

Module Interface Specification for STEM Moiré GPA

Alexandre Pofelski
macid: pofelska
github: slimpotatoes

November 27, 2017

1 Revision History

Date	Version	Notes
27/11/2017	1.0	MIS First draft

2 Symbols, Abbreviations and Acronyms

The same Symbols, Abbreviations and Acronyms as in the SRS, the TestPlan and the MG (available in [STEM Moiré GPA](#) repository) are used in the Module Interface Specifications document.

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of STEM Moiré GPA Control Module (M 2)	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Access Routine Semantics	4
7	MIS of STEM Moiré GPA GUI Module (M 3)	4
7.1	Module	4
7.2	Uses	4
7.3	Syntax	5
7.3.1	Exported Access Programs	5
7.4	Semantics	5
7.4.1	State Variables	5
7.4.2	Environment Variables	5
7.4.3	Access Routine Semantics	6
8	MIS of Input Module (M 4)	8
8.1	Module	8
8.2	Uses	8
8.3	Syntax	8
8.3.1	Exported Access Programs	8
8.4	Semantics	9
8.4.1	State Variables	9
8.4.2	Access Routine Semantics	9
9	MIS of SMH Simulation (M 5)	9
9.1	Module	9
9.2	Uses	10
9.3	Syntax	10

9.3.1	Exported Access Programs	10
9.4	Semantics	10
9.4.1	State Variables	10
9.4.2	Access Routine Semantics	10
10	MIS of GPA Module (M 6)	10
10.1	Module	10
10.2	Uses	11
10.3	Syntax	11
10.3.1	Exported Access Programs	11
10.4	Semantics	11
10.4.1	State Variables	11
10.4.2	Access Routine Semantics	11
11	MIS of Mask Module (M 7)	12
11.1	Module	12
11.2	Uses	12
11.3	Syntax	12
11.3.1	Exported Access Programs	12
11.4	Semantics	12
11.4.1	State Variables	12
11.4.2	Access Routine Semantics	12
12	MIS of Unstrained region (M 8)	13
12.1	Module	13
12.2	Uses	13
12.3	Syntax	13
12.3.1	Exported Access Programs	13
12.4	Semantics	13
12.4.1	State Variables	13
12.4.2	Access Routine Semantics	13
13	MIS of Conversion Module (M 9)	13
13.1	Module	13
13.2	Uses	14
13.3	Syntax	14
13.3.1	Exported Access Programs	14
13.4	Semantics	14
13.4.1	State Variables	14
13.4.2	Access Routine Semantics	14

14 MIS of 2D Strain Tensor Module (M 10)	14
14.1 Module	14
14.2 Uses	14
14.3 Syntax	15
14.3.1 Exported Access Programs	15
14.4 Semantics	15
14.4.1 State Variables	15
14.4.2 Access Routine Semantics	15
15 MIS of Fourier Transform Module (M 11)	15
15.1 Module	16
15.2 Uses	16
15.3 Syntax	16
15.3.1 Exported Access Programs	16
15.4 Semantics	16
15.4.1 State Variables	16
15.4.2 Access Routine Semantics	16
16 MIS of Gradient Module (M 12)	17
16.1 Module	17
16.2 Uses	17
16.3 Syntax	17
16.3.1 Exported Access Programs	17
16.4 Semantics	17
16.4.1 State Variables	17
16.4.2 Access Routine Semantics	17
17 MIS of Least Square Fit Method Module (M 13)	17
17.1 Module	17
17.2 Uses	18
17.3 Syntax	18
17.3.1 Exported Access Programs	18
17.4 Semantics	18
17.4.1 State Variables	18
17.4.2 Access Routine Semantics	18
18 MIS of Phase Operation Module (M 14)	18
18.1 Module	18
18.2 Uses	18
18.3 Syntax	19
18.3.1 Exported Access Programs	19
18.4 Semantics	19
18.4.1 State Variables	19

18.4.2 Access Routine Semantics	19
19 MIS of Data Structure Module (M 15)	19
19.1 Module	19
19.2 Uses	20
19.3 Syntax	20
19.3.1 Exported Access Programs	20
19.4 Semantics	20
19.4.1 State Variables	20
19.4.2 Access Routine Semantics	21
20 MIS of Generic GUI/Plot Module (M 16)	21
20.1 Module	21
20.2 Uses	21
20.3 Syntax	21
20.3.1 Exported Access Programs	21
20.4 Semantics	22
20.4.1 State Variables	22
20.4.2 Access Routine Semantics	22
21 Appendix	24

3 Introduction

The following document details the Module Interface Specifications for STEM Moiré GPA. The full documentation and implementation can be found in [STEM Moiré GPA](#) repository.

