Module Interface Specification for Slope Stability Analysis Program (SSP)

Henry Frankis and Brooks MacLachlan

December 3, 2018

1 Revision History

Date	Version	Notes
11/12/18	1.0	Initial updates based on template
11/21/18	1.1	Finished updating all of the modules
11/29/18	1.2	Added additional constants to the genetic algorithm module for the adding of vertices to slip surfaces

2 Symbols, Abbreviations and Acronyms

See Section 2 of the Software Requirements Specification (SRS) document, available in the GitHub repository for the project.

Contents

3 Introduction

The following document details the Module Interface Specifications for SSP, a program for determining the critical slip surface and corresponding factor of safety for a given sloped mass of soil. The document is intended to ease understanding of the design of SSP and should be used as a resource for any maintenance of SSP.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at the GitHub repository for the project.

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 $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \ldots | c_n \Rightarrow r_n)$. The notation for quantifiers is from Gries and Schneider (1993).

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

Data Type	Notation	Description
character	char	a single symbol or digit
boolean	$\mathbb{B}$	a value from the set {true, false}
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$

The specification of SSP uses some derived data types: sequences, strings, and tuples. Sequences are ordered lists of elements of the same data type, denoted by brackets enclosing the type of the data elements. If a sequence has fixed dimensions, the notation of the type will include the dimensions in superscript. Strings are sequences of characters. Tuples contain a list of values, potentially of different types, each associated with a field identifier. When a tuple is referenced in this document, a link to an appendix section that specifies the fields of the tuple will be provided. In addition, SSP 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 Numerical Algorithms

Morgenstern-Price (Section 13)

The non-linear nature of the systems of equations in the Morgenstern-Price solver algorithm requires that the equations for the factor of safety (IM1), the interslice normal-to-shear force ratio (IM2), and the interslice normal forces (IM3) are solved iteratively, with an initial guess for two of the values, typically the factor of safety and interslice normal-to-shear force ratio.

Genetic Algorithm (Section 9)

SSP uses a genetic algorithm to find the coordinates of the critical slip surface vertices that minimize the factor of safety, as described in IM4. The genetic algorithm generates a set of initial potential slip surfaces, and subsequent generations are created by merging and mutating slip surfaces with low factors of safety from the previous generation. The minimum factor of safety after several generations is assumed to correspond to the critical slip surface.

[This section is not on the template. I've left it in for now because the information does seem useful, but maybe this is not the right place for it? Maybe this should go to an appendix? —BM]

6 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Control
	Input
	Genetic Algorithm
	Kinematic Admissibility
	Slip Weighting
	Slip Slicing
	Morgenstern-Price Calculation
	Slice Property Calculation
	Output
Software Decision	Sequence Data Structure
	Random Number Generation
	Plotting

Table 1: Module Hierarchy

7 MIS of the Control Module

7.1 Module

Control

7.2 Uses

Input (Section 8), Output (Section 15), GenAlg (Section 9), Sequence (Section 16)

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Data Types

N/A

7.3.3 Exported Access Programs

Name	In	Out	Exceptions
Control	string	-	-

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

The access program is called with a string parameter.

7.4.4 Access Routine Semantics

control(*fname*):

- transition:

  Modifies the state of the Input Module, Genetic Algorithm Module, and Output Module.

7.4.5 Local Functions

N/A

8 MIS of the Input Module

8.1 Module

Input

8.2 Uses

Sequence (Section 16)

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Data Types

coord	=	tuple of (x : $\mathbb{R}$, y : $\mathbb{R}$)
coords	=	[coord]
paramsLayers	=	tuple of (strat : coords, phi : $\mathbb{R}$, coh : $\mathbb{R}$, gam : $\mathbb{R}$, gams : $\mathbb{R}$) (Appendix 19.1.1)
paramsPiez	=	tuple of (piez : coords, gamw : $\mathbb{R}$) (Appendix 19.1.2)
paramsSearch	=	tuple of (Xext, Xetr, Ylim : $[\mathbb{R}]^{1,2}$) (Appendix 19.1.3)
paramsSoln	=	tuple of (ltor, ftype, evnslc, cncvu, obtu : $\mathbb{B}$) (Appendix 19.1.4)

