: Boolean # if the point is within Ontario
- selectedOption: String # the component that user wants to investigate
- rateOption: String # the salt application rate that user wants to investigate

#### 7.4.2 Environment Variables

- Keyboard: takes input from the keyboard by typing.
- File: the geojson file that contains the shape of Ontario.

### 7.4.3 Assumptions

This module use the open source geojson file that contain the Ontario boundary, by drawing polygons with the vertex coordinate. The coordinates for those vertexes are assumed to be reliable.

### 7.4.4 Access Routine Semantics

**convertLongitude(*long*):**

- output:  $\text{out} := \text{long}: \mathbb{R}$
- exception:  $\text{exc} := \text{ValueError: isNaN}(\text{parseFloat}(\text{long}))$

**convertLatitude(*lat*):**

- output:  $\text{out} := \text{lat}: \mathbb{R}$
- exception:  $\text{exc} := \text{ValueError: isNaN}(\text{parseFloat}(\text{lat}))$

**checkCoordinate(*long*, *lat*):**

- output:  $\text{out} := \text{isWithinOntario}: \text{Boolean}$
- exception:  $\text{exc} := \text{InputOutOfOntarioError: } \neg((\text{long}, \text{lat}) \in \text{Ontario})$

### 7.4.5 Local Functions

**isNaN(*e*):**

- output:  $\text{out} := \text{Boolean} \# \text{ check if the input } e \text{ is NaN}$

**parseFloat(*s*):**

- output:  $\text{out} := s: \mathbb{R} \# \text{ convert the input } s \text{ from type string to float, if it is not number and can not be converted, it would return NaN}$

**InputOutOfOntarioError:**

- output:  $\text{out} := \text{Exception} \# \text{ raise this exception if the input is out of Ontario}$

## 8 MIS of Data Searching Module

This module finds the data needed in the database.

### 8.1 Module

DataSearching

### 8.2 Uses

Input Module (Section 7), Chloride Exposure Database Generation Module (Section 19)

### 8.3 Syntax

#### 8.3.1 Exported Constants

None

#### 8.3.2 Exported Access Programs

Name	In	Out	Exceptions
loadData	dataOption: String, rateOption: String	filePath: String	-
findClosest	long: $\mathbb{R}$ , lat: $\mathbb{R}$	index: $\mathbb{N}$	-
searchData	long: $\mathbb{R}$ , lat: $\mathbb{R}$	data: sequence of $\mathbb{R}$	-

### 8.4 Semantics

#### 8.4.1 State Variables

- data: sequence of  $\mathbb{R}$  # the sequence of predicted chloride exposure data that the user want

#### 8.4.2 Environment Variables

- File: the database file that contain the yearly chloride exposure data within Ontario.

#### 8.4.3 Assumptions

All locations within Ontario, if it is not on water, must contain valid data.

#### 8.4.4 Access Routine Semantics

**loadData**(*dataOption*, *rateOption*):

- output:  $\text{out} := \text{filePath} \# \text{filePath}$  for the data that user want
- exception:  $\text{exc} := \text{FileNotFoundError} = \nexists \text{filename}$

**findClosest**(*long*, *lat*): If the input coordinate is not the exact one in database, find the grid that it belongs to and return the index of center coordinate.

- output:  $\text{out} := \text{index}: \mathbb{N}$

**searchData**(*long*, *lat*):

- output:  $\text{out} := \text{data} = \text{search}(\text{findClosest}(\text{long}, \text{lat})) \#$  sequence of  $\mathbb{R}$ , the chloride exposure result
- exception:  $\text{exc} := \text{None}$

#### 8.4.5 Local Functions

None



## 9 MIS of Output Visualization Module

This module provides the visualization of the resulting chloride exposure trend.

