

# Dikes Overtopping Kernel - Technical documentation

Generated by Doxygen 1.8.9.1

Fri Mar 25 2016 10:14:01



# Contents

<b>1</b>	<b>Modules Index</b>	<b>1</b>
1.1	Modules List . . . . .	1
<b>2</b>	<b>Data Type Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	dllovertopping Module Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Function/Subroutine Documentation . . . . .	8
4.1.2.1	calculateqo . . . . .	8
4.1.2.2	calculateqof . . . . .	9
4.1.2.3	calcvalue . . . . .	10
4.1.2.4	geometry_c_f . . . . .	11
4.1.2.5	getlanguage . . . . .	11
4.1.2.6	setlanguage . . . . .	11
4.1.2.7	validateinputc . . . . .	11
4.1.2.8	validateinputf . . . . .	12
4.1.2.9	versionnumber . . . . .	13
4.2	factormodulertooverlapping Module Reference . . . . .	13
4.2.1	Function/Subroutine Documentation . . . . .	13
4.2.1.1	calculategammab . . . . .	13
4.2.1.2	calculategammabeta . . . . .	14
4.2.1.3	calculategammaf . . . . .	14
4.2.1.4	calculatetanalpha . . . . .	15
4.3	formulamodulertooverlapping Module Reference . . . . .	16
4.3.1	Function/Subroutine Documentation . . . . .	17
4.3.1.1	adjustinfluencefactors . . . . .	17
4.3.1.2	calculateanglewaveattack . . . . .	17

4.3.1.3	calculatebreakerlimit	18
4.3.1.4	calculatebreakerparameter	18
4.3.1.5	calculatewavelength	19
4.3.1.6	calculatewaveovertoppingdischarge	19
4.3.1.7	calculatewaverunup	20
4.3.1.8	calculatewavesteepness	21
4.3.1.9	cubicroots	21
4.3.1.10	isequalreal	22
4.3.1.11	isequalzero	22
4.3.1.12	realrootscubicfunction	22
4.3.1.13	rootsdepressedcubic	23
4.3.1.14	rootsgeneralcubic	24
4.4	geometrymodulertoovertopping Module Reference	25
4.4.1	Function/Subroutine Documentation	26
4.4.1.1	adjustnonhorizontalberms	26
4.4.1.2	allocatevectorsgeometry	26
4.4.1.3	basicgeometrytest	27
4.4.1.4	calculatehorzdistance	28
4.4.1.5	calculatehorzlengths	28
4.4.1.6	calculatesegmentslopes	29
4.4.1.7	checkcrosssection	30
4.4.1.8	copygeometry	30
4.4.1.9	deallocategeometry	31
4.4.1.10	determinesegmenttypes	31
4.4.1.11	initializegeometry	32
4.4.1.12	mergesquentialberms	33
4.4.1.13	removeberms	34
4.4.1.14	removedikesegments	35
4.4.1.15	splitcrosssection	35
4.5	mainmodulertoovertopping Module Reference	36
4.5.1	Function/Subroutine Documentation	37
4.5.1.1	calculateovertopping	37
4.5.1.2	calculateovertoppingnegativefreeboard	37
4.5.1.3	calculateovertoppingsection	38
4.5.1.4	calculatewaveovertopping	39
4.5.1.5	checkinputdata	40
4.5.1.6	checkmodelfactors	41
4.5.1.7	converttoovertoppinginput	41
4.5.1.8	interpolateresultssections	42
4.6	modulelogging Module Reference	43

4.6.1	Variable Documentation	43
4.6.1.1	currentlogging	43
4.6.1.2	maxfilenamelength	43
4.7	overtoppinginterface Module Reference	43
4.7.1	Variable Documentation	44
4.7.1.1	varmodelfactorcriticalovertopping	44
4.8	overtoppingmessages Module Reference	44
4.8.1	Function/Subroutine Documentation	44
4.8.1.1	getlanguage	44
4.8.1.2	getovertoppingformat	44
4.8.1.3	getovertoppingmessage	45
4.8.1.4	getovertoppingparameter	45
4.8.1.5	setlanguage	46
4.8.2	Variable Documentation	46
4.8.2.1	language	46
4.8.2.2	maxmsg	46
4.8.2.3	maxpar	46
4.9	typedefinitionsrtooverlapping Module Reference	46
4.9.1	Variable Documentation	48
4.9.1.1	berm_max	48
4.9.1.2	berm_min	48
4.9.1.3	fb_max	48
4.9.1.4	fb_min	48
4.9.1.5	fn_max	48
4.9.1.6	fn_min	48
4.9.1.7	foreshore_max	48
4.9.1.8	foreshore_min	48
4.9.1.9	frunup1_max	48
4.9.1.10	frunup1_min	49
4.9.1.11	frunup2_max	49
4.9.1.12	frunup2_min	49
4.9.1.13	frunup3_max	49
4.9.1.14	frunup3_min	49
4.9.1.15	fs_max	49
4.9.1.16	fs_min	49
4.9.1.17	margindiff	49
4.9.1.18	margingrad	49
4.9.1.19	mz2_max	50
4.9.1.20	mz2_min	50
4.9.1.21	rfactor_max	50

4.9.1.22	<a href="#">rfactor_min</a>	50
4.9.1.23	<a href="#">slope_max</a>	50
4.9.1.24	<a href="#">slope_min</a>	50
4.9.1.25	<a href="#">xdiff_min</a>	50
4.9.1.26	<a href="#">z2_iter_max1</a>	50
4.9.1.27	<a href="#">z2_iter_max2</a>	50
4.9.1.28	<a href="#">z2_margin</a>	51
4.10	<a href="#">waverunup Module Reference</a>	51
4.10.1	<a href="#">Function/Subroutine Documentation</a>	51
4.10.1.1	<a href="#">convergedwithresidu</a>	51
4.10.1.2	<a href="#">determinestartingvalue</a>	51
4.10.1.3	<a href="#">findsmallestresidu</a>	52
4.10.1.4	<a href="#">innercalculation</a>	52
4.10.1.5	<a href="#">iterationwaverunup</a>	53
4.11	<a href="#">zfunctionswtiovertopping Module Reference</a>	53
4.11.1	<a href="#">Detailed Description</a>	54
4.11.2	<a href="#">Function/Subroutine Documentation</a>	54
4.11.2.1	<a href="#">adjustprofile</a>	54
4.11.2.2	<a href="#">calculateqorto</a>	55
4.11.2.3	<a href="#">profileinstructure</a>	56
4.11.2.4	<a href="#">zfunclogratios</a>	56
<b>5</b>	<b>Data Type Documentation</b>	<b>59</b>
5.1	<a href="#">overtoppinginterface::overtoppinggeometrytype Type Reference</a>	59
5.1.1	<a href="#">Detailed Description</a>	60
5.1.2	<a href="#">Member Data Documentation</a>	60
5.1.2.1	<a href="#">normal</a>	60
5.1.2.2	<a href="#">npoints</a>	60
5.1.2.3	<a href="#">roughness</a>	60
5.1.2.4	<a href="#">xcoords</a>	60
5.1.2.5	<a href="#">ycoords</a>	60
5.2	<a href="#">overtoppinginterface::overtoppinggeometrytypef Type Reference</a>	61
5.2.1	<a href="#">Detailed Description</a>	61
5.2.2	<a href="#">Member Data Documentation</a>	61
5.2.2.1	<a href="#">normal</a>	61
5.2.2.2	<a href="#">npoints</a>	61
5.2.2.3	<a href="#">roughness</a>	62
5.2.2.4	<a href="#">xcoords</a>	62
5.2.2.5	<a href="#">ycoords</a>	62
5.3	<a href="#">modulelogging::tlogging Type Reference</a>	62