8.3.3 Exported Access Programs

Name	In	Out	Exceptions
load_params	string	-	fileNotExist, badFileExtension, unexpectedInput
verify_params	-	-	badSlopeGeometry, badEffAngleFriction, badCohesion, badDryUnitWeight, badSatUnitWeight, badPiezGeometry, badWatUnitWeight
strat	-	coords	-
slopeX	-	$[\mathbb{R}]$	-
slopeY	-	$[\mathbb{R}]$	-
phi	-	$\mathbb{R}$	-
coh	-	$\mathbb{R}$	-
gam	-	$\mathbb{R}$	-
gams	-	$\mathbb{R}$	-
piez	-	coords	-
piezX	-	$[\mathbb{R}]$	-
piezY	-	$[\mathbb{R}]$	-
gamw	-	$\mathbb{R}$	-
xExt	-	$[\mathbb{R}]^{1,2}$	-
xEtr	-	$[\mathbb{R}]^{1,2}$	-
yLim	-	$[\mathbb{R}]^{1,2}$	-
ltor	-	$\mathbb{B}$	-
ftype	-	$\mathbb{B}$	-
evnslc	-	$\mathbb{B}$	-
cncvu	-	$\mathbb{B}$	-
obtu	-	$\mathbb{B}$	-

8.4 Semantics

8.4.1 State Variables

slope : paramsLayers
piez : paramsPiez
search : paramsSearch
soln : paramsSoln

8.4.2 Environment Variables

in_file : String

- *in_file* represents a file stored in the file system of the hardware running SSP.

8.4.3 Assumptions

- load_params is called before any of the other access programs.

8.4.4 Access Routine Semantics

load_params(*fname*):

- transition:

  slope, *piez*, *search*, *soln* := *slope'*, *piez'*, *search'*, *soln'*
  where *slope'*, *piez'*, *search'*, and *soln'* are populated based on the contents of *in_file*.

- exceptions:

  exc := (*fname* does not exist in file system ⇒ fileNotExist
  | *fname*[(|*fname*| - 4)..(|*fname*| - 1)] = ".out" ⇒ badFileExtension
  | *in_file* is not formatted correctly ⇒ unexpectedInput)

verify_params():

- exceptions:

  $exc := (\neg(\forall(i : \mathbb{Z} | i \in [0..|slope.strat| - 2] : slope.strat[i].x - slope.strat[i+1].x \leq 0)) \Rightarrow$ badSlopeGeometry
  | $\neg(0 < slope.phi < 90) \Rightarrow$ badEffAngleFriction
  | $\neg(0 < slope.coh) \Rightarrow$ badCohesion
  | $\neg(0 < slope.gam) \Rightarrow$ badDryUnitWeight
  | $\neg(0 < slope.gams) \Rightarrow$ badSatUnitWeight

$| \neg(\forall(i : \mathbb{Z} | i \in [0..|piez.piez| - 2] : piez.piez[i].x - piez.piez[i+1].x \leq 0))$
$\vee\, piez.piez[0].x \neq slope.strat[0].x$
$\vee\, piez.piez[|piez.piez| - 1].x \neq slope.strat[|slope.strat| - 1].x$
$\Rightarrow$ badPiezGeometry
$| \ (slope.strat[|slope.strat| - 1].x > slope.strat[0].x \Rightarrow \forall(i : \mathbb{Z} | i \in [0..1] :$
$search.Xext[i] > slope.strat[|slope.strat| - 1].x$
$\vee\, search.Xext[i] < slope.strat[0].x$
$\vee\, search.Xetr[i] > slope.strat[|slope.strat| - 1].x$
$\vee\, search.Xetr[i] < slope.strat[0].x)$
$| \ slope.strat[|slope.strat| - 1].x < slope.strat[0].x \Rightarrow \forall(i : \mathbb{Z} | i \in [0..1] :$
$search.Xext[i] < slope.strat[|slope.strat| - 1].x$
$\vee\, search.Xext[i] > slope.strat[0].x$
$\vee\, search.Xetr[i] < slope.strat[|slope.strat| - 1].x$
$\vee\, search.Xetr[i] > slope.strat[0].x))$
$\Rightarrow$ badSlipGuess)

strat():

- output:

$$out := slope.strat$$

slopeX():

- output:

$$out := slope.strat[0].x || slope.strat[1].x || \\ \ldots || slope.strat[|slope.strat| - 1].x$$

slopeY():

- output:

$$out := slope.strat[0].y || slope.strat[1].y || \\ \ldots || slope.strat[|slope.strat| - 1].y$$

phi():

- output:

$$out := slope.phi$$

coh():

- output:

$$out := slope.coh$$

gam():

- output:

$$out := slope.gam$$

gams():

- output:

$$out := slope.gams$$

piez():

- output:

$$out := piez.piez$$

piezX():

- output:

$$out := piez.piez[0].x||piez.piez[1].x|| \\ \ldots||piez.piez[|piez.piez|-1].x$$

piezY():

- output:

$$out := piez.piez[0].y||piez.piez[1].y|| \\ \ldots||piez.piez[|piez.piez|-1].y$$

gamw():

- output:

$$out := piez.gamw$$

xExt():

- output:

$$out := search.Xext$$

xEtr():

- output:

$$out := search.Xetr$$

yLim():

- output:

$$out := search.Ylim$$

ltor():

