Module Interface Specification for Attitude Check

Adrian Sochaniwsky

April 4, 2024

1 Revision History

Date	Version	Notes
March 17, 2024	1.0	Initial draft
April 4, 2024	2.0	Update design

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/adrian-soch/attitude_check/blob/main/docs/SRS/SRS.pdf.

Contents

3 Introduction

The following document details the Module Interface Specifications for Attitude Check.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/adrian-soch/attitude_check.

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 Attitude Check.

Data Type	Notation	Description
real	$\mathbb{R}$	any number in (-∞, ∞)
boolean	$\mathbb{B}$	value in $[\text{false} = 0, \text{true} = 1]$
matrix	$\mathbb{R}^{m \times n}$	matrix of any number in (-∞, ∞)
vector	$\mathbb{R}^{m}$	column vector of any number in (-∞, ∞)
quaternion	$\mathbb{H}^4$	a quaternion $\in \mathbb{R}^4$, see SRS for details

Attitude Check 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**
Behaviour-Hiding Module	Attitude Check Module Initializers Module
Software Decision Module	Math Module Matrix Math Module Quaternion Module Error Handling Module Utilities Module

Table 1: Module Hierarchy

6 MIS of Attitude Check Module

6.1 Module

attitude_check

6.2 Uses

Initializers Module (Sec 7)
Error Handling Module (Sec 8)
Quaternion Module (Sec 10)
Matrix Math Module (Sec 11)
Math Module (Sec 12)

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
AttitudeCheck	-	-	-
AttitudeCheck	$\mathbb{R}$, $\mathbb{R}$	-	ValueError
AttitudeCheck	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	-	ValueError
update	$\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}$	$\mathbb{R}^4$	-
update	$\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}$	$\mathbb{R}^4$	-
get_mag_norm	$\mathbb{H}^4$, $\mathbb{R}^3$	$\mathbb{R}^3$	-
set_gain	$\mathbb{R}$	-	ValueError
get_gain	-	$\mathbb{R}$	-
get_initial_orientation	$\mathbb{R}^3, \mathbb{R}^3$	-	-
get_initial_orientation	$\mathbb{R}^3$	-	-

6.4 Semantics

6.4.1 State Variables

$$\text{imuGain} : \mathbb{R}$$

$$\text{margGain} : \mathbb{R}$$

$$q : \mathbb{H}^4$$

6.4.2 Access Routine Semantics

AttitudeCheck():

- transition: imuGain:=0.033, margGain:=0.041, q:=[1, 0, 0, 0]

AttitudeCheck(g1, g2):

- transition: imuGain:=g1, margGain:=g2, q:=[1, 0, 0, 0]
- exception: ValueError if 0 > g1, g2 > 1

AttitudeCheck(g1, g2, w, x, y, z):

- transition: imuGain:=g1, margGain:=g2, q:=[w, x, y, z]
- exception: ValueError if $(0 >$ g1, g2 $> 1) \vee (|q| \neq 1)$

$\mathrm{update}(\mathrm{acc}, \mathrm{gyr}, \mathrm{mag}, \mathrm{dt})$:

- transition: $q =:$ out
- output: $out := q + \left(\dot{q}_\omega - \mathrm{margGain}\frac{\mathrm{transpose}(J(q, {}^E\mathbf{b}))f_{g,b}(q, \mathrm{acc}, {}^E\mathbf{b}, \mathrm{mag})}{\mathrm{norm}(\mathrm{transpose}(J(q, {}^E\mathbf{b}))f_{g,b}(q, \mathrm{acc}, {}^E\mathbf{b}, \mathrm{mag}))}\right)\mathrm{dt}$ where

$$
\begin{aligned}
\dot{q}_\omega =& \frac{1}{2}\mathrm{quat\_prod}(q, \mathrm{Quaternion}(0, \mathrm{gyr})) \\
{}^E\mathbf{b} =& \mathrm{get\_mag\_norm}(q, [m_x, m_y, m_z]) \\
[a_x, a_y, a_z] =& \frac{\mathrm{acc}}{|\mathrm{acc}|} \\
[m_x, m_y, m_z] =& \frac{\mathrm{mag}}{|\mathrm{mag}|} \\
f_{g,b} =& \begin{bmatrix} 2(q_x q_z - q_w q_y) - a_x \\ 2(q_w q_x + q_y q_z) - a_y \\ 2(\frac{1}{2} - q_x^2 - q_y^2) - a_z \\ 2b_x(\frac{1}{2} - q_y^2 - q_z^2) + 2b_z(q_x q_z - q_w q_y) - m_x \\ 2b_x(q_x q_y - q_w q_z) + 2b_z(q_w q_x + q_y q_z) - m_y \\ 2b_x(q_w q_y + q_x q_z) + 2b_z(\frac{1}{2} - q_x^2 - q_y^2) - m_z \end{bmatrix} \\
J =& \begin{bmatrix} -2q_y & 2q_z & -2q_w & 2q_x \\ 2q_x & 2q_w & 2q_z & 2q_y \\ 0 & -4q_x & -4q_y & 0 \\ -2b_z q_y & 2b_z q_z & -4b_x q_y - 2b_z q_w & -4b_x q_z + 2b_z q_x \\ -2b_x q_z + 2b_z q_x & 2b_x q_y + 2b_z q_w & 2b_x q_x + 2b_z q_z & -2b_x q_w + 2b_z q_y \\ 2b_x q_y & 2b_x q_z - 4b_z q_x & 2b_x q_w - 4b_z q_y & 2b_x q_x \end{bmatrix}
\end{aligned}
$$