5.3.1	Detailed Description	62
5.3.2	Member Data Documentation	63
5.3.2.1	filename	63
5.3.2.2	verbosity	63
5.4	tpdefinitionsrtoovtopping::tpgeometry Type Reference	63
5.4.1	Detailed Description	64
5.4.2	Member Data Documentation	64
5.4.2.1	nbermsements	64
5.4.2.2	ncoordinates	64
5.4.2.3	psi	64
5.4.2.4	roughnessfactors	64
5.4.2.5	segmentslopes	65
5.4.2.6	segmenttypes	65
5.4.2.7	xcoorddiff	65
5.4.2.8	xcoordinates	65
5.4.2.9	ycoorddiff	65
5.4.2.10	ycoordinates	65
5.5	tpdefinitionsrtoovtopping::tpload Type Reference	65
5.5.1	Detailed Description	66
5.5.2	Member Data Documentation	66
5.5.2.1	h	66
5.5.2.2	hm0	67
5.5.2.3	phi	67
5.5.2.4	tm_10	67
5.6	tpdefinitionsrtoovtopping::tpovertopping Type Reference	67
5.6.1	Detailed Description	68
5.6.2	Member Data Documentation	68
5.6.2.1	qo	68
5.6.2.2	z2	68
5.7	tpdefinitionsrtoovtopping::tpovertoppinginput Type Reference	68
5.7.1	Detailed Description	70
5.7.2	Member Data Documentation	70
5.7.2.1	computedovertopping	70
5.7.2.2	criticalovertopping	70
5.7.2.3	factordeterminationq_b_f_b	70
5.7.2.4	factordeterminationq_b_f_n	70
5.7.2.5	frunup1	70
5.7.2.6	frunup2	70
5.7.2.7	frunup3	71
5.7.2.8	fshallow	71

5.7.2.9	m_z2	71
5.7.2.10	reductionfactorforeshore	71
5.7.2.11	relaxationfactor	71
5.7.2.12	typerunup	71
5.8	overtoppinginterface::tpprofilecoordinate Type Reference	72
5.8.1	Detailed Description	72
5.8.2	Member Data Documentation	72
5.8.2.1	roughness	72
5.8.2.2	xcoordinate	73
5.8.2.3	zcoordinate	73
<b>6</b>	<b>File Documentation</b>	<b>75</b>
6.1	dllOvertopping.f90 File Reference	75
6.1.1	Detailed Description	76
6.2	factorModuleRTOovertopping.f90 File Reference	76
6.2.1	Detailed Description	76
6.3	formulaModuleRTOovertopping.f90 File Reference	76
6.3.1	Detailed Description	77
6.4	geometryModuleRTOovertopping.f90 File Reference	77
6.4.1	Detailed Description	78
6.5	mainModuleRTOovertopping.f90 File Reference	79
6.5.1	Detailed Description	79
6.6	ModuleLogging.f90 File Reference	79
6.6.1	Detailed Description	80
6.7	overtoppingInterface.f90 File Reference	80
6.7.1	Detailed Description	80
6.8	OvertoppingMessages.f90 File Reference	80
6.8.1	Detailed Description	81
6.9	typeDefinitionsRTOovertopping.f90 File Reference	81
6.9.1	Detailed Description	83
6.10	waveRunup.f90 File Reference	83
6.10.1	Detailed Description	83
6.11	zFunctionsWTIOovertopping.f90 File Reference	83
6.11.1	Detailed Description	84
<b>Index</b>		<b>85</b>



# Chapter 1

## Modules Index

### 1.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">dllovertopping</a>	
Calculate one type of overtopping . . . . .	7
<a href="#">factormodulertoovertopping</a> . . . . .	13
<a href="#">formulamodulertoovertopping</a> . . . . .	16
<a href="#">geometrymodulertoovertopping</a> . . . . .	25
<a href="#">mainmodulertoovertopping</a> . . . . .	36
<a href="#">modulelogging</a> . . . . .	43
<a href="#">overtoppinginterface</a> . . . . .	43
<a href="#">overtoppingmessages</a> . . . . .	44
<a href="#">typedefinitionsrtoovertopping</a> . . . . .	46
<a href="#">waverunup</a> . . . . .	51
<a href="#">zfunctionswtiovertopping</a>	
Module for the Limit State Functions (Z-functions) for wave overtopping . . . . .	53



## Chapter 2

# Data Type Index

### 2.1 Class List

Here are the data types with brief descriptions:

<a href="#">overtoppinginterface::overtoppinggeometrytype</a>	59
<a href="#">overtoppinginterface::overtoppinggeometrytypef</a>	61
<a href="#">modulelogging::tlogging</a>	
TLogging: structure for steering the logging	62
<a href="#">typedefinitionsrtoovertopping::tpgeometry</a>	
TpGeometry: structure with geometry data	63
<a href="#">typedefinitionsrtoovertopping::tpload</a>	
TpLoad: structure with load parameters	65
<a href="#">typedefinitionsrtoovertopping::tpovertopping</a>	
TpOvertopping: structure with overtopping results	67
<a href="#">typedefinitionsrtoovertopping::tpovertoppinginput</a>	
OvertoppingModelFactors: C-structure with model factors	68
<a href="#">overtoppinginterface::tpprofilecoordinate</a>	72



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">dllOvertopping.f90</a>	Main entry for the dll DikesOvertopping FUNCTIONS/SUBROUTINES exported from dll↔ Overtopping.dll: . . . . .	75
<a href="#">factorModuleRTOvertopping.f90</a>	This file contains a module with functions for the slope angle and influence factors . . . . .	76
<a href="#">formulaModuleRTOvertopping.f90</a>	This file contains a module with the core computations for Dikes Overtopping . . . . .	76
<a href="#">geometryModuleRTOvertopping.f90</a>	This file contains a module with the core computations for Dikes Overtopping related to the geometry . . . . .	77
<a href="#">mainModuleRTOvertopping.f90</a>	This file contains a module with the core computations for Dikes Overtopping . . . . .	79
<a href="#">ModuleLogging.f90</a>	Module for steering the extra logging . . . . .	79
<a href="#">overtoppingInterface.f90</a>	This file contains the parameters and types (structs) as part of the interface to and from dll↔ Overtopping . . . . .	80
<a href="#">OvertoppingMessages.f90</a>	This file contains the messages in the overtopping dll, in Dutch or English . . . . .	80
<a href="#">typeDefinitionsRTOvertopping.f90</a>	This file contains a module with the type definitions for Dikes Overtopping . . . . .	81
<a href="#">waveRunup.f90</a>	This file contains a module with the core computations for Dikes Overtopping . . . . .	83
<a href="#">zFunctionsWTIOvertopping.f90</a>	This file contains the limit state functions for wave overtopping within WTl . . . . .	83



## Chapter 4

# Module Documentation

### 4.1 dllovertopping Module Reference

Calculate one type of overtopping.