- output:

$$out := soln.ltor$$

ftype():

- output:

$$out := soln.ftype$$

evnslc():

- output:

$$out := soln.evnslc$$

cncvu():

- output:

$$out := soln.cncvu$$

obtu():

- output:

$$out := soln.obtu$$

8.4.5 Local Functions

N/A

9 MIS of the Genetic Algorithm Module

9.1 Module

GenAlg

9.2 Uses

9.2.1 Imported Access Programs

Input (Section 8), MorgPriceSolver (Section 13), Slicer (Section 12), KinAdm (Section 10), SlipWeighter (Section 11), Sequence (Section 16), Rand (Section 17)

9.3 Syntax

9.3.1 Exported Constants

MIN_GENS = 100
NUM_SLIPS = 20
REL_DIFF = 0.00005
INIT_NUM_VERTICES = 4
NUM_ADDS = 2

9.3.2 Exported Data Types

slip	=	tuple of (surf : coords, Fs : $\mathbb{R}$, G : coords, X : coords, wt : $\mathbb{R}$)
slips	=	[slip]

9.3.3 Exported Access Programs

Name	In	Out	Exceptions
genetic_alg	-	$\mathbb{R}$, coords, coords, coords	-

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

genetic_alg():

- output:

  $out :=$ weighter(*slip_surfs*)[0].*surf*, weighter(*slip_surfs*)[0].*Fs*, weighter(*slip_surfs*)[0].*G*, and weighter(*slip_surfs*)[0].*X*, where *slip_surfs*, of type slips, is developed by:

  * using rand to randomly generate coordinates for NUM_SLIPS potential slip surfaces, where the entry and exit x-coordinate for each slip surface are computed according to *generate_slips(xEtr)* and *generate_slips(xExt)*. Corresponding y-coordinates are determined by interpolating on the slope geometry. The total number of coordinates for each slip surface is INIT_NUM_VERTICES.
  * using kinAdm to verify that the geometry of each potential slip surface is physically realizable. If any are not, new slip surfaces are randomly generated until NUM_SLIPS valid slip surfaces have been generated,
  * using slicer to redefine each slip surface's coordinates based on the desired number of slices
  * using morg_price to determine the *Fs*, *G*, and *X* fields of each slip surface
  * using weighter to determine the *wt* field of each slip surface
  * using rand to generate a new pool of NUM_SLIPS slip surfaces by applying crossovers and mutations to the previous generation, with the more highly-weighted members having a greater likelihood of contributing to the subsequent generations
  * applying kinAdm, slicer, morg_price, and weighter to the new generation
  * repeating the above two steps, but every time $\frac{\text{MIN\_GENS}}{\text{NUM\_ADDS}+1}$ generations have occured, vertices are added to the pool of slip surfaces halfway between each of the existing vertices, so that the new slip surfaces each have $(\text{INIT\_NUM\_VERTICES} * 2) - 1$ vertices.
  * repeating until at least MIN_GENS have occured and the relative difference between subsequent generations is less than REL_DIFF.

9.4.5 Local Functions

generate_slips($Xrange$) : $[\mathbb{R}] \rightarrow \mathbb{R}$
generate_slips($Xrange$) $= (Xrange[0] + rand() * (Xrange[1] - Xrange[0]))$

10 MIS of the Kinematic Admissibility Module

10.1 Module

KinAdm

10.2 Uses

Input (Section 8), Sequence (Section 16)

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Data Types

N/A

10.3.3 Exported Access Programs

Name	In	Out	Exceptions
kinAdm	slip	$\mathbb{B}$	-

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

- The *surf* field is populated for every member of the input sequence of slip data.

10.4.4 Access Routine Semantics

kinAdm(*slip_surf*):

- output:

$out := (\ \neg(\forall(i:\mathbb{Z}|i \in [0..|slip\_surf.surf| - 2] : slip\_surf.surf[i].x - slip\_surf.surf[i+1].x \leq 0))$
$\vee \neg is\_on\_slope(slip\_surf.surf[0])$
$\vee \neg is\_on\_slope(slip\_surf.surf[|slip\_surf.surf - 1|])$
$\vee \neg is\_in\_slope(slip\_surf.surf)$
$\vee (cncvu() \wedge \neg(is\_concave\_up(slip\_surf.surf)))$
$\vee (obtu() \wedge \neg(has\_no\_sharp\_angles(slip\_surf.surf)))$
$\Rightarrow false$
$|else \Rightarrow true)$

[Not sure if I'm allowed to use "else" here but don't know how else to express the "else" case succintly —BM]