### 9.1 Module

dataVisualization

### 9.2 Uses

Data Searching Module (Section [8](#))

### 9.3 Syntax

#### 9.3.1 Exported Constants

None

#### 9.3.2 Exported Access Programs

Name	In	Out	Exceptions
generateChartData	data: sequence of $\mathbb{R}$	-	-
downloadData	-	-	-
downloadJPG	-	-	-

### 9.4 Semantics

#### 9.4.1 State Variables

None

#### 9.4.2 Environment Variables

- Screen: The graphs are displayed on the screen.
- File: The files that could be downloaded to the local machine

#### 9.4.3 Assumptions

None

#### 9.4.4 Access Routine Semantics

**generateChartData(*data*):**

- transition: display the graphs using the result data from Data Searching Module.
- output: `out := None`
- exception: `exc := None`

**downloadData():**

- transition: download the data to the local machine
- output: `out := None`
- exception: `exc := None`

**downloadJPG():**

- transition: download the visualization to the local machine
- output: `out := None`
- exception: `exc := None`

#### 9.4.5 Local Functions

None

## 10 MIS of Data Model Reading Module

This module loads the climate and traffic data from the external file and store it in the data format that could be used for calculation.

### 10.1 Module

calculation\_loadT

### 10.2 Uses

None

### 10.3 Syntax

#### 10.3.1 Exported Constants

None

#### 10.3.2 Exported Access Programs

Name	In	Out	Exceptions
new calculation_load	filename: String	calculation_loadT	FileNotFoundError

### 10.4 Semantics

A data structure designed to store the data from climate and traffic model.

#### 10.4.1 State Variables

- long: sequence of  $\mathbb{R}$  # longitude get from climate and traffic model
- lat: sequence of  $\mathbb{R}$  # latitude get from climate and traffic model
- AADT: sequence of  $\mathbb{R}$  # annual average daily traffic per lane
- AADTT: sequence of  $\mathbb{R}$  # annual average daily truck traffic per lane
- t1: sequence of  $\mathbb{R}$  # number of days with snowfall
- h\_total: sequence of  $\mathbb{R}$  # the total snowfall during a winter season
- t2: sequence of  $\mathbb{R}$  # number of days with snow melting

### 10.4.2 Environment Variables

File: the file with all climate model data and traffic model data

### 10.4.3 Assumptions

None

### 10.4.4 Access Routine Semantics

**calculation\_load:**

- transition: Read and store the data from the climate model and traffic model file
- output:  $\text{out} := \text{self}$
- exception:  $\text{exc} := \text{FileNotFoundError} = \# \text{ filename}$

### 10.4.5 Local Functions

None

## 11 MIS of Constant Module

This module stores the constants used for calculation.

### 11.1 Module

constantT

### 11.2 Uses

None

### 11.3 Syntax

#### 11.3.1 Exported Constants

Name	Value	Note
salt_application_rate	0.07	salt application rate
W_lane	3.75	lane width
V_speed	100	heavy vehicle speed
b	0.56	tire width
K	0.75	ratio of the tire width that is not a groove to the tire width
h_film	0.0001	depth of the water film picked up in each rotation
water_density	997	density of water
V	62.1371	truck speed
chloride_ratio	0.61	molar mass ratio of chloride ions over deicing salts
d	3.5	distance between road edge and nearby bridge structure
ldv_ratio	6	ratio of chloride ions sprayed and splashed by trucks over light-duty vehicles

#### 11.3.2 Exported Access Programs

None

### 11.4 Semantics

#### 11.4.1 State Variables

None

#### 11.4.2 Environment Variables

None

### **11.4.3 Assumptions**

None

### **11.4.4 Access Routine Semantics**

None

### **11.4.5 Local Functions**

None

## 12 MIS of Deicing Salts Calculation Module

This module provides the calculation for the quantity of deicing salts applied per day with snowfall

### 12.1 Module

deicing\_salts\_cal

### 12.2 Uses

Constant Module (Section 11), Data Model Reading Module (Section 10)

### 12.3 Syntax

#### 12.3.1 Exported Constants

None

#### 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
deicing_salts_cal	h_total: sequence of $\mathbb{R}$ , t1: sequence of $\mathbb{R}$	M_app: sequence of $\mathbb{R}$	-

### 12.4 Semantics

#### 12.4.1 State Variables

None

#### 12.4.2 Environment Variables

None

#### 12.4.3 Assumptions

None

#### 12.4.4 Access Routine Semantics

deicing\_salts\_cal( $h\_total, t1$ ):

- transition: None

- output:  $out := \frac{salt\_application\_rate * h\_total}{(W\_lane * t1)}$ , where `salt_application_rate` and `W_lane` are constant value get from Constant Module, `h_total` and `t1` are read from Data Model Reading Module
- exception: `exc := None`