#### Functions/Subroutines

- subroutine, public [calculateqo](#) (load, geometryInput, dikeHeight, modelFactors, overtopping, success, error↔Text, verbosity, logFile)  
*Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF↔: convert C-like input structures to Fortran input structures.*
- subroutine, public [calculateqof](#) (load, geometryF, dikeHeight, modelFactors, overtopping, success, errorText, logging)  
*Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.*
- subroutine, public [calczvalue](#) (criticalOvertoppingRate, modelFactors, Qo, z, success, errorMessage)  
*Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.*
- subroutine, public [validateinputc](#) (geometryInput, dikeHeight, modelFactors, success, errorText)  
*Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.*
- subroutine, public [validateinputf](#) (geometryF, dikeHeight, modelFactors, errorStruct)  
*Subroutine that validates the geometry.*
- subroutine, public [setlanguage](#) (lang)  
*Subroutine that sets the language for error and validation messages.*
- subroutine, public [getlanguage](#) (lang)  
*Subroutine that gets the language for error and validation messages.*
- subroutine, public [versionnumber](#) (version)  
*Subroutine that delivers the version number.*
- type(overtoppinggeometrytypef) function [geometry\\_c\\_f](#) (geometryInput)  
*Private subroutine that converts geometry from c-pointer to fortran struct.*

#### 4.1.1 Detailed Description

Calculate one type of overtopping.

## 4.1.2 Function/Subroutine Documentation

4.1.2.1 subroutine, public dllovertopping::calculateqo ( type(tpload), intent(in) *load*, type(overtoppinggeometrytype), intent(in) *geometryInput*, real(kind=wp), intent(in) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, type (tpovertopping), intent(out) *overtopping*, logical, intent(out) *success*, character(len=\*), intent(out) *errorText*, integer, intent(in) *verbosity*, character(len=\*), intent(in) *logFile* )

Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF: convert C-like input structures to Fortran input structures.

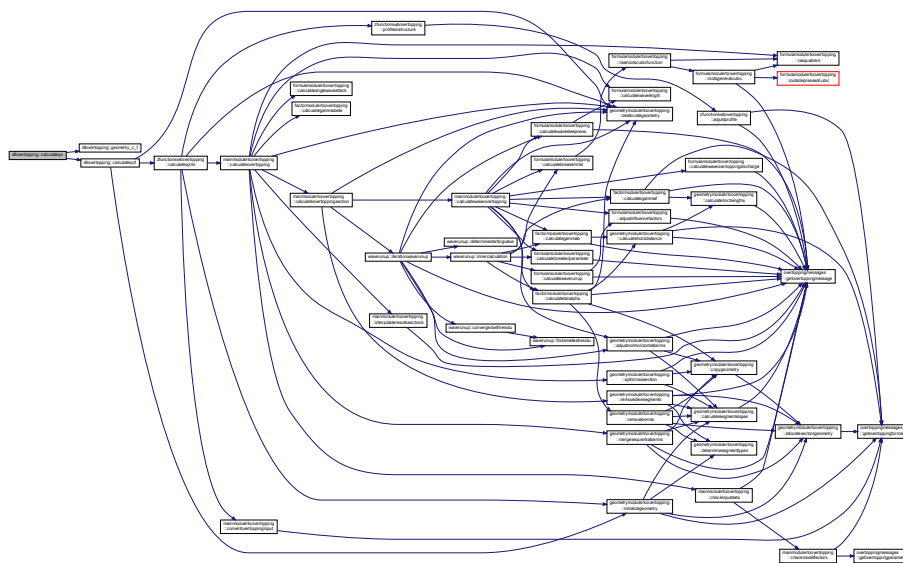


## Parameters

in	<i>geometryinput</i>	struct with geometry and roughness as c-pointers
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelfactors
out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success
out	<i>errortext</i>	error message (only set if not successful)
in	<i>verbosity</i>	level of verbosity
in	<i>logfile</i>	filename of logfile

Definition at line 43 of file dllovertopping.f90.

Here is the call graph for this function:



4.1.2.2 subroutine, public dllovertopping::calculateqof ( type(tpload), intent(in) *load*, type(overtoppinggeometrytypef), intent(in) *geometryF*, real(kind=wp), intent(in) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, type(tpovertopping), intent(out) *overtopping*, logical, intent(out) *success*, character(len=\*) , intent(out) *errorText*, type(tlogging), intent(in) *logging* )

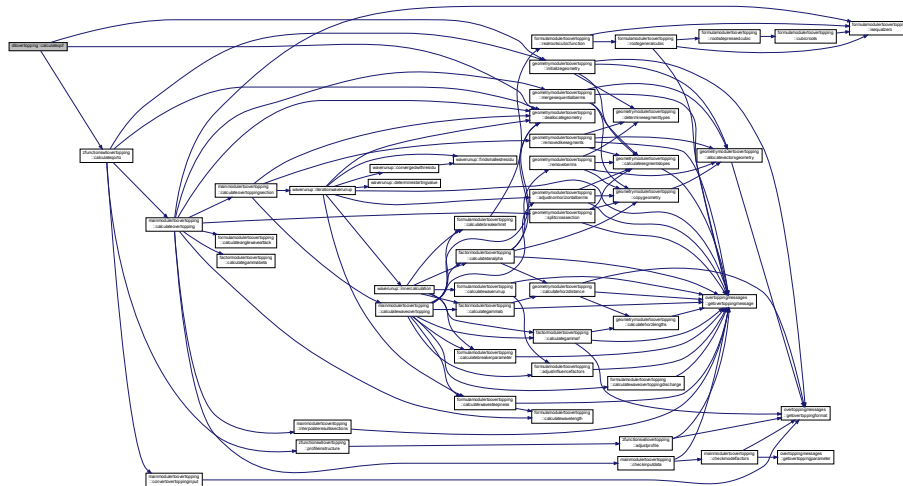
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.