10.4.5 Local Functions

linSlope(*point1*, *point2*) : coord × coord → $\mathbb{R}$
linSlope(*point1*, *point2*) $= \frac{point2.y - point1.y}{point2.x - point1.x}$

is_on_slope(*point*) : coord → $\mathbb{B}$
is_on_slope(*point*) $= (\exists(i:\mathbb{Z}|i \in [0..|slope.strat| - 1] : point = slope.strat[i]))$
$\vee (\exists(i:\mathbb{Z}|i \in [0..|slope.strat| - 2] : point.y = linSlope(slope.strat[i], slope.strat[i+1]) * point.x + \frac{slope.strat[i].y}{linSlope(slope.strat[i], slope.strat[i+1]) * slope.strat[i].x}))$

is_in_slope(*surf*) : coords → $\mathbb{B}$
is_in_slope(*surf*) $= (\forall(i:\mathbb{Z}|i \in [1..|surf| - 2] : (\forall(j:\mathbb{Z}|j \in [0..|slope.strat| - 2] \wedge slope.strat[j].x \leq surf[i].x < slope.strat[j+1].x : surf[i].y < (slope.strat[j].y + (surf[i].x - slope.strat[j].x) * linSlope(slope.strat[j], slope.strat[j+1]))))))$

is_concave_up(*surf*) : coords → $\mathbb{B}$
is_concave_up(*surf*) $= (\forall(i:\mathbb{Z}|i \in [0..|surf| - 3] : linSlope(surf[i+1], surf[i+2]) \geq linSlope(surf[i], surf[i+1]))$

distance(*point1*, *point2*) : coord × coord → $\mathbb{R}$
distance(*point1*, *point2*) $= \sqrt{(point1.x - point2.x)^2 + (point1.y - point2.y)^2}$

has_no_sharp_angles(*surf*) : coords → $\mathbb{B}$
has_no_sharp_angles(*surf*) $= (\forall(i:\mathbb{Z}|i \in [0..|surf| - 3] :$
$\arccos \frac{(distance(surf[i], surf[i+1]))^2 + (distance(surf[i+1], surf[i+2]))^2 - (distance(surf[i], surf[i+2]))^2}{2 * distance(surf[i], surf[i+1]) * distance(surf[i+1], surf[i+2])} \geq 1.9199))$

11 MIS of the Slip Weighting Module

11.1 Module

SlipWeighter

11.2 Uses

Sequence (Section 16)

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Data Types

N/A

11.3.3 Exported Access Programs

Name	In	Out	Exceptions
weighter	slips	slips	-

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

- The Fs field is populated for every member of the input sequence of slip data.

11.4.4 Access Routine Semantics

weighter($slip\_surfs$):

- output:

$$out := slip\_surfs\text{'} \text{ such that } slip\_surfs\text{'} = assign\_weights(sort\_Fs(slip\_surfs))$$

11.4.5 Local Functions

sort_Fs($unsorted$) : slips $\rightarrow$ slips sort_Fs($unsorted$) $=$ $sorted$ such that
$\forall(a : slip | a \in unsorted : \exists(b : slip | b \in sorted : b = a \wedge count(a, A) = count(b, B))) \wedge \forall(i : \mathbb{Z} | i \in [0..|unsorted| - 1] : sorted[i].Fs \leq sorted[i+1].Fs)$

count(a, A) : slip $\times$ slips $\rightarrow \mathbb{Z}$
count(a, A) $= +(x : slip | x \in A \wedge x = a : 1)$

assign_weights(s) : slips $\rightarrow$ slips
assign_weights(s) $= s'$ such that
$\frac{s'[0].wt = s[0].Fs - s[|s|-1].Fs}{+(j:\mathbb{Z}|j \in [0..|s|-1]:s[j].Fs - s[|s|-1].Fs)}$ and
$\forall(i : \mathbb{Z} | i \in [1..|s| - 1] : s'[i].wt = s'[i-1].wt + \frac{s[i].Fs - s[|s|-1].Fs}{+(j:\mathbb{Z}|j \in [0..|s|-1]:s[j].Fs - s[|s|-1].Fs)})$

12 MIS of the Slip Slicing Module

12.1 Module

Slicer

12.2 Uses

Sequence (Section 16)

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Data Types

N/A

12.3.3 Exported Access Programs

Name	In	Out	Exceptions
slicer	coords, $\mathbb{Z}$	coords	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumption

- The integer input to *slicer* is greater than the size of the slip input to *slicer*.