#### 12.4.5 Local Functions

None



## 13 MIS of Melted Water Thickness Module

This module provides the calculation for the daily water film thickness on the road

### 13.1 Module

melted\_water\_thickness

### 13.2 Uses

Data Model Reading Module (Section 10)

### 13.3 Syntax

#### 13.3.1 Exported Constants

None

#### 13.3.2 Exported Access Programs

Name	In	Out	Exceptions
melted_water_thickness	h_total: sequence of $\mathbb{R}$ , t2: sequence of $\mathbb{R}$	h_app: sequence of $\mathbb{R}$	-

### 13.4 Semantics

#### 13.4.1 State Variables

None

#### 13.4.2 Environment Variables

None

#### 13.4.3 Assumptions

None

#### 13.4.4 Access Routine Semantics

**melted\_water\_thickness**( $h\_total, t2$ ):

- transition: None
- output:  $out := \frac{h\_total}{t2}$ , where h\_total and t1 are read from Data Model Reading Module
- exception:  $exc := \text{None}$

### 13.4.5 Local Functions

None

## 14 MIS of Single Water SAS Calculation Module

This module determine water sprayed and splashed by one truck using a (CFD)-based analytical model, taking into account of the four primary mechanisms of vehicle spray and splash: capillary adhesion, tread pickup, bow wave, and side wave.

### 14.1 Module

`single_water_SAS_cal`

### 14.2 Uses

Constant Module (Section 11), Melted Water Thickness Module (Section 13)

### 14.3 Syntax

#### 14.3.1 Exported Constants

None

#### 14.3.2 Exported Access Programs

Name	In	Out	Exceptions
<code>single_water_SAS_cal</code>	<code>h_app</code> : sequence of $\mathbb{R}$	<code>SD_total</code> : sequence of $\mathbb{R}$	-

### 14.4 Semantics

#### 14.4.1 State Variables

None

#### 14.4.2 Environment Variables

None

#### 14.4.3 Assumptions

None

#### 14.4.4 Access Routine Semantics

`single_water_SAS_cal(h_app)`:

- transition: None

- output:  $out := SD_{CA} + SD_{TP} + SD_{BW} + SD_{SW}$  # the mass of water per unit air volume kicked up by each passing truck is the sum of the four mechanisms, calculated by the local functions below.
- exception:  $exc := \text{None}$

#### 14.4.5 Local Functions

$V_{speed}, b, K, h_{film}, \rho_{water}, V'$  are constants read from Constant Module.

**mass\_flow\_rate**( $h_{app}$ ):

- transition:  $\text{None}$
- output:  $out :=$

$$\begin{cases} MR_{CA} = V_{speed} \times b \times K \times h_{film} \times \rho_{water} & \text{for } CA \\ MR_{TP} = V_{speed} \times b \times (1 - K) \times h_{app} \times \rho_{water} & \text{for } TP \\ MR_{BW} = MR_{SW} = 0.5 \times V_{speed} \times b \times \\ (h_{app} - K \times h_{film} - (1 - K) \times h_{app}) \times \rho_{water} & \text{for } BW \text{ and } SW \end{cases}$$

- exception:  $exc := \text{None}$

**spray\_density**( $MR_{CA}, MR_{TP}, MR_{BW}, MR_{SW}$ ):

- transition:  $\text{None}$
- output:  $out :=$

$$\begin{cases} SD_{CA} = (-2.69 \times 10^{-5} \times V' + 2.43 \times 10^{-3}) \times MR_{CA} & \text{for } CA \\ SD_{TP} = (1.16 \times 10^{-5} \times V' - 5.25 \times 10^{-5}) \times MR_{TP} & \text{for } TP \\ SD_{BW} = (2.67 \times 10^{-5} \times V' - 4.71 \times 10^{-4}) \times MR_{BW} & \text{for } BW \\ SD_{SW} = (1.65 \times 10^{-5} \times V' - 3.99 \times 10^{-4}) \times MR_{SW} & \text{for } SW \end{cases}$$

- exception:  $exc := \text{None}$

#### 14.4.6 Local Functions

$\text{None}$

## 15 MIS of Single Chloride Ions SAS Calculation Module

This module determines the chloride ions sprayed and splashed by one truck.