## Parameters

in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelFactors
out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success
out	<i>errortext</i>	error message (only set if not successful)
in	<i>logging</i>	logging struct

Definition at line 74 of file dllovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.1.2.3** subroutine, public dllovertopping::calcvalue ( real(kind=wp), intent(in) *criticalOvertoppingRate*, type(tpovertoppinginput), intent(inout) *modelFactors*, real(kind=wp), intent(in) *Qo*, real(kind=wp), intent(out) *z*, logical, intent(out) *success*, character(len=\*) , intent(out) *errorMessage* )

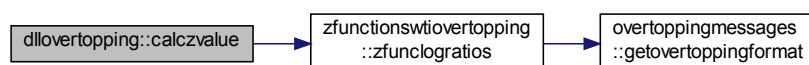
Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.

#### Parameters

in	<i>criticalovertoppingrate</i>	critical overtoppingrate
in, out	<i>modelfactors</i>	struct with modelfactors
in	<i>qo</i>	calculated discharge
out	<i>z</i>	z value
out	<i>errorMessage</i>	error message (only if not successful)
out	<i>success</i>	flag for success

Definition at line 107 of file dllovertopping.f90.

Here is the call graph for this function:



#### 4.1.2.4 `type(overtoppinggeometrytype) function dllovertopping::geometry_c_f ( type(overtoppinggeometrytype), intent(in) geometryInput ) [private]`

Private subroutine that converts geometry from c-pointer to fortran struct.

##### Parameters

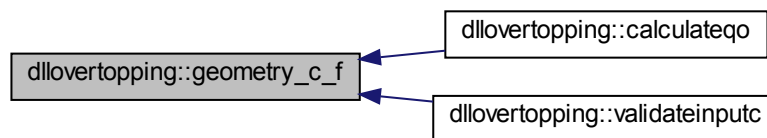
in	<i>geometryinput</i>	struct with geometry and roughness as c-pointers
----	----------------------	--------------------------------------------------

##### Returns

fortran struct with geometry and roughness

Definition at line 310 of file dllovertopping.f90.

Here is the caller graph for this function:



#### 4.1.2.5 `subroutine, public dllovertopping::getlanguage ( character(len=*), intent(out) lang )`

Subroutine that gets the language for error and validation messages.

Definition at line 276 of file dllovertopping.f90.

#### 4.1.2.6 `subroutine, public dllovertopping::setlanguage ( character(len=*), intent(in) lang )`

Subroutine that sets the language for error and validation messages.

Definition at line 263 of file dllovertopping.f90.

#### 4.1.2.7 `subroutine, public dllovertopping::validateinputc ( type(overtoppinggeometrytype), intent(in) geometryInput, real(kind=wp), intent(in) dikeHeight, type(tpovertoppinginput), intent(inout) modelFactors, logical, intent(out) success, character(len=*), intent(out) errorText )`

Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.

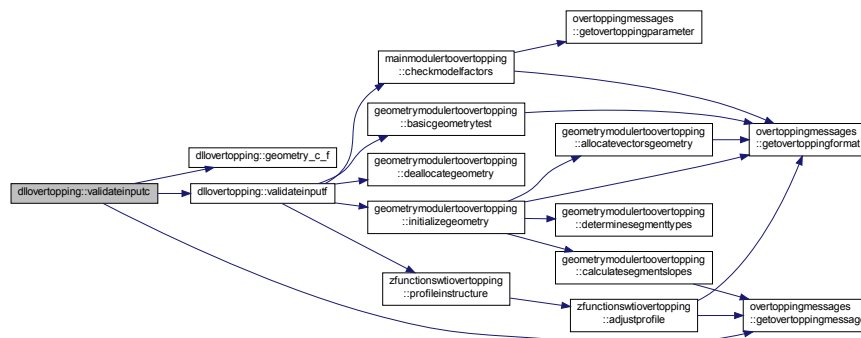
##### Parameters

in	<i>geometryinput</i>	struct with geometry and roughness as c-pointers
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelfactors
out	<i>success</i>	flag for success

out	<i>errortext</i>	error message (only set if not successful)
-----	------------------	--------------------------------------------

Definition at line 127 of file dllOvertopping.f90.

Here is the call graph for this function:



**4.1.2.8** subroutine, public dlovertopping::validateinputf ( type(overtoppinggeometrytypef), intent(in) *geometryF*, real(kind=wp), intent(in) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, type(terrormessages), intent(inout) *errorStruct* )

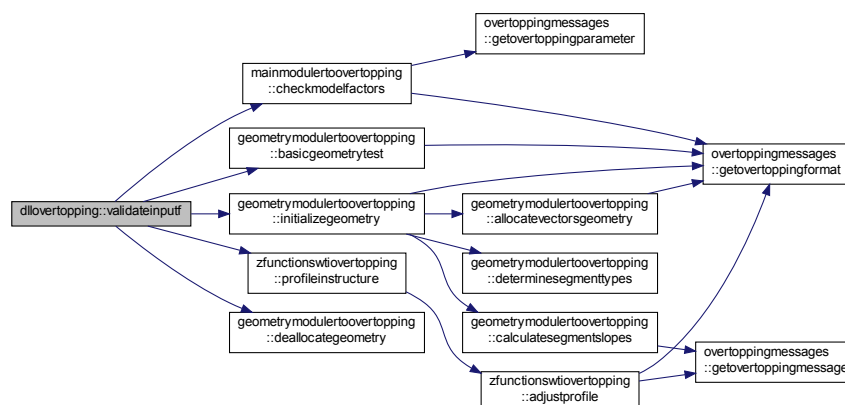
Subroutine that validates the geometry.

#### Parameters

in	<i>geometryf</i>	struct with geometry and roughness
in	<i>dikeheight</i>	dike height
in,out	<i>modelFactors</i>	struct with modelFactors
in,out	<i>errorstruct</i>	error message (only set if not successful)

Definition at line 178 of file dllOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.2.9 subroutine, public dllovertopping::versionnumber ( character(len=\*), intent(out) version )

Subroutine that delivers the version number.

Parameters

out	version	version number
-----	---------	----------------

Definition at line 288 of file dllovertopping.f90.

## 4.2 factormodulertoovertopping Module Reference

### Functions/Subroutines

- subroutine, public [calculatetanalpha](#) (h, Hm0, z2, geometry, tanAlpha, succes, errorMessage)  
*calculateTanAlpha representative slope angle*
- subroutine, public [calculategammabeta](#) (Hm0, Tm\_10, beta, gammaBeta\_z, gammaBeta\_o)  
*calculateGammaBeta influence factor angle of wave attack*
- subroutine, public [calculategammaf](#) (h, ksi0, ksiOLimit, gammaB, z2, geometry, gammaF, succes, error↔Message)  
*calculateGammaF influence factor roughness*
- subroutine, public [calculategammab](#) (h, Hm0, z2, geometry, gammaB, succes, errorMessage)  
*calculateGammaB influence factor berms*