4 Notation

The structure of the MIS for modules comes from [1], with the addition that template modules have been adapted from [2]. The following table summarizes the primitive data types used by STEM Moiré GPA.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	an integer number
natural	\mathbb{N}	a natural number
real	\mathbb{R}	a real number
image	\mathbb{I}	blabla

The specification of STEM Moiré GPA 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, STEM Moiré GPA 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	
	STEM Moiré GPA Control (M 2, section 6)
	STEM Moiré GPA GUI (M 3, section 7)
	Input (M 4, section 8)
	SMH simulation (M 5, section 9)
Behaviour-Hiding Module	GPA (M 6, section 10)
	Mask (M 7, section 11)
	Unstrained region (M 8, section 12)
	Conversion (M 9, section 13)
	2D strain tensor (M 10, section 14)
	Fourier Transform (M 11, section 15)
	Gradient (M 12, section 16)
Software Decision Module	Least square fitting method (M 13, section 17)
	Phase Operation (M 14, section 18)
	Data structure (M 15, section 19)
	Generic GUI/Plot (M 16, section 20)

Table 1: Module Hierarchy

6 MIS of STEM Moiré GPA Control Module (M 2)

6.1 Module

main

6.2 Uses

- STEM Moiré GPA GUI (M 3, section 7)
- Processing modules
 - Unstrained region (M 8, section 12)
 - Conversion (M 9, section 13)
 - SMH Simulation (M 5, section 9)
 - GPA(M 6, section 10)
 - 2D Strain Tensors (M 10, section 14)
- Input (M 4, section 8)
- Data Structure (M 15, section 19)

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

6.4 Semantics

STEM Moiré GPA is designed to have the process flow driven by user directly through GUI_SMG. The STEM Moiré GPA Control Module uses the events in STEM Moiré GPA GUI to use the processing modules in the order defined by the user.

6.4.1 State Variables

None

6.4.2 Access Routine Semantics

main():

- transition:

GUIFlow() *# Software permanently running until user abort it by closing the GUI*

GUI_Conv() *# Open the entry field GUI for the conversion process*

If one of the event below is triggered by the user, an action is performed by STEM Moiré GPA. The possible events are :

- (event_Input() → Get the path pathISMH and pathIC from the user → load_files(pathISMH,pathIC)) → GUI_SMHexp())
- (event_SimSMH() → SMHsim(load(ISMHexp), load(ICref), load(pISMHexp), load(pICref)) → GUI_SMHsim())
- For each GUI object mask M_j with $j = \{1, 2\}$ drawn by the user in the of GUI_SMHsim() window :
 1. (event_GPA() → gpa(load(FTISMHexp), collect_circ(M_j), id(M_j)) → GUI_Phase())
 2. (event_URef() → ZeroStrain(load_g(id(M_j))(deltagM), load_g(id(M_j))(gMuns), collect_rect(U), id(M_j)) → update GUI_Phase())
 3. (event_Conversion() → Read the n and m entry fields in GUI_Conv → conversion(load_g(id(M_j))(pISMHexp), load_g(id(M_j))(gMuns)), id(M_j))
- (event_StrainCalc() → CalcStrain(load(id(M_1), gCuns), load(id(M_2), gCuns), load(id(M_1), deltagM), load(id(M_2), deltagM)) → GUI_Strain())

7 MIS of STEM Moiré GPA GUI Module (M 3)

Specific GUI module to respect the requirements from the SRS

7.1 Module

GUI_SMG

7.2 Uses

- Generic GUI/Plot (M 16, section 20)
- Data Structure (M 15, section 19)

7.3 Syntax

7.3.1 Exported Access Programs

For the moment, the GUI exceptions are not mentioned to cover the other aspects of STEM Moiré GPA

Name	In	Out	Exceptions
GUIFlow	-	-	-
GUI_SMHexp	-	-	-
GUI_SMHSim	-	-	-
GUI_Phase	-	-	-
GUI_Conv	-	-	-
GUI_Strain	-	-	-
event_Input	-	-	-
event_SMHSim	-	-	-
event_GPA	-	-	-
event_URef	-	-	-
event_StrainCalc	-	-	-
collect_circ	GUI object	object	-
collect_rect	GUI object	object	-
id	GUI object	object	-

7.4 Semantics

STEM Moiré GPA process flow is driven by user through GUI_SMG. User triggers the events that start the selected processing step.