### 15.1 Module

single\_CLSAS\_cal

### 15.2 Uses

Deicing Salts Calculation Module (Section [12](#)), Single Water SAS Calculation Module (Section [14](#))

### 15.3 Syntax

#### 15.3.1 Exported Constants

None

#### 15.3.2 Exported Access Programs

Name	In	Out	Exceptions
single_CLSAS_cal	M_app: sequence of $\mathbb{R}$ , h_app: sequence of $\mathbb{R}$ , SD_total: sequence of $\mathbb{R}$	SD_totalCl: sequence of $\mathbb{R}$	-

### 15.4 Semantics

#### 15.4.1 State Variables

None

#### 15.4.2 Environment Variables

None

#### 15.4.3 Assumptions

None

#### 15.4.4 Access Routine Semantics

**single\_Cl\_SAS\_cal**( $M\_app, h\_app, SD\_total$ ):

- transition: None
- output:  $out := SD\_total * salt\_water\_ratio(M\_app, h\_app) * chloride\_ratio$ , where  $chloride\_ratio$  is a constant read from Constant Module.
- exception:  $exc := \text{None}$

#### 15.4.5 Local Functions

**salt\_water\_ratio**( $M\_app, h\_app$ ):

- transition: None
- output:  $out := \frac{M\_app}{h\_app * water\_density}$  where  $water\_density$  is a constant read from Constant Module.
- exception:  $exc := \text{None}$

## 16 MIS of All Chloride Ions SAS Calculation Module

This module determines chloride ions sprayed and splashed by all vehicles in one winter season

### 16.1 Module

all\_Cl.SAS\_cal

### 16.2 Uses

Constant Module (Section 11), Data Model Reading Module (Section 10), Single Water SAS Calculation Module (Section 14)

### 16.3 Syntax

#### 16.3.1 Exported Constants

None

#### 16.3.2 Exported Access Programs

Name	In	Out	Exceptions
all_Cl.SAS_cal	SD_totalCl: sequence of $\mathbb{R}$ , t2: sequence of $\mathbb{R}$ , AADT: sequence of $\mathbb{R}$ , AADTT: sequence of $\mathbb{R}$	C_s_air: sequence of $\mathbb{R}$	-
updateAADT	AADT: sequence of $\mathbb{R}$	AADT: sequence of $\mathbb{R}$	-
updateAADTT	AADTT: sequence of $\mathbb{R}$	AADTT: sequence of $\mathbb{R}$	-

### 16.4 Semantics

#### 16.4.1 State Variables

None

#### 16.4.2 Environment Variables

None

#### 16.4.3 Assumptions

The AADT and AADTT are assumed to have 2% increase rate every year

#### 16.4.4 Access Routine Semantics

**all\_Cl\_SAS\_cal**( $M\_app, h\_app, SD\_total, t2, AADT, AADTT$ ):

- transition: None
- output:  $out := (\frac{SD\_totalCl}{ldv\_ratio} * (updateAADT(AADT) - updateAADTT(AADTT)) + SD\_totalCl * AADTT) * t2$ , where *ldv\_ratio* is a constant read from Constant Module.
- exception:  $exc := \text{None}$

**updateAADT**( $AADT$ ):

- transition: None
- output:  $out := AADT$  # calculate the AADT for future year, assuming a 2% annual increase rate
- exception:  $exc := \text{None}$

**updateAADTT**( $AADTT$ ):

- transition: None
- output:  $out := AADTT$  # calculate the AADTT for future year, assuming a 2% annual increase rate
- exception:  $exc := \text{None}$

#### 16.4.5 Local Functions

None



## 17 MIS of Chloride on Pier Calculation Module

This module determine the deposition of chloride ions on the pier of the bridge substructure

### 17.1 Module

chloride\_on\_pier

### 17.2 Uses

Constant Module (Section 11), All Chloride Ions SAS Calculation Module (Section 16)

### 17.3 Syntax

#### 17.3.1 Exported Constants

None

#### 17.3.2 Exported Access Programs

Name	In	Out	Exceptions
chloride_on_pier	C_s_air: sequence of $\mathbb{R}$	results: sequence of $\mathbb{R}$	-