### 4.2.1 Function/Subroutine Documentation

- #### 4.2.1.1 subroutine, public factormodulertoovertopping::calculategammab ( real(wp), intent(in) h, real(wp), intent(in) Hm0, real(wp), intent(in) z2, type(tpgeometry), intent(in) geometry, real(wp), intent(out) gammaB, logical, intent(out) succes, character(len=\*), intent(out) errorMessage )

calculateGammaB influence factor berms

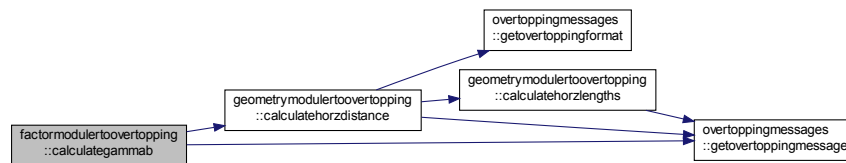
Parameters

in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>z2</i>	2% wave run-up (m)
in	<i>geometry</i>	structure with geometry data
out	<i>gammab</i>	influence factor berms

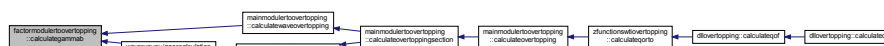
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 286 of file factorModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.2.1.2** subroutine, public factorModuleRTOovertopping::calculategammabeta ( real(wp), intent(inout) *Hm0*, real(wp), intent(inout) *Tm\_10*, real(wp), intent(in) *beta*, real(wp), intent(out) *gammaBeta\_z*, real(wp), intent(out) *gammaBeta\_o* )

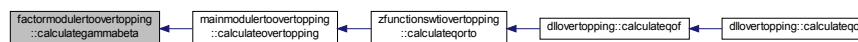
calculateGammaBeta influence factor angle of wave attack

#### Parameters

in, out	<i>hm0</i>	significant wave height (m)
in, out	<i>tm_10</i>	spectral wave period (s)
in	<i>beta</i>	angle of wave attack (degree)
out	<i>gammabeta_z</i>	influence factor angle of wave attack 2% wave run-up
out	<i>gammabeta_o</i>	influence factor angle of wave attack overtopping

Definition at line 115 of file factorModuleRTOovertopping.f90.

Here is the caller graph for this function:



**4.2.1.3** subroutine, public factorModuleRTOovertopping::calculategammaF ( real(wp), intent(in) *h*, real(wp), intent(in) *ksi0*, real(wp), intent(in) *ksi0Limit*, real(wp), intent(in) *gammaB*, real(wp), intent(in) *z2*, type(tpgeometry), intent(in) *geometry*, real(wp), intent(out) *gammaF*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

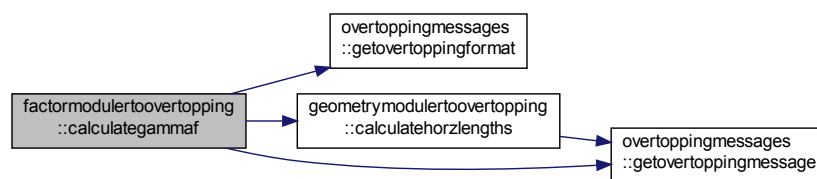
calculateGammaF influence factor roughness

#### Parameters

in	<i>h</i>	local water level (m+NAP)
in	<i>ksi0</i>	breaker parameter
in	<i>ksi0limit</i>	limit value breaker parameter
in	<i>gammab</i>	influence factor berms
in	<i>z2</i>	2% wave run-up (m)
in	<i>geometry</i>	structure with geometry data
out	<i>gammaf</i>	influence factor roughness
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 155 of file factorModuleRTOOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.2.1.4** subroutine, public factormodulertoovertopping::calculatetanalpha ( real(wp), intent(in) *h*, real(wp), intent(in) *Hm0*, real(wp), intent(in) *z2*, type(tpgeometry), intent(in) *geometry*, real(wp), intent(out) *tanAlpha*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

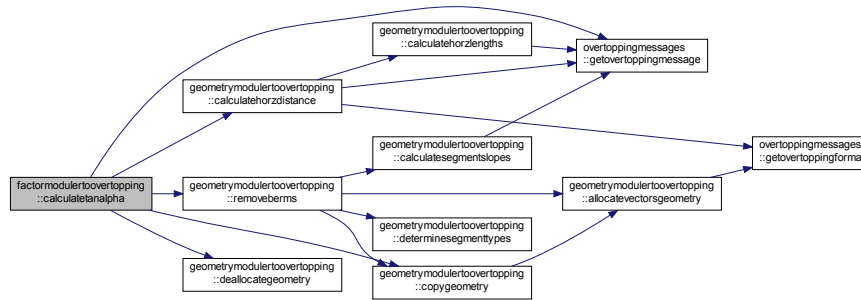
calculateTanAlpha representative slope angle

#### Parameters

in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>z2</i>	2% wave run-up (m)
in	<i>geometry</i>	structure with geometry data
out	<i>tanalpha</i>	representative slope angle
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 36 of file factorModuleRTOOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.3 formulamodulertooverlapping Module Reference

### Functions/Subroutines

- subroutine, public [calculateWaverunup](#) (Hm0, ksi0, ksi0Limit, gammaB, gammaF, gammaBeta, modelFactors, z2, succes, errorMessage)  
*calculateWaveRunup: calculate wave runup*
- subroutine, public [calculateWaveOvertoppingDischarge](#) (h, Hm0, tanAlpha, gammaB, gammaF, gammaBeta, ksi0, hCrest, modelFactors, Qo, succes, errorMessage)  
*calculateWaveOvertoppingDischarge: calculate the wave overtopping discharge*
- subroutine, public [calculateWavelength](#) (Tm\_10, L0)  
*calculateWaveLength: calculate the wave length*
- subroutine, public [calculateWavesteepness](#) (Hm0, Tm\_10, s0, succes, errorMessage)  
*calculateWaveSteepness: calculate the wave steepness*
- subroutine, public [calculateBreakerParameter](#) (tanAlpha, s0, ksi0, succes, errorMessage)  
*calculateBreakerParameter: calculate the breaker parameter*
- subroutine, public [calculateAngleWaveAttack](#) (phi, psi, beta)  
*calculateAngleWaveAttack: calculate the angle of wave attack*
- subroutine, public [calculateBreakerLimit](#) (modelFactors, gammaB, ksi0Limit, succes, errorMessage)  
*calculateBreakerLimit: calculate the breaker limit*
- subroutine, public [adjustInfluenceFactors](#) (gammaB, gammaF, gammaBeta, gammaBetaType, ksi0, ksi0Limit, succes, errorMessage)  
*adjustInfluenceFactors: adjust the influence factors*
- subroutine [realRootsCubicFunction](#) (a, b, c, d, N, x, succes, errorMessage)  
*realRootsCubicFunction: calculate the roots of a cubic function*
- subroutine [rootsGeneralCubic](#) (a, b, c, d, z, succes, errorMessage)  
*rootsGeneralCubic: calculate the roots of a generic cubic function*
- subroutine [rootsDepressedCubic](#) (p, q, z)  
*rootsDepressedCubic: calculate the roots of a depressed cubic function*
- subroutine [cubicRoots](#) (z, roots)



*cubicRoots*: calculate the roots of a cubic function

- logical function, public `isequalreal` (x1, x2)

