Module Interface Specification for STEM Moiré GPA

Alexandre Pofelski
macid: pofelska
github: slimpotatoes

November 28, 2017

1 Revision History

Date	Version	Notes
28/11/2017	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

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
complex	$\mathbb{C}$	a complex number
image space	$\mathbb{I}$	subset of $\mathbb{N}^2$ such that $\mathbb{I} = \{(x, y) \in \mathbb{N} \times \mathbb{N} : 0 \leq x \leq N-1 \wedge 0 \leq y \leq N-1\}$ with $N \in \mathbb{N}$ representing the number of pixels of the image in one direction

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	
Behaviour-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) 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)
Software Decision Module	Fourier Transform (M 11, section 15) Gradient (M 12, section 16) 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 Moié 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()
    $\rightarrow$ Get the path pathISMH and pathIC from the user $\rightarrow$
    load_files(pathISMH,pathIC)) $\rightarrow$ GUI_SMHexp())
  - (event_SimSMH() $\rightarrow$ SMHsim() $\rightarrow$ GUI_SMHsim())
  - For each circle GUI object C_j with $j = \{1, 2\}$
    drawn by the user in the of GUI_SMHsim() window :
    1. (event_GPA() $\rightarrow$ verifyM(collect_circ(C_j)) $\rightarrow$
       gpa(collect_circ(C_j), id(C_j)) $\rightarrow$ GUI_Phase())
    2. (event_URef() $\rightarrow$ verifyU(collect_rect(Rec)) $\rightarrow$
       ZeroStrain(collect_rect(Rec), id(C_j)) $\rightarrow$ update GUI_Phase())
    3. (event_Conversion() $\rightarrow$ Read the n and m entry fields in GUI_Conv $\rightarrow$
       verifyNM(n, m) $\rightarrow$ conversion(n, m, id(C_j))
  - (event_StrainCalc() $\rightarrow$ verifyMdiff(collect_circ(C_1), collect_circ(C_2)) $\rightarrow$
    CalcStrain(id(C_1), id(C_2)) $\rightarrow$ GUI_Strain())

[At current state, the MIS of the control module is not elegant. It is again due to a poor design (or poor understanding) of the control module and its interface with the other modules. The case with the GUI id is not optimum. Ideally, I would like to have the possibility to draw multiple masks and see their intermediate results to only choose the two best ones for the final calculation (StrainCalc(id1,id2)). For that, I would need a new module that could reference the different maks and enable the user to navigate through them. I didn't thought about that enough in advance and got stuck with that version. I will improve that in the future —Author]

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	$(x_c, y_c) \in \mathbb{I}, R \in \mathbb{R}^{+*}$	-
collect_rect	GUI object	$U \subset \mathbb{I}$	-
id	GUI object	id of GUI 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($I_{SMH_{\mathrm{exp}}}$), load($I_{C_{\mathrm{ref}}}$))
  2. plot()

# GUI to display the simulated STEM Moiré hologram using the reference image and to let the user input M on the experimental STEM Moiré hologram (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)

[I made lots of changes with the Input Module between the MG and the MIS. The original design was not optimal and I tried to fit every verification step in this input module. This is clearly not appropriate and I should design another module to verify the GUI inputs (M,U,n and m) separately. I just realized it too late to make the modification before the deadline but I will change that for sure. —Author]

8.1 Module

Input

8.2 Uses

- STEM Moiré 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
load_files	string	-	badFilePath, badPixelP, badPixelPref, badIC, badISMH, badFileFormat
verifyI	-	-	Not2Darray, NotReal
verifyp	-	-	BadPixelP
verifyM	-	-	NoMask, BadMask
verifyU	-	-	BadSizeU, NoU
verifyMdiff	-	-	SameMask
verifyNM	-	-	BadNM

8.4 Semantics

8.4.1 State Variables

data : object

8.4.2 Access Routine Semantics

# Function to load, verify and store $I_{SMH_{exp}}$, $I_{C_{ref}}$, p and p_{ref} (see R 1 and R 2 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. From the .dm3 metafile, $I_{SMH_{\text{exp}}}$, $I_{C_{\text{ref}}}$, p and p_{ref} are extracted.
  3. verifyI($I_{SMH_{\text{exp}}}$), verifyI($I_{C_{\text{ref}}}$)
  4. verifyp(p), verifyp(p_{ref})
  5. The variables $I_{SMH_{\text{exp}}}$, $I_{C_{\text{ref}}}$, p and p_{ref} are stored in the data structure:
     - store(ISMHexp, $I_{SMH_{\text{exp}}}$)
     - store(pISMexp, p)
     - store(ICref, $I_{C_{\text{ref}}}$)

    - store(pICref, p_{ref})

- exception:

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

verifyp(p):