### 17.4 Semantics

#### 17.4.1 State Variables

None

#### 17.4.2 Environment Variables

None

#### 17.4.3 Assumptions

None

#### 17.4.4 Access Routine Semantics

**chloride\_on\_pier**(*C\_s\_air*):

- transition: None
- output:  $\text{out} := C\_s\_air * 0.015 * e^{-0.05*d} + C\_s\_air * 0.985 * e^{-0.5*d}$ , where d is a constant read from Constant Module, 0.015 and 0.985 being the coefficient of the formula.
- exception:  $\text{exc} := \text{None}$

### 17.4.5 Local Functions

None

## 18 MIS of Chloride on Deck Calculation Module

This module determine the deposition of chloride ions on the deck of the bridge substructure

### 18.1 Module

chloride\_on\_deck

### 18.2 Uses

Data Model Reading Module (Section 10)

### 18.3 Syntax

#### 18.3.1 Exported Constants

None

#### 18.3.2 Exported Access Programs

Name	In	Out	Exceptions
chloride_on_deck	h_total: sequence of $\mathbb{R}$ , AADT: sequence of $\mathbb{R}$	results: sequence of $\mathbb{R}$	-

### 18.4 Semantics

#### 18.4.1 State Variables

None

#### 18.4.2 Environment Variables

None

#### 18.4.3 Assumptions

None

#### 18.4.4 Access Routine Semantics

chloride\_on\_deck(h\_total, AADT):

- transition: None
- output:  $out := 0.11 * h\_total - 0.000189 * AADT + 3.349$ . This is a linear regression model.

- exception: `exc := None`

#### **18.4.5 Local Functions**

None

## 19 MIS of Chloride Exposure Database Generation Module

This module performs the calculation process to generate the database, it is related to the R2, R3 in SRS.

### 19.1 Module

calculate

### 19.2 Uses

Data Model Reading Module (Section 10), Constant Module (Section 11), Deicing Salts Calculation Module (Section 12), Melted Water Thickness Module (Section 13), Single Water SAS Calculation Module (Section 14), Single Chloride Ions SAS Calculation Module (Section 15), All Chloride Ions SAS Calculation Module (Section 16), Chloride on Pier Calculation Module (Section 17), Chloride on Deck Calculation Module (Section 18)

### 19.3 Syntax

#### 19.3.1 Exported Constants

None

#### 19.3.2 Exported Access Programs

Name	In	Out	Exceptions
calculate	AADT: sequence of $\mathbb{R}$ , AADTT: sequence of $\mathbb{R}$ , t1: sequence of $\mathbb{R}$ , h <sub>total</sub> : sequence of $\mathbb{R}$ , t2: se- quence of $\mathbb{R}$	result: sequence of $\mathbb{R}$	DataMissingError, DataInvalidError
savefile	long: sequence of $\mathbb{R}$ , lat: sequence of $\mathbb{R}$ , results: sequence of $\mathbb{R}$	file: String	-

### 19.4 Semantics

#### 19.4.1 State Variables

None

#### 19.4.2 Environment Variables

File: the result of calculation will be stored in an output csv file.

### 19.4.3 Assumptions

The map of Ontario is divided into multiple 25km \* 25km grid (as mentioned in SRS) and the coordinates are the center of those grids. The locations inside each grid are consider to have same chloride exposure rate.

### 19.4.4 Access Routine Semantics

**calculate**( $AADT, AADTT, t1, h_{total}, t2$ ):

- transition: use all the formulas from calculate\_step1 to calculate\_step6, conclude the final result
- output:  $out := result \# \text{Sequence of } \mathbb{R}$
- exception:  $exc :=$

Expression	Exception
$\exists e \in [AADT, AADTT, h_{total}, t1, t2], e = \emptyset$	DataMissingError
$(\exists i \in [0.. AADT  - 1], AADTT[i] > AADT[i]) \vee (\neg(t1, t2 \in (0, 365)))$	DataInvalidError

**savefile**( $long, lat, results$ ):

- transition: Save the longitude, latitude and the corresponding results for each grid to a csv file, which is the prediction of chloride exposure rate. The file has a row label as coordinate and a column label as year.
- output:  $out := \text{file}$
- exception:  $exc := \text{None}$

### 19.4.5 Local Functions

None