12.4.4 Access Routine Semantics

slicer(*slip_surf*, *num_slices*):

- output:

  $out := (soln.evnslc \Rightarrow$ *slip_surf'* obtained by repeatedly applying *slip_surf*[large_segment(*slip_sur*
  || midpoint(*slip_surf*[large_segment(*slip_surf*)], *slip_surf*[large_segment(*slip_surf*)+1])
  || *slip_surf*[large_segment(*slip_surf*)+1] until |*slip_surf'*| = *num_slices*
  $|\neg soln.evnslc \Rightarrow$ *slip_surf'* such that $\forall(i : \mathbb{Z} | i \in [0..|slip\_surf| - 2] : slip\_surf'[i * round\_down(\frac{num\_slices}{|slip\_surf| - 1})..(i+1) * round\_down(\frac{num\_slices}{|slip\_surf| - 1})] = subslice(round\_down(\frac{num\_slices}{|slip\_surf| - 1}), slip\_surf[i], slip\_surf[i+1]))$

12.4.5 Local Functions

large_segment(*surf*) : coords $\rightarrow \mathbb{Z}$
large_segment(*surf*) = *index* such that
$\forall(i : \mathbb{Z} | i \in [0..|surf| - 2] : surf[index + 1] - surf[index] \geq surf[i+1] - surf[i])$

midpoint(*point1*, *point2*) : coord $\times$ coord $\rightarrow$ coord
midpoint(*point1*, *point2*) $= < \frac{point1.x + point2.x}{2}, \frac{point1.y + point2.y}{2} >$

round_down(*num*) : $\mathbb{R} \rightarrow \mathbb{Z}$
round_down(*num*) = *rounded_num* such that
$\forall(i : \mathbb{Z} | i \leq num < i + 1 : rounded\_num = i)$

subslice(*n*, *point1*, *point2*) : $\mathbb{Z} \times$ coord $\times$ coord $\rightarrow$ coords
subslice(*n*, *point1*, *point2*) = *subslices* such that
$\forall(i : \mathbb{Z} | i \in [0..n] : subslices[i].x = point1.x + \frac{i}{n} * (point2.x - point1.x) \wedge subslices[i].y = point1.y + \frac{i}{n} * (point2.y - point1.y))$

13 MIS of the Morgenstern-Price Calculation Module

13.1 Module

MorgPriceSolver

13.2 Uses

Input (Section 8), PropertyCalc (Section 14), Sequence (Section 16)

13.3 Syntax

13.3.1 Exported Constants

MAX_DIFF = 0.000001
MAX_ITER = 20
MIN_FS = 0.5

13.3.2 Exported Data Types

N/A

13.3.3 Exported Access Programs

Name	In	Out	Exceptions
morg_price	slip	slip	-

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

N/A

13.4.4 Access Routine Semantics

morg_price(*slip_surf*):

- output:

  out := *slip_surf'* where *slip_surf'.Fs*, *slip_surf'.G*, and *slip_surf'.X* satisfy the following system of equations, taken from the SRS document. The equations are presented with the symbols from the SRS document for readability, though the symbols F_{S}, G, and X correspond to *slip_surf'.Fs*, *slip_surf'.G*, and *slip_surf'.X*, respectively.

  $$\text{(IM1)}: \; F_{\mathrm{S}} = \frac{\sum_{i=1}^{n-1}\left[R_i \prod_{c=i}^{n-1} \Psi_c\right] + R_n}{\sum_{i=1}^{n-1}\left[T_i \prod_{c=i}^{n-1} \Psi_c\right] + T_n}$$

  $$\text{(IM2)}: \; C_{\mathrm{num},i} = \begin{cases} b_1\left[G_1 + H_1\right]\tan\left(\alpha_1\right) & i = 1 \\ b_i\left[\left(G_i + G_{i-1}\right) + \left(H_i + H_{i-1}\right)\right]\tan\left(\alpha_i\right) \\ \quad + h_i\left(-2\, U_{\mathrm{t},i}\sin\left(\beta_i\right)\right) & 2 \leq i \leq n-1 \\ b_n\left[G_{n-1} + H_{n-1}\right]\tan\left(\alpha_{n-1}\right) & i = n \end{cases}$$

  $$C_{\mathrm{den},i} = \begin{cases} b_1 G_1 f_1 & i = 1 \\ b_i\left(f_i G_i + f_{i-1} G_{i-1}\right) & 2 \leq i \leq n-1 \\ b_n G_{n-1} f_{n-1} & i = n \end{cases}$$

  $$\lambda = \frac{\sum_{i=1}^{n} C_{\mathrm{num},i}}{\sum_{i=1}^{n} C_{\mathrm{den},i}}$$

  $$\text{(IM3)}: \; G_i = \begin{cases} \frac{(F_{\mathrm{S}})T_1 - R_1}{\Phi_i} & i = 1 \\ \frac{\Psi_{i-1}\cdot G_{i-1} + (F_{\mathrm{S}})\cdot T_i - R_i}{\Phi_i} & 2 \leq i \leq n-1 \\ 0 & i = 0 \vee i = n \end{cases}$$