7.4.1 State Variables

None

7.4.2 Environment Variables

Win_Flow: GUI object
Win_SMHexp: GUI object
Win_SMHSim: GUI object
Win_Phase: GUI object
Win_FTSMH: GUI object
Win_Conv: GUI object
Win_Deltag: GUI object
Win_Strain: GUI object
button_Input: GUI object
button_SMHSim: GUI object

button_GPA: GUI object
 button_URef: GUI object
 button_StrainCalc: GUI object

7.4.3 Access Routine Semantics

GUI embedding the process flow into buttons triggering events. It is the user role to execute the process flow

GUIFlow():

- transition:
 1. Win_Flow=fig('Win_Flow')
 2. button(Win_Flow,5,'Input','SMHSim','GPA','URef','StrainCalc')
 3. plot()

Events triggered by each button pressed by the user

event_Input():

- transition: Trigger event_Input when button_Input pressed

event_SMHSim():

- transition: Trigger event_SMHSim when button_SMHSim pressed

event_GPA():

- transition: Trigger event_GPA when button_GPA pressed

event_URef():

- transition: Trigger event_URef when button_URef pressed

event_StrainCalc():

- transition: Trigger event_StrainCalc when button_StrainCalc press

GUI to display the display the input files $I_{SMH_{exp}}$, $I_{C_{ref}}$

GUI_SMHexp():

- transition:

```

1. Win_SMHexp=fig('Win_SMHexp',load( $I_{SMH_{exp}}$ ), load( $I_{C_{ref}}$ ))
2. plot()

# GUI to display the simulation of the STEM Moiré hologram using the reference image and
to let the user input  $M$  (from R 4, R 5)
GUI_SMHSim():
    • transition:
        1. Win_SMHSim=fig('Win_SMHSim',load(FTISMHexp),load(FTISMHsim),circle( $M$ ))
        2. plot()

# GUI to display the phase resulting from the GPA algorithm and to let the user input  $U$ 
(from R 8)
GUI_Phase(id):
    • transition:
        1. Win_Phase=fig('Win_Phase',load_g(id)(PhasegM),rectangle( $U$ ))
        2. Win_Deltag=fig('Win_Deltag',load_g(id)(deltagM))
        3. plot()

# GUI to display the window to let the user input  $n$  and  $m$  (from R 11)
GUI_Conv():
    • transition:
        1. Win_Conv=fig('Win_Conv',entry_field( $n$ ),entry_field( $m$ ))
        2. plot()

# GUI to display the window showing the final strain maps (from R 14)
GUI_Strain():
    • transition:
        1. Win_Strain=fig('Win_Strain',load(Exx),load(Eyy),load(Exy),load(Rxx))
        2. plot()

# Reader of the GUI objects drawn by the user (circle  $M$  or rectangle  $U$ )
collect_circ(A)

```

- output: C such that
 1. Execute `read_user_GUI(A)`
 2. Verify the type of the object `read_user_GUI(A)` to match a circle
 3. Output $C=(x_c, y_c, R)$ with (x_c, y_c) the coordinate (pixel number) of the center of the circle A and R the radius of the circle A .

`collect_rect(A)`

- output: S such that
 1. Execute `read_user_GUI(A)`
 2. Verify the type of the object `read_user_GUI(A)` to match a rectangle
 3. Get the coordinate of the upper left corner (x_0, y_0) and the coordinate of the bottom right corner (x_1, y_1) .
 4. output $S = ([x_0, x_1], [y_0, y_1])$

Function to read the unique id of a GUI object

`id(A)`

- output: B such that B is the id part of the `read_user_GUI(A)` output from the Generic GUI/Plot Module.

8 MIS of Input Module (M 4)

8.1 Module

Input

8.2 Uses

- STEM Moié GPA GUI (M 3, section 7)
- Data Structure (M 15, section 19)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
<code>load_files</code>	string	-	<code>badFilePath</code> , <code>badPixelP</code> , <code>badPixelPref</code> , <code>badIC</code> , <code>badISMH</code> , <code>badFileFormat</code>

8.4 Semantics

8.4.1 State Variables

data : object

8.4.2 Access Routine Semantics

Function to load $I_{SMH_{exp}}$, $I_{C_{ref}}$, p and p_{ref} and respect R 1 from the SRS.

load_files(pathISMH,pathIC):