$\mathrm{update}(\mathrm{acc}, \mathrm{gyr}, \mathrm{dt})$:

- transition: $q =:$ out

- output: $\text{out} := q + \left(\dot{q}_\omega - \text{imuGain}\frac{\text{transpose}(J(q))f_g(q,\text{acc})}{\text{norm}(\text{transpose}(J(q))f_g(q,\text{acc}))}\right)\text{dt}$ where

$$\begin{aligned}
\dot{q}_\omega =& \frac{1}{2}\text{quat_prod}(q, \text{Quaternion}(0, \text{gyr})) \\
[a_x, a_y, a_z] =& \frac{\text{acc}}{|\text{acc}|} \\
J =& \begin{bmatrix} -2q_y & 2q_z & -2q_w & 2q_x \\ 2q_x & 2q_w & 2q_z & 2q_y \\ 0 & -4q_x & -4q_y & 0 \end{bmatrix} \\
f_g =& \begin{bmatrix} 2(q_xq_z - q_wq_y) - a_x \\ 2(q_wq_x + q_yq_z) - a_y \\ 2(\frac{1}{2} - q_x^2 - q_y^2) - a_z \end{bmatrix}
\end{aligned}$$

get_mag_norm(q, mag):

- output: ${}^E\mathbf{b} := [b_x, 0, b_z]$ where

$$\begin{aligned}
\text{m_quat} =& \text{quat_prod}(\text{Quaternion}(0, m_x, m_y, m_z), \text{conjugate}(q)) \\
b_x =& \text{norm}([\text{m_quat}_x, \text{m_quat}_y]) \\
b_z =& \text{m_quat}_z
\end{aligned}$$

get_gain():

- output: out:=[imuGain, margGain]

set_gain(g1, g2):

- transition: imuGain:=g1, margGain:=g2
- exception: ValueError if 0 > gain > 1

get_initial_orientation(a, m):

- transition: q := mag_to_quat(m, a)

get_initial_orientation(a):

- transition: q := acc_to_quat(a)

7 MIS of Initializers Module

7.1 Module

Initializers

7.2 Uses

Quaternion Module (Sec 10)
Matrix Math Module (Sec 11)
Math Module (Sec 12)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
acc_to_quat	$\mathbb{R}^3$	$\mathbb{H}^4$	-
mag_to_quat	$\mathbb{R}^3, \mathbb{R}^3$	$\mathbb{H}^4$	-

7.4 Semantics

7.4.1 Access Routine Semantics

acc_to_quat($\mathbf{a}$):

- output: out := euler_to_quat(θ, ϕ, ψ) given

$$\theta = \text{atan2}(a_y, a_z)$$
$$\phi = \text{atan2}(-a_x, \sqrt{a_y^2 + a_z^2})$$
$$\psi = 0$$

where $[a_x, a_y, a_z] = \dfrac{\mathbf{a}}{|\mathbf{a}|}$

mag_to_quat($\mathbf{m}, \mathbf{a}$):

- output: out := euler_to_quat(θ, ϕ, ψ) given

$$\theta = \text{atan2}(a_y, a_z)$$
$$\phi = \text{atan2}(-a_x, \sqrt{a_y^2 + a_z^2})$$
$$\psi = \text{atan2}(m_z \sin\phi - m_y \cos\phi, m_x \cos\theta + \sin\theta(m_y \sin\phi + m_z \cos\phi))$$

where $[a_x, a_y, a_z] = \dfrac{\mathbf{a}}{|\mathbf{a}|}$ and $[m_x, m_y, m_z] = \dfrac{\mathbf{m}}{|\mathbf{m}|}$

8 MIS of Error Handler Module

8.1 Module

error_handler

8.2 Uses

none

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
error_handler	string	-	string

8.4 Semantics

8.4.1 Access Routine Semantics

error_handler(exception):

- exception: Exception := exception

9 MIS of Utilities Module

9.1 Module

utilities

9.2 Uses

Math Module (Sec 12)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
euler_to_quat	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	$\mathbb{H}^4$	-
rotm_to_quat	$\mathbb{R}^{3\times 3}$	$\mathbb{H}^4$	-

9.4 Semantics

9.4.1 Access Routine Semantics

euler_to_quat(roll, pitch, yaw):

- output: See https://www.euclideanspace.com/maths/geometry/rotations/conversions/eulerToQuaternion/index.htm

rotm_to_quat($\mathbf{R}$):