  $$\text{(GD8)}: \; X = \lambda \cdot f \cdot G$$

  The solution method is to start with initial guesses $F_{\mathrm{S}} = 1$ and $\lambda = 0$ and use them to compute F_{S} using IM1 and G using IM3, then use these values to compute a new guess for λ using IM2. This iteration continues until the absolute difference between F_{S} in the current iteration and in the previous iteration is less than MAX_DIFF, or until the absolute difference between λ in the current iteration and in the previous iteration is less than MAX_DIFF. When this occurs, X is computed using GD8, and *slip_surf'*.Fs, *slip_surf'*.G, *slip_surf'*.X $:= F_{\text{S}}, G, X$. If MAX_ITER iterations occur, the solution is considered to be non-converging. If the solution converges but $F_{\text{S}} <$ MIN_FS, the solution is considered to be spurious. In either case, *slip_surf'*.Fs, *slip_surf'*.G, *slip_surf'*.X $:= 1000, [], []$.

13.4.5 Local Functions

N/A

14 MIS of the Slice Property Calculation Module

14.1 Module

PropertyCalc

14.2 Uses

Input (Section 8), Sequence (Section 16)

14.3 Syntax

14.3.1 Exported Constants

N/A

14.3.2 Exported Data Types

paramsInternalForce	=	tuple of (Ub, Ut, W, H : $[\mathbb{R}]$) (Appendix 19.1.5)
paramsAngles	=	tuple of (alpha, beta : $[\mathbb{R}]$) (Appendix 19.1.6)

14.3.3 Exported Access Programs

Name	In	Out	Exceptions
prop_calc	slip	-	-
ub	-	$[\mathbb{R}]$	-
ut	-	$[\mathbb{R}]$	-
w	-	$[\mathbb{R}]$	-
h	-	$[\mathbb{R}]$	-
alpha	-	$[\mathbb{R}]$	-
beta	-	$[\mathbb{R}]$	-
hts	-	$[\mathbb{R}]$	-

14.4 Semantics

14.4.1 State Variables

$force$: paramsInternalForce
$angles$: paramsAngles
$heights$: $[\mathbb{R}]$

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

- prop_calc is called before any of the other access programs.

14.4.4 Access Routine Semantics

prop_calc($slip\_surf$):

- transition:

  The equations used below contain symbols from the SRS document for this project for the sake of brevity. The SRS should be consulted for the definitions of these symbols.

$force, angles, heights :=$ *force', angles', soil'heights'* where
$\forall(i : \mathbb{Z} | i \in [1..|slip\_surf| - 1] :$
$$force.Ub[i] = \ell_{\mathrm{b},i} \begin{cases} (y_{\mathrm{wt},i} - y_{\mathrm{slip},i})\,\gamma_{\mathrm{w}} & y_{\mathrm{wt},i} > y_{\mathrm{slip},i} \\ 0 & y_{\mathrm{wt},i} \leq y_{\mathrm{slip},i} \end{cases}$$
$$\wedge\, force.Ut[i] = \ell_{\mathrm{s},i} \begin{cases} (y_{\mathrm{wt},i} - y_{\mathrm{us},i})\,\gamma_{\mathrm{w}} & y_{\mathrm{wt},i} > y_{\mathrm{us},i} \\ 0 & y_{\mathrm{wt},i} \leq y_{\mathrm{us},i} \end{cases}$$
$$\wedge\, force.W[i] = b_i \begin{cases} (y_{\mathrm{us},i} - y_{\mathrm{slip},i})\,\gamma_{\mathrm{Sat}} & y_{\mathrm{wt},i} \geq y_{\mathrm{us},i} \\ (y_{\mathrm{us},i} - y_{\mathrm{wt},i})\,\gamma + (y_{\mathrm{wt},i} - y_{\mathrm{slip},i})\,\gamma_{\mathrm{Sat}} & y_{\mathrm{us},i} > y_{\mathrm{wt},i} > y_{\mathrm{slip},i} \\ (y_{\mathrm{us},i} - y_{\mathrm{slip},i})\,\gamma & y_{\mathrm{wt},i} \leq y_{\mathrm{slip},i} \end{cases}$$
$$\wedge\, force.H[i] = \begin{cases} \frac{[y_{\mathrm{us},i} - y_{\mathrm{slip},i}]^2}{2}\gamma_{\mathrm{w}} + [y_{\mathrm{wt},i} - y_{\mathrm{us},i}]^2\,\gamma_{\mathrm{w}} & y_{\mathrm{wt},i} \geq y_{\mathrm{us},i} \\ \frac{[y_{\mathrm{wt},i} - y_{\mathrm{slip},i}]^2}{2}\gamma_{\mathrm{w}} & y_{\mathrm{us},i} > y_{\mathrm{wt},i} > y_{\mathrm{slip},i} \\ 0 & y_{\mathrm{wt},i} \leq y_{\mathrm{slip},i} \end{cases}$$
$$\wedge\, angles.alpha[i] = \arctan\left(\frac{y_{\mathrm{slip},i} - y_{\mathrm{slip},i-1}}{x_{\mathrm{slip},i} - x_{\mathrm{slip},i-1}}\right)$$
$$\wedge\, angles.beta[i] = \arctan\left(\frac{y_{\mathrm{us},i} - y_{\mathrm{us},i-1}}{x_{\mathrm{us},i} - x_{\mathrm{us},i-1}}\right)$$
$$\wedge\, heights[i] = 0.5 * ((y_{\mathrm{us},i} - y_{\mathrm{slip},i}) + (y_{\mathrm{us},i+1} - y_{\mathrm{slip},i+1})))$$

