Module Interface Specification for STEM Moiré GPA

Alexandre Pofelski
macid: pofelska
github: slimpotatoes

November 26, 2017

1 Revision History

Date	Version	Notes
26/11/2017	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.

addition to document

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

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
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	
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

6.4.2 Access Routine Semantics

main():

- transition: A reflechir

  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() →
  $\text{SMHsim}(\text{load}(\text{ISMHexp}), \text{load}(\text{ICref}), \text{load}(\text{pISMHexp}), \text{load}(\text{pICref})$
  → 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() → $\text{gpa}(\text{load}(\text{FTISMHexp}), \text{collect\_circ}(M_j), \text{id}(M_j))$
     → GUI_Phase())
  2. (event_URef() →
     $\text{ZeroStrain}(\text{load\_g}(\text{id}(M_j))(\text{deltagM}), \text{load\_g}(\text{id}(M_j))(\text{gMuns}),$
     $\text{collect\_rect}(U), \text{id}(M_j))$ → update GUI_Phase())
  3. (event_Conversion() → Read the n and m entry fields in GUI_Conv →
     $\text{conversion}(\text{load\_g}(\text{id}(M_j))(\text{pISMHexp}), \text{load\_g}(\text{id}(M_j)(\text{gMuns})), \text{id}(M_j))$
- (event_StrainCalc() →
  $\text{CalcStrain}(\text{load}(\text{id}(M_1), \text{gCuns}), \text{load}(\text{id}(M_2), \text{gCuns}), \text{load}(\text{id}(M_1), \text{deltagM}),$
  $\text{load}(\text{id}(M_2), \text{deltagM}))$ → GUI_Strain())

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

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

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	-

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

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

7.4.2 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_Inpu 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 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 $S = ([x_0, x_1], [y_0, y_1])$

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
load_files	string	-	FilePath

8.4 Semantics

8.4.1 State Variables

data : object

8.4.2 Access Routine Semantics

load_files(pathISMH,pathIC):

- transition: pathISMH and pathIC are the file paths for the input files. The following procedure is performed:
  1. The .dm3 files are read and their respective metafiles are collected.
  2. From the metafile, $I_{SMH_{\text{exp}}}$, $I_{C_{\text{ref}}}$, p and p_{ref} are extracted.
  3. 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:

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	$I_{SMH_{\text{exp}}} : \mathbb{R}^2 \to \mathbb{R}$ $I_{C_{\text{ref}}} : \mathbb{R}^2 \to \mathbb{R}$, $p \in \mathbb{R}^{+*}$, $p_{\text{ref}} \in \mathbb{R}^{+*}$	$\widetilde{I}_{SMH_{\text{exp}}} : \mathbb{R}^2 \to \mathbb{C}$ $\widetilde{I}_{SMH_{\text{sim}}} : \mathbb{R}^2 \to \mathbb{C}$ $N_{\text{lim}} \in \mathbb{N}^*$	Nlim.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, $\widetilde{I}_{SMH_{\text{exp}}}$) such that

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

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

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

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

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

- 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	$\widetilde{I}_{SMH_{\rm exp}} : \mathbb{R}^2 \to \mathbb{C}$, $M : \mathbb{R}^2 \to \mathbb{R}$, $\vec{g}^{M_{\rm exp}} : \mathbb{R}^2 \to \mathbb{R}$, id : id GUI object	$P_{\vec{g}} : \mathbb{R}^2 \to \mathbb{R}$, $\overrightarrow{\Delta g} : \mathbb{R}^2 \to \mathbb{R}^2$, $P_{\Delta\vec{g}} : \mathbb{R}^2 \to \mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

data : object

10.4.2 Access Routine Semantics

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

- transition:
  1. M, $\overrightarrow{g}^{M_{\exp}} = \text{MCirc}(M)$
  2. store_g(id,gMuns,$\overrightarrow{g}^{M_{\exp}}$)
  3. Calculate $P_{\vec{g}}$ such that
  $$\forall \vec{r} \in \mathbb{R}^2,\ P_{\vec{g}}(\vec{r}) = \arg(i\mathcal{FT}[M \times \widetilde{I}_{SMH_{\exp}}])$$
  4. store(id,deltagM,$\overrightarrow{\Delta g}$) such that
  $$\forall \vec{r} \in \mathbb{R}^2,\ \Delta\overrightarrow{g}(\vec{r}) = \frac{1}{2\pi}\text{grad}(\text{unwrap}(P_{\vec{g}}(\vec{r}))) - \overrightarrow{g}^{M_{\exp}}(\vec{r})$$
  5. store(id,PhasegM,$P_{\Delta\vec{g}}$) such that
  $$\forall \vec{r} \in \mathbb{R}^2,\ P_{\Delta\vec{g}}(\vec{r}) = \text{wrap}(\text{unwrap}[P_{\vec{g}}(\vec{r})] - 2\pi\overrightarrow{g}^{M_{\exp}}(\vec{0}) \cdot \vec{r})$$
- exception:

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	$(x_c, y_c) \in \mathbb{N}^2$, $R \in \mathbb{R}^{+*}$	$M : \mathbb{R}^2 \to \mathbb{R}$, $\overrightarrow{g_0} : \mathbb{R}^2 \to \mathbb{R}^2$	-

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

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

  - $\overrightarrow{g_0}$ such that

$$\forall \vec{r} \in \mathbb{R}^2,\ \overrightarrow{g_0}(\vec{r}) = \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$

- exception:

12 MIS of Unstrained region (M 8)

12.1 Module

UProfCalc

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	$\overrightarrow{\Delta g}^M : \mathbb{R}^2 \to \mathbb{R}^2$, $U \in \mathbb{R}^2$, $\overrightarrow{g}^{M_{\text{exp}}} : \mathbb{R}^2 \to \mathbb{R}^2$, id : id GUI object	$\overrightarrow{\Delta g}^M_{\text{cor}} : \mathbb{R}^2 \to \mathbb{R}^2$, $\overrightarrow{g}^{M_{\text{exp}}}_{\text{uns}} : \mathbb{R}^2 \to \mathbb{R}^2$	-

12.4 Semantics

12.4.1 State Variables

12.4.2 Access Routine Semantics

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

- transition:
  1. store(id,deltagM,$\overrightarrow{\Delta g}^M_{\text{cor}}$) such that
  $$\overrightarrow{\Delta g}^M_{\text{cor}} = \overrightarrow{\Delta g}^M - \text{lsfm}(\overrightarrow{\Delta g}^M, U)$$
  2. store(id,gMuns,$\overrightarrow{g}^{M_{\text{exp}}}_{\text{uns}}$) such that
  $$\overrightarrow{g}^{M_{\text{exp}}}_{\text{uns}} = \overrightarrow{g}^{M_{\text{exp}}} + \text{lsfm}(\overrightarrow{\Delta g}^M, U)$$
- 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	$p \in \mathbb{R}$, $(n, m) \in \mathbb{N}^2$, $\overrightarrow{g}^{M_{\text{exp}}}_{\text{uns}} : \mathbb{R}^2 \to \mathbb{R}^2$, id : id GUI object	$\overrightarrow{g_{uns}}^{C_{exp}} : \mathbb{R}^2 \to \mathbb{R}^2$	-

13.4 Semantics

13.4.1 State Variables

data : object

13.4.2 Access Routine Semantics

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

- transition: store_g(id,gCuns,$\overrightarrow{g_j}^{C_{\text{exp}}}_{\text{uns}}$) such that
$$\forall \vec{r} \in \mathbb{R}^2,\ \overrightarrow{g_j}^{C_{\text{exp}}}_{\text{uns}}(\vec{r}) = \overrightarrow{g_j}^{M_{\text{exp}}}_{\text{uns}}(\vec{r}) + p \times \begin{bmatrix} n \\ m \end{bmatrix}$$
- 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	$g_{1_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $g_{2_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $\Delta g_{1_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $\Delta g_{2_{\text{uns}}}^{C_{\text{exp}}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$	$\varepsilon_{xx} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\varepsilon_{yy} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\varepsilon_{xy} : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\omega_{xy} : \mathbb{R}^2 \rightarrow \mathbb{R}$	-

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

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

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

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

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

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

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

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

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

- exception:

18 MIS of Phase Operation Module (M 14)

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

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

- 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	GUI object × string × object	-	-
read_g	GUI object × string	object	-

19.4 Semantics

19.4.1 State Variables

 # Structure of the object carrying the data information

data : object

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

- data(Exy)=$I\varepsilon_{\mathrm{xy}}$
- data(Rxy)=$Iomega_{\mathrm{xy}}$

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)

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	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

20.4.2 Access Routine Semantics

plot():

- transition:
- output: Display on the Hardware all the GUI objects

fig('label', optional GUI objects):

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

button(*number*,'labels'):

- transition: [if appropriate —SS]
- output: Create *number* buttons GUI objects with their respective 'labels'

entry_field(b):

- transition: [if appropriate —SS]
- output: Create a entry field GUI object to collect the input b from the user

circle(C(user_param)):

- transition: [if appropriate —SS]
- output: Create a circle C GUI object drawn by the user

rectangle(R(user_param)):