*isEqualReal*: are two reals (almost) equal

- logical function, public `isequalzero` (x)

*isEqualZero*: is a real (almost) zero

### 4.3.1 Function/Subroutine Documentation

4.3.1.1 subroutine, public formulamodulertoovertopping::adjustinfluencefactors ( real(wp), intent(inout) *gammaB*, real(wp), intent(inout) *gammaF*, real(wp), intent(inout) *gammaBeta*, integer, intent(in) *gammaBetaType*, real(wp), intent(in) *ksi0*, real(wp), intent(in) *ksi0Limit*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

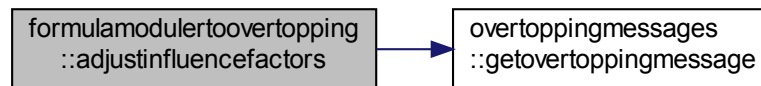
adjustInfluenceFactors: adjust the influence factors

Parameters

in, out	<i>gammab</i>	influence factor berms
in, out	<i>gammaf</i>	influence factor roughness
in, out	<i>gammabeta</i>	influence factor angle of wave attack
in	<i>gammabetatype</i>	type influence factor angle of wave attack: 1 = wave run-up, 2 = overtopping
in	<i>ksi0</i>	breaker parameter
in	<i>ksi0limit</i>	limit value breaker parameter
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 377 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.2 subroutine, public formulamodulertoovertopping::calculateanglewaveattack ( real(wp), intent(in) *phi*, real(wp), intent(in) *psi*, real(wp), intent(out) *beta* )

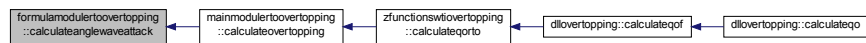
calculateAngleWaveAttack: calculate the angle of wave attack

Parameters

in	<i>phi</i>	wave direction (degree)
in	<i>psi</i>	dike normal (degree)
out	<i>beta</i>	angle of wave attack (degree)

Definition at line 285 of file formulaModuleRTOovertopping.f90.

Here is the caller graph for this function:

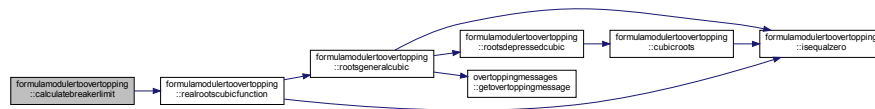


**4.3.1.3** subroutine, public formulaModuleRTOovertopping::calculateBreakerLimit ( type (tpOvertoppingInput), intent(in) *modelFactors*, real(wp), intent(in) *gammaB*, real(wp), intent(out) *ksiOLimit*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

calculateBreakerLimit: calculate the breaker limit

Definition at line 308 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.3.1.4** subroutine, public formulaModuleRTOovertopping::calculateBreakerParameter ( real(wp), intent(in) *tanAlpha*, real(wp), intent(in) *s0*, real(wp), intent(out) *ksi0*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

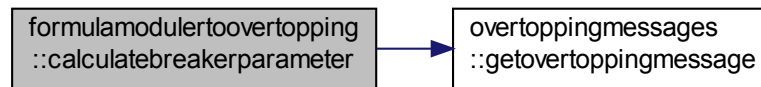
calculateBreakerParameter: calculate the breaker parameter

#### Parameters

in	<i>tanalpha</i>	representative slope angle
in	<i>s0</i>	wave steepness
out	<i>ksi0</i>	breaker parameter
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 244 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



```
4.3.1.5 subroutine, public formulamodulertovertopping::calculatewavelength ( real(wp), intent(in) Tm_10, real(wp), intent(out) L0 )
```

calculateWaveLength: calculate the wave length

## Parameters

in	$tm_{-}10$	spectral wave period (s)
out	$l0$	wave length (m)

Definition at line 181 of file formulaModuleRTOovertopping.f90.

Here is the caller graph for this function:



```

4.3.1.6 subroutine, public formulamodulertoovtopping::calculatewaveovertoppingdischarge ( real(wp), intent(in) h, real(wp),
intent(in) Hm0, real(wp), intent(in) tanAlpha, real(wp), intent(in) gammaB, real(wp), intent(in) gammaF, real(wp),
intent(in) gammaBeta, real(wp), intent(in) ksi0, real(wp), intent(in) hCrest, type(tpovertoppinginput), intent(in)
modelFactors, real(wp), intent(out) Qo, logical, intent(out) succes, character(len=*) , intent(out) errorMessage )

```

calculateWaveOvertoppingDischarge: calculate the wave overtopping discharge

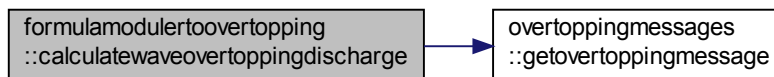
### Parameters

in	$h$	local water level (m+NAP)
in	$hm0$	significant wave height (m)
in	$\tan\alpha$	representative slope angle
in	$\gamma_b$	influence factor berms
in	$\gamma_r$	influence factor roughness

in	<i>gammabeta</i>	influence factor angle of wave attack
in	<i>ksi0</i>	breaker parameter
in	<i>hcrest</i>	crest level (m+NAP)
in	<i>modelfactors</i>	structure with model factors
out	<i>qo</i>	wave overtopping discharge (l/m per s)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 84 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.3.1.7** subroutine, public formulamodulertooveropping::calculatewaverunup ( real(wp), intent(in) *Hm0*, real(wp), intent(in) *ksi0*, real(wp), intent(in) *ksi0Limit*, real(wp), intent(inout) *gammaB*, real(wp), intent(inout) *gammaF*, real(wp), intent(inout) *gammaBeta*, type (tpovertoppinginput), intent(in) *modelFactors*, real(wp), intent(out) *z2*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

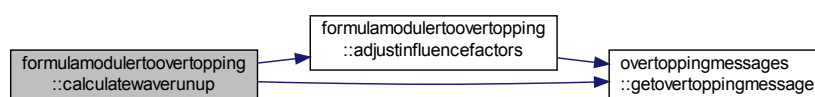
calculateWaveRunup: calculate wave runup

#### Parameters

in	<i>hm0</i>	significant wave height (m)
in	<i>ksi0</i>	breaker parameter
in	<i>ksi0limit</i>	limit value breaker parameter
in,out	<i>gammab</i>	influence factor berms
in,out	<i>gammaf</i>	influence factor roughness
in,out	<i>gammabeta</i>	influence factor angle of wave attack
in	<i>modelfactors</i>	structure with model factors
out	<i>z2</i>	2% wave run-up (m)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 35 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.3.1.8** subroutine, public formulamodulertoovertopping::calculatewavesteepness ( real(wp), intent(in) *Hm0*, real(wp), intent(in) *Tm\_10*, real(wp), intent(out) *s0*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