- output: None
- exception:
  $\neg(p > 0) \Rightarrow$ badPixelP

verifyI(I):

- output: None
- exception:
  $(\exists \vec{r} \in \mathbb{I} : I(\vec{r}) \notin \mathbb{R}) \Rightarrow$ NotReal
  $\neg(I : \mathbb{I} \rightarrow \mathbb{R}) \Rightarrow$ Not2Darray

# Functions to verify the size of the unstrained reference input U (see R 9), the size of the mask input M_j and if both masks are different (see R 6).

verifyU(U):

- output: None
- exception:
  $(U = \emptyset) \Rightarrow$ NoU
  $\neg(U \subset \mathbb{I}) \Rightarrow$ BadSizeU

verifyM(M_j):

- output: None
- exception:
  $(M_j = \emptyset) \Rightarrow$ NoMask
  $\neg((x_c, y_c) \in \mathbb{I}) \Rightarrow$ BadMask

verifyMdiff(M_1, M_2):

- output: None
- exception:
  $(M_1 = M_2) \Rightarrow$ SameMask

verifyMN(M_1, M_2):

- output: None
- exception:
  $\neg(n \in \mathbb{N} \land m \in \mathbb{N}) \Rightarrow$ BadNM

9 MIS of SMH Simulation (M 5)

# Module simulating the STEM Moiré hologram using a reference image (see R 3 in the SRS)

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	-	-	WarnNlimzero

9.4 Semantics

9.4.1 State Variables

data : object

9.4.2 Access Routine Semantics

SMHsim():

- transition:
  1. Load the inputs from the data object
     - $I_{SMH_{\text{exp}}}$=load(ISMHexp)
     - $I_{C_{\text{ref}}}$=load(ICref)
     - p=load(pISMHexp)
     - p_{ref}=load(pICref)
  2. store(FTISMHexp, $\widetilde{I}_{SMH_{\text{exp}}}$) such that

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

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

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

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

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

- exception:
  $(N_{lim} = 0) \Rightarrow$ WarnNlimzero # *Exception warns the user that the sampling parameter chosen is oversampling the crystal periodicity. Classic HRSTEM GPA case !! It's not a STEM Moiré interferometry experiment but the software still works for the classic case*

10 MIS of GPA Module (M 6)

Module performing the GPA algorithm to isolate one spatial frequency and output the phase component (see R 7 in the SRS)

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	$C \in \mathbb{N}^2 \times \mathbb{R}$, id : id of GUI object	-	-

10.4 Semantics

10.4.1 State Variables

data : object

10.4.2 Access Routine Semantics

gpa(C,id):

- transition:
  1. $M, \overrightarrow{g}^{M_{\text{exp}}} = \text{MCirc}(C)$
  2. store_g(id, Mask, C), store_g(id, gMuns, $\overrightarrow{g}^{M_{\text{exp}}}$)
  3. $\widetilde{I}_{SMH_{\text{exp}}}$=load(FTISMHexp)
  4. Calculate $P_{\vec{g}}$ such that
  $$\forall \vec{r} \in \mathbb{R}^2,\ P_{\vec{g}}(\vec{r}) = \arg(i\mathcal{FT}[M \times \widetilde{I}_{SMH_{\text{exp}}}])$$
  5. store_g(id,deltagM,$\overrightarrow{\Delta g}$) such that
  $$\forall \vec{r} \in \mathbb{I},\ \Delta \overrightarrow{g}(\vec{r}) = \frac{1}{2\pi}\text{grad}(\text{unwrap}(P_{\vec{g}}(\vec{r}))) - \overrightarrow{g}^{M_{\text{exp}}}(\vec{r})$$
  6. store_g(id,PhasegM,$P_{\Delta\vec{g}}$) such that
  $$\forall \vec{r} \in \mathbb{I},\ P_{\Delta\vec{g}}(\vec{r}) = \text{wrap}(\text{unwrap}[P_{\vec{g}}(\vec{r})] - 2\pi\overrightarrow{g}^{M_{\text{exp}}}(\vec{0}) \cdot \vec{r})$$
- exception: none

11 MIS of Mask Module (M 7)

# Module transforming the circle drawn to a mask function usable by the GPA module.

11.1 Module

Mask

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
MCirc	$(x_c, y_c) \in \mathbb{I}$, $R \in \mathbb{R}^{+*}$	$M : \mathbb{I} \to \mathbb{R}$, $\overrightarrow{g_0} : \mathbb{I} \to \mathbb{R}^2$	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,$\overrightarrow{g_0}$
  - M such that
    $$\forall \vec{r} \in \mathbb{I},\ M(x, y) = \begin{cases} 1, & (x - x_c)^2 + (y - y_c)^2 \leq R^2 \\ 0, & (x - x_c)^2 + (y - y_c)^2 > R^2 \end{cases}$$
  - $\overrightarrow{g_0}$ such that
    $$\forall \vec{r} \in \mathbb{I},\ \overrightarrow{g_0}(\vec{r}) = \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$
- exception:
  $\neg(R > 0) \Rightarrow$ BadRadius
  $\neg((x_c, y_c) \in \mathbb{I}) \Rightarrow$ BadCenter