- transition: pathISMH and pathIC are the file paths for the input files. The following procedure is performed:

1. Verify the format of the files to be .dm3
2. The .dm3 files are read and their respective metafiles are collected.
3. From the metafile, $I_{SMH_{exp}}$, $I_{C_{ref}}$, p and p_{ref} are extracted.
4. Verify $I_{SMH_{exp}}$, $I_{C_{ref}}$ to be 2D arrays of real numbers.
5. Verify p and p_{ref} to be strictly positive
6. The variables $I_{SMH_{exp}}$, $I_{C_{ref}}$, p and p_{ref} are stored in the data structure:
 - store(ISMHexp, $I_{SMH_{exp}}$)
 - store(pISMexp, p)
 - store(ICref, $I_{C_{ref}}$)
 - store(pICref, p_{ref})

- exception:

$\neg(p > 0)$	\Rightarrow badPixelP
$\neg(p_{ref} > 0)$	\Rightarrow badPixelP
$(\exists \vec{r} \in \mathbb{I}, I_{SMH_{exp}}(\vec{r}) \notin \mathbb{R})$	\Rightarrow badISMH
$(\exists \vec{r} \in \mathbb{I}, II_{C_{ref}}(\vec{r}) \notin \mathbb{R})$	\Rightarrow badIC
If the file targeted by pathISMH or pathIC doesn't exist	\Rightarrow badFilePath
If the file format is not appropriate	\Rightarrow badFileFormat

9 MIS of SMH Simulation (M 5)

9.1 Module

SMHSimCalc

9.2 Uses

- Fourier Transform (M 11, section 15)
- Data Structure (M 15, section 19)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
SMHsim	$I_{SMH_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ $I_{C_{\text{ref}}} : \mathbb{R}^2 \rightarrow \mathbb{R}, p \in \mathbb{R}^{+*}, p_{\text{ref}} \in \mathbb{R}^{+*}$	$\tilde{I}_{SMH_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{C}$ $\tilde{I}_{SMH_{\text{sim}}} : \mathbb{R}^2 \rightarrow \mathbb{C}$ $N_{\text{lim}} \in \mathbb{N}^*$	WarnNlim.zero()

9.4 Semantics

9.4.1 State Variables

data : object

9.4.2 Access Routine Semantics

SMHsim($I_{SMH_{\text{exp}}}, I_{C_{\text{ref}}}, p, p_{\text{ref}}$):

- transition:

1. store(FTISMHexp, $\tilde{I}_{SMH_{\text{exp}}}$) such that

$$\tilde{I}_{SMH_{\text{exp}}}(\vec{v}) = \mathcal{FT}[I_{SMH_{\text{exp}}}(\vec{r})]$$

2. store(FTISMHsim, $\tilde{I}_{SMH_{\text{sim}}}$) such that

$$\tilde{I}_{SMH_{\text{sim}}}(\vec{v}) = \frac{1}{p^2} \sum_{\vec{q} \in Q_{\text{lim}}} \mathcal{FT}[I_{C_{\text{ref}}}(\vec{v} - \frac{\vec{q}}{p})]$$

with $Q_{\text{lim}} = \{\forall(n, m) \in \mathbb{Z}^2 \cap [-N_{\text{lim}}, N_{\text{lim}}]^2, \vec{q} = n\vec{u}_x + m\vec{u}_y\}$

and $N_{\text{lim}} = \Xi(\frac{p}{p_{\text{ref}}})$ with Ξ the floor function

- exception:

10 MIS of GPA Module (M 6)

10.1 Module

GPACalc

10.2 Uses

- Mask (M 7, section 11)
- Fourier Transform (M 11, section 15)
- Phase (M 14, section 18)
- Gradient (M 12, section 16)
- Data Structure (M 15, section 19)

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
gpa	$\tilde{I}_{SMH_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{C}$, $M : \mathbb{N}^2 \times \mathbb{R}^{+*}$, id : id GUI object	$P_{\vec{g}} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\overrightarrow{\Delta g} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $P_{\Delta \vec{g}} : \mathbb{R}^2 \rightarrow \mathbb{R}$	NoMask, NoId

10.4 Semantics

10.4.1 State Variables

data : object

10.4.2 Access Routine Semantics

$\text{gpa}(\tilde{I}_{SMH_{\text{exp}}}, M, id)$:

- transition:

1. $M, \vec{g}^{M_{\text{exp}}} = \text{MCirc}(M)$
2. $\text{store_g}(id, \text{gMuns}, \vec{g}^{M_{\text{exp}}})$
3. Calculate $P_{\vec{g}}$ such that

$$\forall \vec{r} \in \mathbb{R}^2, P_{\vec{g}}(\vec{r}) = \arg(i\mathcal{FT}[M \times \tilde{I}_{SMH_{\text{exp}}}])$$

4. $\text{store}(id, \text{deltagM}, \overrightarrow{\Delta g})$ such that

$$\forall \vec{r} \in \mathbb{R}^2, \Delta \vec{g}(\vec{r}) = \frac{1}{2\pi} \text{grad}(\text{unwrap}(P_{\vec{g}}(\vec{r}))) - \vec{g}^{M_{\text{exp}}}(\vec{r})$$

5. $\text{store}(id, \text{PhasegM}, P_{\Delta \vec{g}})$ such that

$$\forall \vec{r} \in \mathbb{R}^2, P_{\Delta \vec{g}}(\vec{r}) = \text{wrap}(\text{unwrap}[P_{\vec{g}}(\vec{r})] - 2\pi \vec{g}^{M_{\text{exp}}}(\vec{0}) \cdot \vec{r})$$

- exception:
 $(M = \emptyset \Rightarrow \text{NoMask})$
 $(id = \emptyset \Rightarrow \text{NoId})$

11 MIS of Mask Module (M 7)

11.1 Module

Mask

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
MCirc	$(x_c, y_c) \in \mathbb{N}^2$, $R \in \mathbb{R}^{+*}$	$M : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\vec{g}_0 : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	BadRadius, BadCenter

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Access Routine Semantics

MCirc(x_c, y_c, R):

- output: M, \vec{g}_0
 - M such that

$$M(x, y) = \begin{cases} 1, & (x - x_c)^2 + (y - y_c)^2 \leq R^2 \\ 0, & (x - x_c)^2 + (y - y_c)^2 > R^2 \end{cases}$$

- \vec{g}_0 such that

$$\forall \vec{r} \in \mathbb{R}^2, \vec{g}_0(\vec{r}) = \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$

- exception:
 - $(\neg R > 0 \Rightarrow \text{BadRadius})$
 - $(\neg(x_c \in \mathbb{N} \wedge y_c \in \mathbb{N}) \Rightarrow \text{BadCenter})$

12 MIS of Unstrained region (M 8)

12.1 Module

URefCalc

12.2 Uses

- Least Square Fit (M 13, section 17)
- STEM Moiré GPA GUI (M 3, section 7)
- Data Structure (M 15, section 19)

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
ZeroStrain	$\overrightarrow{\Delta g}^M : \mathbb{R}^2 \rightarrow \mathbb{R}^2, U \in \mathbb{R}^2, \overrightarrow{g}^{M_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2, id : \text{id GUI object}$	$\overrightarrow{\Delta g}_{\text{cor}}^M : \mathbb{R}^2 \rightarrow \mathbb{R}^2, \overrightarrow{g}_{\text{uns}}^{M_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	BadU

12.4 Semantics

12.4.1 State Variables

data : object

12.4.2 Access Routine Semantics

ZeroStrain($\overrightarrow{\Delta g}^M, U, id$):

- transition:

1. store($id, \text{deltagM}, \overrightarrow{\Delta g}_{\text{cor}}^M$) such that

$$\overrightarrow{\Delta g}_{\text{cor}}^M = \overrightarrow{\Delta g}^M - \text{lsfm}(\overrightarrow{\Delta g}^M, U)$$

2. store($id, \text{gMuns}, \overrightarrow{g}_{\text{uns}}^{M_{\text{exp}}}$) such that

$$\overrightarrow{g}_{\text{uns}}^{M_{\text{exp}}} = \overrightarrow{g}^{M_{\text{exp}}} + \text{lsfm}(\overrightarrow{\Delta g}^M, U)$$

- exception:

13 MIS of Conversion Module (M 9)

13.1 Module

MtoCConv

13.2 Uses

- Data Structure (M 15, section 19)

13.3 Syntax

13.3.1 Exported Access Programs

Name	In	Out	Exceptions
conversion	$p \in \mathbb{R}$, $(n, m) \in \mathbb{N}^2$, $\vec{g}_{\text{uns}}^{M_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $id :$ id GUI object	$\vec{g}_{\text{uns}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	BadNM

13.4 Semantics

13.4.1 State Variables

data : object

13.4.2 Access Routine Semantics

conversion($p, n, m, \vec{g}_{\text{uns}}^{M_{\text{exp}}}, id$):