- output: See https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/

10 MIS of Quaternion Module

10.1 Module

quaternion

10.2 Uses

Matrix Math Module (Sec 11)
Math Module (Sec 12)

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
Quaternion	-	-	-
Quaternion	$\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}$	-	ValueError
quat_prod	$\mathbb{H}^4, \mathbb{H}^4$	$\mathbb{H}^4$	-
normalize	-	-	-
conjugate	-	$\mathbb{H}^4$	-
to_array	-	$\mathbb{R}^4$	-
get_w	-	$\mathbb{R}$	-
get_x	-	$\mathbb{R}$	-
get_y	-	$\mathbb{R}$	-
get_z	-	$\mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

quat : $\mathbb{H}^4$

10.4.2 Access Routine Semantics

Quaternion():

- transition: quat $:= [q_w = 1, q_x = 0, q_y = 0, q_z = 0]$

Quaternion(w, x, y, z):

- transition: quat $:= [q_w = w, q_x = x, q_y = y, q_z = z]$
- exception: ValueError when $|\text{quat}| = 0$

quat_prod(p, q):

- output:

$$q_{\text{out}} := \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}$$

conjugate():

- output: $q_{\text{conj}} := [q_w, -q_x, -q_y, -q_z]$

normalize():

- transition: quat := $\left[\frac{\text{quat}_w}{d}, \frac{\text{quat}_x}{d}, \frac{\text{quat}_y}{d}, \frac{\text{quat}_z}{d}\right]$ where $d = \sqrt{\text{quat}_w^2 + \text{quat}_x^2 + \text{quat}_y^2 + \text{quat}_z^2}$

to_array():

- output: $\mathbf{R} = [q_w, q_x, q_y, q_z]$

get_w():

- output: y:=q_w

get_x():

- output: y:=q_x

get_y():

- output: y:=q_y

get_z():

- output: y:=q_z

11 MIS of Matrix Math Module

11.1 Module

matrix

11.2 Uses

Math Module (Sec 12)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
$*$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times m}$	$\mathbb{R}^{n\times n}$	-
$*$	$\mathbb{R}^{m\times n} \times \mathbb{R}$	$\mathbb{R}^{m\times n}$	-
$+$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$	$\mathbb{R}^{m\times n}$	-
-	$\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$	$\mathbb{R}^{m\times n}$	-
transpose	$\mathbb{R}^{m\times n}$	$\mathbb{R}^{n\times m}$	-
norm	$\mathbb{R}^{n}$	$\mathbb{R}^{n}$	-

11.4 Semantics

11.4.1 Access Routine Semantics

$\mathbf{A} * \mathbf{B}$:

- output: $m := \mathbf{C}$

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $\mathbf{B} = [b_{i,j}]_{n\times m}$. Then $\mathbf{C} = \mathbf{A} * \mathbf{B}$ with $c_{i,j} = a_{i,0}b_{0,j} + a_{i,1}b_{1,j} ... a_{i,n}b_{n,j}$.

$\mathbf{A} * k$:

- output: $m := \mathbf{C}$

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $k = \mathbb{R}$. Then $\mathbf{C} = \mathbf{A} * k$ with $c_{i,j} = ka_{i,j}$.

$\mathbf{A} + \mathbf{B}$:

- output: $m := \mathbf{A} + \mathbf{B}$

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $\mathbf{B} = [b_{i,j}]_{m\times n}$. Then $\mathbf{A} + \mathbf{B} = [a_{i,j} + b_{i,j}]_{m\times n}$.

$\mathbf{A} - \mathbf{B}$:

- output: $m := \mathbf{A} - \mathbf{B}$

Let $\mathbf{A} = [a_{i,j}]_{m \times n}$ and $\mathbf{B} = [b_{i,j}]_{m \times n}$. Then $\mathbf{A} - \mathbf{B} = [a_{i,j} - b_{i,j}]_{m \times n}$.

transpose($\mathbf{A}$):

- output: $\mathbf{A}^{\mathbf{T}}$

$[\mathbf{A}^T]_{i,j} = [\mathbf{A}]_{j,i}$

norm($\mathbf{A}$):

- output: $y := \dfrac{\mathbf{A}}{|\mathbf{A}|}$

12 MIS of Math Module

12.1 Module

math

12.2 Uses

None

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
sin	$\mathbb{R}$	$\mathbb{R}$	-
cos	$\mathbb{R}$	$\mathbb{R}$	-
asin	$\mathbb{R}$	$\mathbb{R}$	-
atan2	$\mathbb{R}$, $\mathbb{R}$	$\mathbb{R}$	-

12.4 Semantics

12.4.1 Access Routine Semantics

sin(x):

- output: $y := \sin(x)$

cos(x):

- output: $y := \cos(x)$

asin(x):

- output: $y := \sin(x)$

atan2(y, x):

- output: See https://en.wikipedia.org/wiki/Atan2#Definition_and_computation
- exception: ValueError if $x == 0, y == 0$

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.