12 MIS of Unstrained region Module (M 8)

# Module correcting the g vector of reference using the user input U (see R 10 in the SRS)

12.1 Module

URefCalc

12.2 Uses

- Least Square Fit (M 13, section 17)
- Data Structure (M 15, section 19)

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
ZeroStrain	$U \subset \mathbb{I}$, id : id of GUI object	-	-

12.4 Semantics

12.4.1 State Variables

data : object

12.4.2 Access Routine Semantics

ZeroStrain(U, id):

- transition:
  1. $\overrightarrow{\Delta g}^{M_{\exp}}$=load_g($id$, deltag), $\overrightarrow{g}^{M_{\exp}}$=load_g($id$, gMuns)
  2. store(U, U)
  3. store(id, deltagM, $\overrightarrow{\Delta g}^{M_{\exp}}_{\text{cor}}$) such that

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

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

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

- exception:
  ($\neg U \subset \mathbb{I} \Rightarrow$ BadSizeU)

13 MIS of Conversion Module (M 9)

# Module converting the Moire g vector into the crystalline g vector (see R 12 in the SRS)

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	$(n,m) \in \mathbb{N}^2$, id : id GUI object	-	-

13.4 Semantics

13.4.1 State Variables

data : object

13.4.2 Access Routine Semantics

conversion(n, m, id):

- transition:

  1. store_g(id, shift, (n, m))
  2. $\overrightarrow{g_j}^{M_\text{exp}}_\text{uns}$=load_g($id$,gMuns), p=load(pISMHref)
  3. store_g(id,gCuns,$\overrightarrow{g_j}^{C_\text{exp}}_\text{uns}$) such that

$$\forall \vec{r} \in \mathbb{I},\ \overrightarrow{g_j}^{C_\text{exp}}_\text{uns}(\vec{r}) = \overrightarrow{g_j}^{M_\text{exp}}_\text{uns}(\vec{r}) + \frac{1}{p} \times \begin{bmatrix} n \\ m \end{bmatrix}$$

- exception:

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

# Module calculating the strain and rotation tensors using two non collinear crystalline wave vectors (see R 13 in the SRS)

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	$(id : \text{id of GUI object})^2$	-	MissingGInfo

14.4 Semantics

14.4.1 State Variables

data : object

14.4.2 Access Routine Semantics

CalcStrain($id1, id2$):

- transition:
  1. Load the variables from the data structure
     - $g_{1_{\text{uns}}}^{C_{\text{exp}}}$ = load_g($id1$, gCuns)
     - $\Delta g_{1_{\text{uns}}}^{C_{\text{exp}}}$ = load_g($id1$, deltagM)
     - $g_{2_{\text{uns}}}^{C_{\text{exp}}}$ = load_g($id2$, gCuns)
     - $\Delta g_{2_{\text{uns}}}^{C_{\text{exp}}}$ = load_g($id2$, deltagM)
  2. Form $G_{\text{uns}}^{\text{exp}}$ and ΔG^{exp} matrices such that

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

  3. Calculate ∇u^{exp} such that

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

  4. Calculate ε^{exp} and ω^{exp}

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

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

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

- exception:
  $(g_{1_{\text{uns}}}^{C_{\exp}} = \emptyset \vee g_{2_{\text{uns}}}^{C_{\exp}} = \emptyset \vee \Delta g_{1_{\text{uns}}}^{C_{\exp}} = \emptyset \vee \Delta g_{2_{\text{uns}}}^{C_{\exp}} = \emptyset) \Rightarrow$ MissingGinfo

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**
$\mathcal{FT}$	$f : \mathbb{R}^2 \to \mathbb{R}$	$f : \mathbb{R}^2 \to \mathbb{C}$	-
i$\mathcal{FT}$	$f : \mathbb{R}^2 \to \mathbb{C}$	$f : \mathbb{R}^2 \to \mathbb{R}$	-

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: $\widetilde{f}(\nu,\mu)$ such that

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

- exception:

# Calculate the 2D inverse Fourier transform of a function $\widetilde{f}$

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

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

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

- 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	$f : \mathbb{R}^2 \to \mathbb{R}$	$f : \mathbb{R}^2 \to \mathbb{R}^2$	-