- transition: store_g($id, g_{\text{Cuns}}, \vec{g}_{\text{uns}}^{C_{\text{exp}}}$) such that

$$\forall \vec{r} \in \mathbb{R}^2, \vec{g}_{\text{uns}}^{C_{\text{exp}}}(\vec{r}) = \vec{g}_{\text{uns}}^{M_{\text{exp}}}(\vec{r}) + p \times \begin{bmatrix} n \\ m \end{bmatrix}$$

- exception:

14 MIS of 2D Strain Tensor Module (M 10)

14.1 Module

2D_Strain

14.2 Uses

Data Structure (M 15, section 19)

14.3 Syntax

14.3.1 Exported Access Programs

Name	In	Out	Exceptions
CalcStrain	$g_{1_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $g_{2_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $\Delta g_{1_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $\Delta g_{2_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	$\varepsilon_{xx} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\varepsilon_{yy} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\mathbb{R}^2 \rightarrow \mathbb{R}$, $\varepsilon_{xy} : \mathbb{R}^2 \rightarrow \mathbb{R}$, \mathbb{R} , $\omega_{xy} : \mathbb{R}^2 \rightarrow \mathbb{R}$	-

14.4 Semantics

14.4.1 State Variables

data : object

14.4.2 Access Routine Semantics

CalcStrain($g_{1_{\text{uns}}}^{C_{\text{exp}}}, g_{2_{\text{uns}}}^{C_{\text{exp}}}, \Delta g_{1_{\text{uns}}}^{C_{\text{exp}}}, \Delta g_{2_{\text{uns}}}^{C_{\text{exp}}}$):

- transition:

1. Form $G_{\text{uns}}^{\text{exp}}$ and ΔG^{exp} matrices such that

$$G_{\text{uns}}^{\text{exp}} = \begin{bmatrix} g_{1_{\text{uns}x}}^{C_{\text{exp}}} & g_{2_{\text{uns}x}}^{C_{\text{exp}}} \\ g_{1_{\text{uns}y}}^{C_{\text{exp}}} & g_{2_{\text{uns}y}}^{C_{\text{exp}}} \end{bmatrix}, \quad \Delta G^{\text{exp}}(\vec{r}) = \begin{bmatrix} \Delta g_{1_x}^{C_{\text{exp}}} & \Delta g_{1_y}^{C_{\text{exp}}} \\ \Delta g_{2_x}^{C_{\text{exp}}} & \Delta g_{2_y}^{C_{\text{exp}}} \end{bmatrix}$$

2. Calculate ∇u^{exp} such that

$$\nabla u^{\text{exp}} = ([G_{\text{uns}}^{\text{exp}} + \Delta G^{\text{exp}}]^T)^{-1} G_{\text{uns}}^{\text{exp}T} - I_d$$

3. Calculate ε^{exp} and ω^{exp}

$$\varepsilon^{\text{exp}} = \frac{1}{2}(\nabla u^{\text{exp}} + (\nabla u^{\text{exp}})^T) = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_{yy} \end{bmatrix}$$

$$\omega^{\text{exp}} = \frac{1}{2}(\nabla u^{\text{exp}} - (\nabla u^{\text{exp}})^T) = \begin{bmatrix} 0 & \omega_{xy} \\ -\omega_{xy} & 0 \end{bmatrix}$$

4. store(Exx, ε_{xx}), store(Eyy, ε_{yy}), store(Exy, ε_{xy}), store(Rxy, ω_{xy})

- exception:

15 MIS of Fourier Transform Module (M 11)

2D Fourier transform

15.1 Module

FTCalc

15.2 Uses

None

15.3 Syntax

15.3.1 Exported Access Programs

Name	In	Out	Exceptions
\mathcal{FT}	$f : \mathbb{R}^2 \rightarrow \mathbb{R}$	$f : \mathbb{R}^2 \rightarrow \mathbb{C}$	-
$\text{i}\mathcal{FT}$	$f : \mathbb{R}^2 \rightarrow \mathbb{C}$	$f : \mathbb{R}^2 \rightarrow \mathbb{R}$	-

15.4 Semantics

15.4.1 State Variables

None

15.4.2 Access Routine Semantics

Calculate the 2D Fourier transform of a function f

$\mathcal{FT}(f(x, y))$:

- output: $\tilde{f}(\nu, \mu)$ such that

$$\forall(\nu, \mu) \in \mathbb{R}^2 \wedge \forall(x, y) \in \mathbb{R}^2, \tilde{f}(\nu, \mu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2i\pi(\nu x + \mu y)} dx dy$$

- exception:

Calculate the 2D inverse Fourier transform of a function \tilde{f}

$\text{i}\mathcal{FT}(\tilde{f}(\nu, \mu))$:

- output: $f(x, y)$ such that

$$\forall(x, y) \in \mathbb{R}^2 \wedge \forall(\nu, \mu) \in \mathbb{R}^2, f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{f}(\nu, \mu) e^{2i\pi(\nu x + \mu y)} dx dy$$

- exception:

16 MIS of Gradient Module (M 12)

2D Gradient

16.1 Module

GradCalc

16.2 Uses

None

16.3 Syntax

16.3.1 Exported Access Programs

Name	In	Out	Exceptions
grad	$f : \mathbb{R}^2 \rightarrow \mathbb{R}$	$f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	-

16.4 Semantics

16.4.1 State Variables

16.4.2 Access Routine Semantics

Calculate the gradient of a 2D function f

grad(f):

- output: $\nabla f(x, y)$ such that

$$\forall (x, y) \in \mathbb{R}^2, \nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x, y) \\ \frac{\partial f}{\partial y}(x, y) \end{bmatrix}$$

- exception:

17 MIS of Least Square Fit Method Module (M 13)

2D linear least square method to fit a function f

17.1 Module

LSFMCalc

17.2 Uses

None

17.3 Syntax

17.3.1 Exported Access Programs

Name	In	Out	Exceptions
lsfm	$f : \mathbb{R}^2 \rightarrow \mathbb{R}^2, U \text{ in } \mathbb{R}^2$	$f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	-

17.4 Semantics

17.4.1 State Variables

17.4.2 Access Routine Semantics

Calculate the 2D fit of a function f using the linear least square method on a domain $U = ([x_0, x_1]; [y_0, y_1]) \in \mathbb{R}^2$

lsfm(f,U):

- output: $fit(x, y) = ax + by$ such that

$$\begin{aligned} \forall (x, y) \in U, E(a, b) &= \int_{x_0}^{x_1} \int_{y_0}^{y_1} [f(x, y) - fit(x, y)]^2 dx dy \text{ is minimized} \\ \Rightarrow \frac{\partial E}{\partial a} = 0 \wedge \frac{\partial E}{\partial b} = 0 &\Rightarrow a = \frac{\int_{x_0}^{x_1} \int_{y_0}^{y_1} x f(x, y) dx dy}{\int_{x_0}^{x_1} \int_{y_0}^{y_1} x^2 dx dy} \wedge b = \frac{\int_{x_0}^{x_1} \int_{y_0}^{y_1} y f(x, y) dx dy}{\int_{x_0}^{x_1} \int_{y_0}^{y_1} y^2 dx dy} \end{aligned}$$

- exception:

18 MIS of Phase Operation Module (M 14)

18.1 Module

PhaseCalc

18.2 Uses

None

18.3 Syntax

18.3.1 Exported Access Programs

Name	In	Out	Exceptions
unwrap	$f : \mathbb{R}^2 \rightarrow] - \pi, \pi]$	$f : \mathbb{R}^2 \rightarrow \mathbb{R}$	-
wrap	$f : \mathbb{R}^2 \rightarrow \mathbb{R}$	$f : \mathbb{R}^2 \rightarrow] - \pi, \pi]$	-
arg	$z \in \mathbb{C}$	$\phi \in] - \pi, \pi]$	-

18.4 Semantics

18.4.1 State Variables

18.4.2 Access Routine Semantics

wrap(f):

- output: g such that

$$\forall (x, y) \in \mathbb{R}^2, \exists k \in \mathbb{Z} | g(x, y) = f(x, y) + 2k\pi \wedge g(x, y) \in] - \pi, \pi]$$

- exception:

unwrap(f):

- output: g such that

$$\begin{aligned} & \forall (x, y) \in \mathbb{R}^2, \exists k \in \mathbb{Z} | g(x, y) = f(x, y) + 2k\pi \wedge g \text{ is continuous} \\ \Rightarrow & \forall (x, y) \in \mathbb{R}^2, \exists k \in \mathbb{Z} | \lim_{(x, y) \rightarrow (x_0, y_0)} g(x, y) = g(x_0, y_0) = f(x_0, y_0) + 2k\pi \end{aligned}$$

- exception:

arg(z):