- transition: [if appropriate —SS]
- 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
$P_{\Delta\overrightarrow{g_j}^{M_{\mathrm{exp}}}}(\vec{r})$, $\overrightarrow{g_j}^{M_{\mathrm{exp}}}$, $\Delta\overrightarrow{g_j}^{M_{\mathrm{exp}}}(\vec{r})$
$\Delta\overrightarrow{g_j}^{M_{\mathrm{exp}}}(\vec{r}), U, \overrightarrow{g_j}^{M_{\mathrm{exp}}}$
$\overrightarrow{g_j}_{\mathrm{uns}}^{M_{\mathrm{exp}}}, \Delta\overrightarrow{g_j}_{\mathrm{cor}}^{M_{\mathrm{exp}}}(\vec{r})$
$\overrightarrow{g_j}_{\mathrm{uns}}^{M_{\mathrm{exp}}}, \Delta\overrightarrow{g_j}_{\mathrm{cor}}^{M_{\mathrm{exp}}}(\vec{r}), \overrightarrow{q_{n_j,m_j}}, p$
$\Delta\overrightarrow{g_j}^{C_{exp}}(\vec{r}), \overrightarrow{g_j}_{uns}^{C_{exp}}$

$$\begin{aligned}
\forall\vec{r}\in\mathbb{I},\ G_{\mathrm{uns}}^{\mathrm{exp}} &= \begin{bmatrix} g_{1_{\mathrm{uns}_x}}^{C_{\mathrm{exp}}} & g_{2_{\mathrm{uns}_x}}^{C_{\mathrm{exp}}} \\ g_{2_{\mathrm{uns}_x}}^{C_{\mathrm{exp}}} & g_{2_{\mathrm{uns}_y}}^{C_{\mathrm{exp}}} \end{bmatrix} \\
\forall\vec{r}\in\mathbb{I},\ \Delta G^{\mathrm{exp}}(\vec{r}) &= \begin{bmatrix} \Delta g_{1_x}^{C_{\mathrm{exp}}}(\vec{r}) & \Delta g_{1_y}^{C_{\mathrm{exp}}}(\vec{r}) \\ \Delta g_{2_x}^{C_{\mathrm{exp}}}(\vec{r}) & \Delta g_{2_y}^{C_{\mathrm{exp}}}(\vec{r}) \end{bmatrix} \\
\forall\vec{r}\in\mathbb{I},\ G^{\mathrm{exp}}(\vec{r}) &= G_{\mathrm{uns}}^{\mathrm{exp}} + \Delta G^{\mathrm{exp}}(\vec{r}) \\
\forall\vec{r}\in\mathbb{I},\ \nabla u^{\mathrm{exp}}(\vec{r}) &= (G^{\mathrm{exp}}(\vec{r})^T)^{-1} G_{\mathrm{uns}}^{\mathrm{exp}T} - I_d
\end{aligned} \tag{1}$$

Then, the rotation ω^{exp} and strain $\varepsilon^{\mathrm{exp}}$ tensors are determined on each pixel as follows:

$$\begin{aligned}
\forall\vec{r}\in\mathbb{I},\ \nabla u^{\mathrm{exp}}(\vec{r}) &= \varepsilon^{\mathrm{exp}}(\vec{r}) + \omega^{\mathrm{exp}}(\vec{r}) \\
\forall\vec{r}\in\mathbb{I},\ \varepsilon^{\mathrm{exp}}(\vec{r}) &= \begin{bmatrix} \varepsilon_{xx^{\mathrm{exp}}}(\vec{r}) & \varepsilon_{xy^{\mathrm{exp}}}(\vec{r}) \\ \varepsilon_{xy^{\mathrm{exp}}}(\vec{r}) & \varepsilon_{yy^{\mathrm{exp}}}(\vec{r}) \end{bmatrix} \\
\forall\vec{r}\in\mathbb{I},\ \omega^{\mathrm{exp}}(\vec{r}) &= \begin{bmatrix} 0 & \omega_{xy^{\mathrm{exp}}}(\vec{r}) \\ -\omega_{xy^{\mathrm{exp}}}(\vec{r}) & 0 \end{bmatrix} \\
\forall\vec{r}\in\mathbb{I},\ \varepsilon^{\mathrm{exp}}(\vec{r}) &= \frac{1}{2}(\nabla u^{\mathrm{exp}}(\vec{r}) + (\nabla u^{\mathrm{exp}}(\vec{r}))^T) \\
\forall\vec{r}\in\mathbb{I},\ \omega^{\mathrm{exp}}(\vec{r}) &= \frac{1}{2}(\nabla u^{\mathrm{exp}}(\vec{r}) - (\nabla u^{\mathrm{exp}}(\vec{r}))^T)
\end{aligned} \tag{2}$$