16.4 Semantics

16.4.1 State Variables

None

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	$f : \mathbb{R}^2 \to \mathbb{R}^2$, $U \in \mathbb{R}^2$	$f : \mathbb{R}^2 \to \mathbb{R}^2$	-

17.4 Semantics

17.4.1 State Variables

None

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

$$\forall (x, y) \in U,\ E(a, b) = \int_{x_0}^{x_1} \int_{y_0}^{y_1} [f(x, y) - fit(x, y)]^2 dxdy \text{ is minimized}$$

$$\Rightarrow \frac{\partial E}{\partial a} = 0 \wedge \frac{\partial E}{\partial b} = 0 \Rightarrow a = \frac{\int_{x_0}^{x_1} \int_{y_0}^{y_1} xf(x, y)dxdy}{\int_{x_0}^{x_1} \int_{y_0}^{y_1} x^2 dxdy} \wedge b = \frac{\int_{x_0}^{x_1} \int_{y_0}^{y_1} yf(x, y)dxdy}{\int_{x_0}^{x_1} \int_{y_0}^{y_1} y^2 dxdy}$$

- exception:

18 MIS of Phase Operation Module (M 14)

# Module embedding a few functions to make possible some mathematical operations on the phase (parameter bound between $]-\pi, \pi]$*).*

18.1 Module

PhaseCalc

18.2 Uses

None

18.3 Syntax

18.3.1 Exported Access Programs

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

18.4 Semantics

18.4.1 State Variables

None

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:
  $(\exists (x_0, y_0) \in \mathbb{R}^2 : f(x_0, y_0) \notin]-\pi, \pi]) \Rightarrow$ BadPhase

unwrap(f):

- output: g such that

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

- 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 × object	-	-
read	string	object	-
store_g	id GUI object × string × object	-	-
read_g	id GUI object × string	object	-

19.4 Semantics

19.4.1 State Variables

# Structure of the object carrying the data information

data : object

- data(ISMHexp)=$I_{SMH_{\text{exp}}} : \mathbb{I} \to \mathbb{R}$
- data(pISMHexp)= $p \in \mathbb{R}^{+*}$
- data(ICref)=$I_{C_{\text{ref}}} : \mathbb{I} \to \mathbb{R}$
- data(pICref)=$p_{\text{ref}} \in \mathbb{R}^{+*}$
- data(FTISMHexp)=$\widetilde{I}_{SMH_{\text{exp}}} : \mathbb{I} \to \mathbb{C}$
- data(FTISMHsim)=$\widetilde{I}_{SMH_{\text{sim}}} : \mathbb{I} \to \mathbb{C}$
- data(URef)=$U \subset \mathbb{I}$
- for each $j = \{1, 2\}$ data(Mj) : object
  - data(Mj)(mask)=$x_c, y_c, R \in \mathbb{N} \times \mathbb{N} \times \mathbb{R}^{+*}$
  - data(Mj)(gMuns)=$\overrightarrow{g_j}^{M_{\text{exp}}}_{\text{uns}} : \mathbb{I} \to \mathbb{R}^2$

  - data(Mj)(deltagM)=$\Delta\overrightarrow{g_j}^{M_{\text{exp}}} : \mathbb{I} \rightarrow \mathbb{R}^2$
  - data(Mj)(PhasegM)=$P_{\Delta\overrightarrow{g_j}^{M_{\text{exp}}}} : \mathbb{I} \rightarrow \mathbb{R}$
  - data(Mj)(shift)=$(n_j, m_j) \in \mathbb{N}^2$
  - data(Mj)(gCuns)=$\overrightarrow{g_j}^{C_{\text{exp}}}_{\text{uns}} : \mathbb{I} \rightarrow \mathbb{R}^2$

- data(Exx)=$\varepsilon_{\text{xx}} : \mathbb{I} \rightarrow \mathbb{R}$
- data(Eyy)=$\varepsilon_{\text{yy}} : \mathbb{I} \rightarrow \mathbb{R}$
- data(Exy)=$\varepsilon_{\text{xy}} : \mathbb{I} \rightarrow \mathbb{R}$
- data(Rxy)=$\omega_{\text{xy}} : \mathbb{I} \rightarrow \mathbb{R}$

19.4.2 Access Routine Semantics

store(a,b):

- transition: data(a)=b

load(a):

- output: data(a)

store_g(id,a,b):

- transition: data(id)(a)=b

load_g(id,a):

- output: data(id)(a)

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

[The description of this module is pretty sparse ... I didn't want to go through all the possible interaction a GUI and a plotter can do. I also think that it is not the major element for the course. On the other hand, a nice GUI makes lots of difference from the user perspective. —Author]

20.1 Module

GUIGene

20.2 Uses

Hardware-Hiding (M 1)

20.3 Syntax

20.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	GUI objects	-	-
fig	string $\times$ 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

None

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.