ub():

- output:

  $out := force.Ub$

ut():

- output:

  $out := force.Ut$

w():

- output:

  $out := force.W$

h():

- output:

  $out := force.H$

alpha():

- output:

$$out := angles.alpha$$

beta():

- output:

$$out := angles.beta$$

hts():

- output:

$$out := heights$$

14.4.5 Local Functions

N/A

15 MIS of the Output Module

15.1 Module

Output

15.2 Uses

Sequence (Section 16), Plot (Section 18)

15.3 Syntax

15.3.1 Exported Constants

N/A

15.3.2 Exported Data Types

N/A

15.3.3 Exported Access Programs

Name	In	Out	Exceptions
verify_output	$\mathbb{R}$	-	negativeFS
output	$\mathbb{R}$, coords, coords, coords, string	-	-

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 Environment Variables

out_file : String

- *out_file* represents a file stored in the file system of the hardware running SSP.

screen : $[\mathbb{Z}]$

- *screen* represents the colour values for each pixel on the screen of the hardware running SSP.

15.4.3 Assumptions

N/A

15.4.4 Access Routine Semantics

verify_output(Fs):

- exceptions:

$$exc := Fs < 0 \Rightarrow \text{negativeFS}$$

output(Fs, $crit\_slip$, G, X, $fname$):

- transition:

  out_file is created at path $fname$ || ".out". The outputs of xEtr(), xExt(), yLim(), ftype(), Fs, $crit\_slip$, G, and X are written to *out_file*. *screen* is modified to display the outputs of plot($crit\_slip.x$, $crit\_slip.y$), plot($G.x$, $G.y$), and plot($X.x$, $X.y$).

15.4.5 Local Functions

N/A

16 MIS of the Sequence Data Structure Module

16.1 Module

Sequence

16.2 Uses

N/A

16.3 Syntax

16.3.1 Exported Constants

N/A

16.3.2 Exported Data Types

[T] = sequence of T, where T is any type

16.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
[_]	Any number of values of type T	[T]	-
_(_)	[T], $\mathbb{Z}$	T	
_(_:_)	[T], $\mathbb{Z}$, $\mathbb{Z}$	[T]	-

16.4 Semantics

16.4.1 State Variables

N/A

16.4.2 Environment Variables

N/A

16.4.3 Assumptions

N/A

16.4.4 Access Routine Semantics

[_](Any number of values):

- output:

  $out :=$ A sequence containing the arguments passed to the function.

_(_)($list$, int):

- output:

  $out := list[int]$

_(_:_)($list$, $int1$, $int2$):

- output:

  $out := list[int1..int2]$

16.4.5 Local Functions

N/A

16.4.6 Considerations

This module is the sequence data type and operations on sequences implemented by Matlab.

17 MIS of the Random Number Generation Module

17.1 Module

Rand

17.2 Uses

N/A

17.3 Syntax

17.3.1 Exported Constants

N/A

17.3.2 Exported Data Types

N/A

17.3.3 Exported Access Programs

Name	In	Out	Exceptions
rand	-	$\mathbb{R}$	-

17.4 Semantics

17.4.1 State Variables

N/A

17.4.2 Environment Variables

N/A

17.4.3 Assumptions

N/A

17.4.4 Access Routine Semantics

rand():

- output:

  $out :=$ A random number in the interval (0,1).

17.4.5 Local Functions

N/A

17.4.6 Considerations

This module is the rand function implemented by Matlab.

18 MIS of the Plotting Module

18.1 Module

Plot

18.2 Uses

N/A

18.3 Syntax

18.3.1 Exported Constants

N/A

18.3.2 Exported Data Types

N/A

18.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
plot	$[\mathbb{R}]$, $[\mathbb{R}]$	-	-

18.4 Semantics

18.4.1 State Variables

N/A

18.4.2 Environment Variables

$screen : [\mathbb{Z}]$

- *screen* represents the colour values for each pixel on the screen of the hardware running SSP.