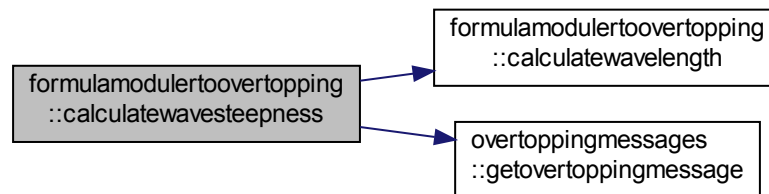
calculateWaveSteepness: calculate the wave steepness

#### Parameters

in	<i>hm0</i>	significant wave height (m)
in	<i>tm_10</i>	spectral wave period (s)
out	<i>s0</i>	wave steepness
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 202 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.3.1.9** subroutine formulamodulertoovertopping::cubicroots ( double complex, intent(in) *z*, double complex, dimension(3), intent(out) *roots* ) [private]

cubicRoots: calculate the roots of a cubic function

#### Parameters

in	<i>z</i>	complex number
out	<i>roots</i>	cubic roots

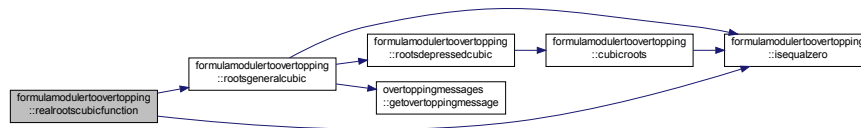
Definition at line 625 of file formulaModuleRTOovertopping.f90.



in	<i>a</i>	coefficient a cubic function
in	<i>b</i>	coefficient b cubic function
in	<i>c</i>	coefficient c cubic function
in	<i>d</i>	coefficient d cubic function
out	<i>n</i>	number of real roots cubic function
out	<i>x</i>	real roots cubic function
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 475 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.1.13 subroutine formulamodulertoovertopping::rootsdepressedcubic ( real(wp), intent(in) *p*, real(wp), intent(in) *q*, double complex, dimension(3), intent(out) *z* ) [private]

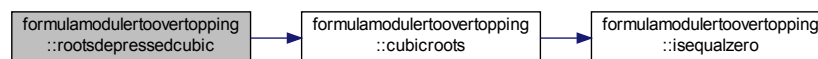
rootsDepressedCubic: calculate the roots of a depressed cubic function

Parameters

in	<i>p</i>	coefficient p depressed cubic
in	<i>q</i>	coefficient q depressed cubic
out	<i>z</i>	roots depressed cubic

Definition at line 584 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.14 subroutine formulamodulertoovertopping::rootsgeneralcubic ( real(wp), intent(in) *a*, real(wp), intent(in) *b*, real(wp), intent(in) *c*, real(wp), intent(in) *d*, double complex, dimension(3), intent(out) *z*, logical, intent(out) *success*, character(len=\*), intent(out) *errorMessage* ) [private]

rootsGeneralCubic: calculate the roots of a generic cubic function





- removeBerms: remove berms*
- subroutine, public [removedikesegments](#) (geometry, index, geometryAdjusted, succes, errorMessage)
  - removeDikeSegments: remove dike segments*
- subroutine, public [splitcrosssection](#) (geometry, L0, NwideBerms, geometrysectionB, geometrysectionF, succes, errorMessage)
  - splitCrossSection: split a cross section*
- subroutine, public [calculatehorzlengths](#) (geometry, yLower, yUpper, horzLengths, succes, errorMessage)
  - calculateHorzLengths: calculate horizontal lengths*
- subroutine, public [calculatehorzdistance](#) (geometry, yLower, yUpper, dx, succes, errorMessage)
  - calculateHorzDistance: calculate horizontal distance*
- subroutine, public [basicgeometrytest](#) (geometryF, success, errorStruct)
  - basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)*

#### 4.4.1 Function/Subroutine Documentation

- 4.4.1.1 subroutine, public geometrymodulertoovertopping::adjustnonhorizontalberms ( type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(out) *geometryFlatBerms*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

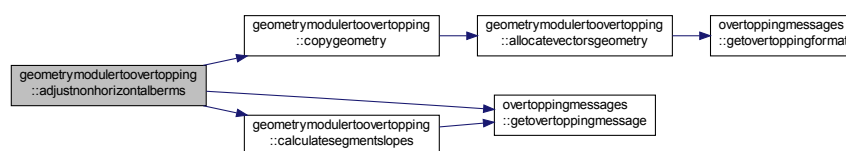
adjustNonHorizontalBerms: adjust non-horizontal berms

##### Parameters

in	<i>geometry</i>	structure with geometry data
out	<i>geometryflatberms</i>	geometry data with horizontal berms
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 526 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



- 4.4.1.2 subroutine, public geometrymodulertoovertopping::allocatevectorsgeometry ( integer, intent(in) *nCoordinates*, type (tpgeometry), intent(inout) *geometry*, logical, intent(out) *succes*, character(len=\*) , intent(inout) *errorMessage* )

allocateVectorsGeometry: allocate the geometry vectors

## Parameters

in	<i>ncoordinates</i>	number of coordinates
in, out	<i>geometry</i>	structure with geometry data
out	<i>success</i>	success flag
in, out	<i>errormessage</i>	error message (only set in case of error)

Definition at line 222 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.3 subroutine, public geometrymodulertooverlapping::basicgeometrytest ( type(overtoppinggeometrytypef), intent(in) *geometryF*, logical, intent(out) *success*, type(terrormessages), intent(inout) *errorStruct* )

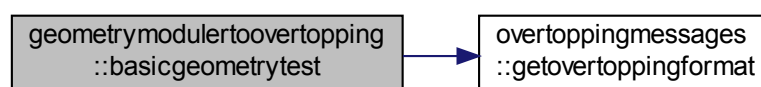
basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)

## Parameters

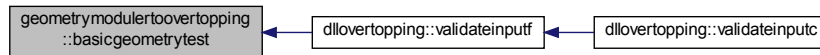
in	<i>geometryf</i>	struct with geometry and roughness
in, out	<i>errorstruct</i>	error message (only set if not successful)
out	<i>success</i>	success flag

Definition at line 1028 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.4 subroutine, public geometrymodulertooverlapping::calculatehorzdistance ( type (tpgeometry), intent(in) *geometry*, real(wp), intent(in) *yLower*, real(wp), intent(in) *yUpper*, real(wp), intent(out) *dx*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

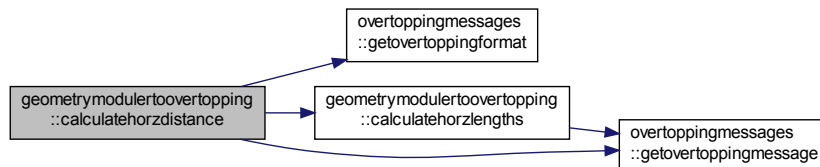
calculateHorzDistance: calculate horizontal distance