- output: ϕ such that

$$\phi = \arg(z) \text{ with } z = e^{i\phi}$$

- exception:

19 MIS of Data Structure Module (M 15)

19.1 Module

DataStruct

19.2 Uses

None

19.3 Syntax

19.3.1 Exported Access Programs

Name	In	Out	Exceptions
store	string \times object	-	-
read	string	object	-
store_g	id GUI object \times string \times object	-	-
read_g	id GUI object \times string	object	-

19.4 Semantics

19.4.1 State Variables

Structure of the object carrying the data information

data : object

- data(ISMHexp) = $I_{SMH_{exp}}$
- data(pISMHexp) = p
- data(ICref) = $I_{C_{ref}}$
- data(pICref) = p_{ref}
- data(FTISMHexp) = $\tilde{I}_{SMH_{exp}}$
- data(FTISMHsim) = $\tilde{I}_{SMH_{sim}}$
- for each j data(Tj) : object
 - data(Tj)(gMuns) = $\vec{g}_j^{M_{exp}}$
 - data(Tj)(deltagM) = $\Delta \vec{g}_j^{M_{exp}}$
 - data(Tj)(PhasegM) = $P_{\Delta \vec{g}_j^{M_{exp}}}$
 - data(Tj)(shift) = (n_j, m_j)
 - data(Tj)(gCuns) = $\vec{g}_j^{C_{exp}}$
- data(Exx) = ε_{xx}
- data(Eyy) = ε_{yy}

- $\text{data}(\text{Exy}) = \varepsilon_{xy}$
- $\text{data}(\text{Rxy}) = \omega_{xy}$

19.4.2 Access Routine Semantics

$\text{store}(a, b)$:

- transition: $\text{data}(a) = b$

$\text{load}(a)$:

- output: $\text{data}(a)$

$\text{store_g}(id, a, b)$:

- transition: $\text{data}(id)(a) = b$

$\text{load_g}(id, a)$:

- output: $\text{data}(id)(a)$

20 MIS of Generic GUI/Plot Module (M 16)

20.1 Module

GUIGene

20.2 Uses

Hardware-Hiding Data Structure

20.3 Syntax

20.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	GUI objects	-	-
fig	$\text{string} \times \text{GUI objects}$	GUI object	-
button	$k \in \mathbb{N}$, string^k	GUI object	-
entry_field	string	GUI object	-
circle	-	GUI object	-
rectangle	-	GUI object	-
read_user_GUI	GUI object	object	-

20.4 Semantics

20.4.1 State Variables

20.4.2 Access Routine Semantics

plot():

- output: Display on the Hardware all the GUI objects

fig('label', optional GUI objects):

- output: Create a window GUI object with the optional GUI objects

button(*number*, 'labels'):

- output: Create *number* buttons GUI objects with their respective 'labels'

entry_field(*b*):

- output: Create a entry field GUI object to collect the input *b* from the user

circle(*C*(user_param)):

- output: Create a circle *C* GUI object drawn by the user

rectangle(*R*(user_param)):

- output: Create a rectangle *R* GUI object drawn by the user

read_user_GUI(A):

- output: B such that B includes the id of the GUI and the type of the GUI

References

- [1] D. M. Hoffman and P. A. Strooper, *Software Design, Automated Testing, and Maintenance: A Practical Approach*. New York, NY, USA: International Thomson Computer Press, 1995.
- [2] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. Upper Saddle River, NJ, USA: Prentice Hall, 2nd ed., 2003.

21 Appendix

[Extra information if required —SS]

All variables

$$\begin{aligned}
 &P_{\Delta \vec{g}_j^{M_{\text{exp}}}}(\vec{r}), \vec{g}_j^{M_{\text{exp}}}, \Delta \vec{g}_j^{M_{\text{exp}}}(\vec{r}) \\
 &\Delta \vec{g}_j^{M_{\text{exp}}}(\vec{r}), U, \vec{g}_j^{M_{\text{exp}}} \\
 &\vec{g}_j^{M_{\text{exp}}}, \Delta \vec{g}_j^{M_{\text{exp}}}(\vec{r}) \\
 &\vec{g}_j^{M_{\text{exp}}}, \Delta \vec{g}_j^{M_{\text{exp}}}(\vec{r}), \overrightarrow{q_{n_j, m_j}}, p \\
 &\Delta \vec{g}_j^{C_{exp}}(\vec{r}), \vec{g}_j^{C_{exp}}
 \end{aligned}$$