18.4.3 Assumptions

N/A

18.4.4 Access Routine Semantics

plot(x, y):

- transition:

  Modifies *screen* to display a plot with x on the horizontal axis and y on the vertical axis.

18.4.5 Local Functions

N/A

18.4.6 Considerations

This module is the plot function implemented by Matlab.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

David Gries and Fred B. Schneider. *A logical approach to discrete math*. Springer-Verlag Inc., New York, 1993.

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.

19 Appendix

19.1 Parameter Tables

19.1.1 Layer Parameters

The elements in the paramsLayers structure, which describe the mass of soil on which slope stability analysis is to be performed, are explained in the table below.

Parameter	Description
$strat$: coords	Coordinates describing the vertices of the slope of soil.
phi : $\mathbb{R}$	The effective angle of friction of the soil.
coh : $\mathbb{R}$	The effective cohesion of the soil.
gam : $\mathbb{R}$	The dry unit weight of the soil.
$gams$: $\mathbb{R}$	The saturated unit weight of the soil.

19.1.2 Piezometric Parameter

The elements in the paramsPiez structure, which describe the water table, are explained in the table below.

Parameter	Description
$piez$: $coords$	Coordinates describing the vertices of the water table. If there is no water table than $piez$ is an empty array.
$gamw$: $\mathbb{R}$	The unit weight of water.

19.1.3 Search Range Parameters

The elements in the paramsSearch structure, which are parameters relating to the range of coordinates between which the critical slip surface may exist, are described in the table below.

Parameter	Description
Xext : $[\mathbb{R}]^{1,2}$	The range of x-ordinates between which the exit point of the critical slip surface may exist. Exit refers to the point of the slip at lower elevation toward which the mass of soil will move during failure.
Xetr : $[\mathbb{R}]^{1,2}$	The range of x-ordinates between which the entry point of the critical slip surface may exist. Entry refers to the point of the slip at higher elevation away from which the mass of soil will move during failure.
Ylim : $[\mathbb{R}]^{1,2}$	The range of y-ordinates between which the critical slip surface may exist. The larger value should be greater than the max y-ordinate of the slope. The smaller value is the lowest elevation to which the critical slip surface may descend.

19.1.4 Solution Parameters

The elements in the paramsSoln structure, which are parameters relating to the solution method, are described in the table below.

Parameter	Description
ltor : $\mathbb{B}$	Direction the slope is expected to experience failure in. If true then the side of the slope with a greater x-ordinate value is at a lower elevation. If false then the side of the slope with a greater x-ordinate is at a higher elevation.
ftype : $\mathbb{B}$	Function to use for interslice normal/shear force ratio variation function. If true then the function is a constant (Spencer's method). If false then the function is a half-sine (standard Morgenstern-Price method).
evnslc : $\mathbb{B}$	Method for slicing a slip surface prior to analysis. If true then slice slip surface into equal x-ordinate widths. If false then slice distance between vertices into even number of slices.
cncvu : $\mathbb{B}$	Concave slip surface admissibility criterion. If true then an admissible slip surface must be concave upwards towards the surface. If false then an admissible slip surface does not need to pass this criterion.
obtu : $\mathbb{B}$	Angle slip surface admissibility criterion. If true then an admissible slip surface must have all interior angles greater than a set limit. If false then an admissible slip surface does not need to pass this criterion.

19.1.5 Internal Force Parameters

The elements in the paramsInternalForce structure, which are parameters relating to the forces acting on a slice, and water in the slope acting on itself, are described in the table below. n refers to the number of slices composing the slip surface under evaluation, and is defined by the Slicer module (section 12).

Parameter	Description
Ub : $[\mathbb{R}]^{1,n}$	Sequence of the force acting on the basal surface of a slice as a result of pore water pressure within the slice. From DD2 of the SRS.
Ut : $[\mathbb{R}]^{1,n}$	Sequence of the force acting on the upper surface of a slice as a result of pore water pressure from standing water on the surface. From DD3 of the SRS.
W : $[\mathbb{R}]^{1,n}$	Sequence of the downward force acting on the slice caused by the mass of the slice and the force of gravity. From DD1 of the SRS.
H : $[\mathbb{R}]^{1,n-1}$	Sequence of the force acting into the interslice surfaces as a result of pore water pressure within the adjacent slices. From DD4 of the SRS.

19.1.6 Angle Parameters

The elements in the paramsAngles structure, which are parameters relating to the angles of the slice surfaces. n refers to the number of slices composing the slip surface under evaluation, and is defined by the Slicer module (section 12).

Parameter	Description
alpha : $[\mathbb{R}]^{1,n}$	Sequence of the angle that the basal surface of the slice makes with the horizontal. From DD5 of the SRS.
beta : $[\mathbb{R}]^{1,n}$	Sequence of the angle that the upper surface of the slice makes with the horizontal. From DD6 of the SRS.