Parameters

in	<i>geometry</i>	structure with geometry data
in	<i>ylower</i>	y-coordinate lower bound (m+NAP)
in	<i>yupper</i>	y-coordinate upper bound (m+NAP)
out	<i>dx</i>	horizontal distance between bounds (m)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 979 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.5 subroutine, public geometrymodulertooverlapping::calculatehorzlengths ( type (tpgeometry), intent(in) *geometry*, real(wp), intent(in) *yLower*, real(wp), intent(in) *yUpper*, real(wp), dimension(geometry%ncoordinates-1), intent(out) *horzLengths*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

calculateHorzLengths: calculate horizontal lengths

Parameters

in	<i>geometry</i>	structure with geometry data
in	<i>ylower</i>	y-coord. lower bound (m+NAP)
in	<i>yupper</i>	y-coord. upper bound (m+NAP)
out	<i>horzlengths</i>	horizontal lengths segments (m)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 883 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



```
4.4.1.6 subroutine, public geometrymodulertooverlapping::calculatesegmentslopes ( type(tpgeometry), intent(inout) geometry,
logical, intent(out) success, character(len=*) , intent(out) errorMessage )
```

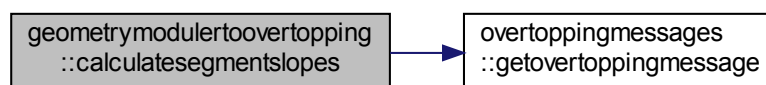
calculateSegmentSlopes: calculate the segment slopes

### Parameters

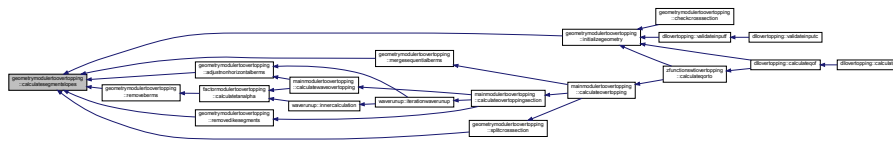
in, out	<i>geometry</i>	structure with geometry data
out	<i>success</i>	flag for success
out	<i>errorMessage</i>	error message

Definition at line 284 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.4.1.7** subroutine, public geometryModuleRTOOvertopping::checkCrossSection ( real(wp), intent(in) *psi*, integer, intent(in) *nCoordinates*, real(wp), dimension (ncoordinates), intent(in) *xCoordinates*, real(wp), dimension (ncoordinates), intent(in) *yCoordinates*, real(wp), dimension(ncoordinates-1), intent(in) *roughnessFactors*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

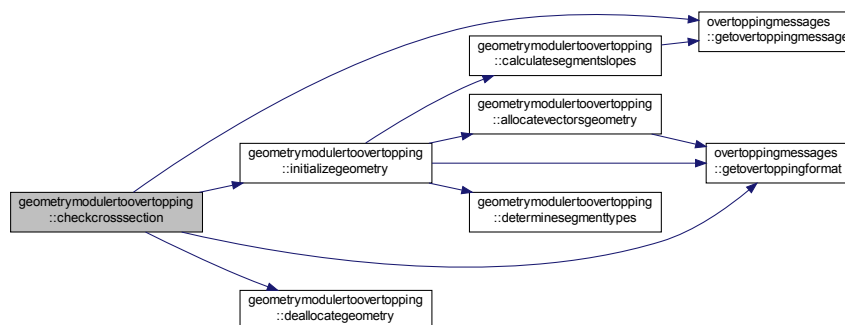
checkCrossSection: check cross section

Parameters

in	<i>psi</i>	dike normal (degree)
in	<i>ncoordinates</i>	number of coordinates
in	<i>xcoordinates</i>	x-coordinates (m)
in	<i>ycoordinates</i>	y-coordinates (m+NAP)
in	<i>roughnessfactors</i>	roughness factors
out	<i>succes</i>	flag for succes
out	<i>errorMessage</i>	error message

Definition at line 35 of file geometryModuleRTOOvertopping.f90.

Here is the call graph for this function:



**4.4.1.8** subroutine, public geometryModuleRTOOvertopping::copyGeometry ( type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(inout) *geometryCopy*, logical, intent(out) *succes*, character(len=\*) , intent(inout) *errorMessage* )

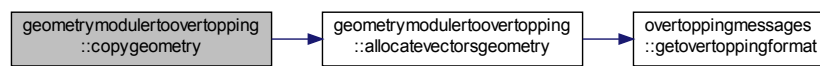
copyGeometry: copy a geometry structure

Parameters

in	<i>geometry</i>	structure with geometry data
in, out	<i>geometrycopy</i>	structure with geometry data copy
out	<i>succes</i>	succes flag
in, out	<i>errormessage</i>	error message, only set in case of error

Definition at line 366 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.4.1.9 subroutine, public geometrymodulertooverlapping::deallocateggeometry ( type (tpgeometry), intent(inout) *geometry* )

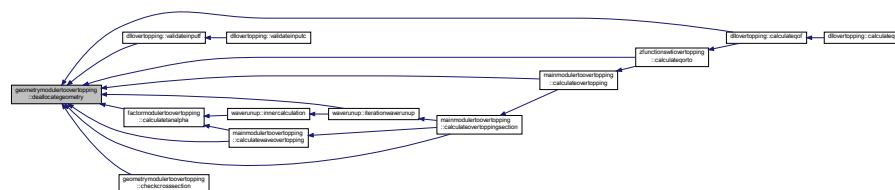
deallocategGeometry: deallocate the geometry vectors

Parameters

in, out	<i>geometry</i>	structure with geometry data
---------	-----------------	------------------------------

Definition at line 261 of file geometryModuleRTOoverlapping.f90.

Here is the caller graph for this function:



#### 4.4.1.10 subroutine, public geometrymodulertooverlapping::determinesegmenttypes ( type (tpgeometry), intent(inout) *geometry* )

determineSegmentTypes: determine the segment types

Parameters

in, out	<i>geometry</i>	structure with geometry data
---------	-----------------	------------------------------

Definition at line 323 of file geometryModuleRTOoverlapping.f90.





4.4.1.12 subroutine, public geometrymodulertooverlapping::mergesquentialberms ( type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(inout) *geometryMergedBerms*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

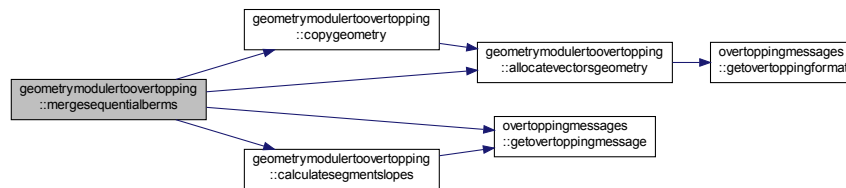
mergeSequentialBerms: merge sequential berms

## Parameters

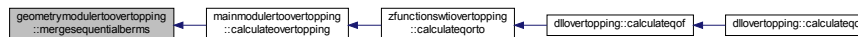
in	<i>geometry</i>	structure with geometry data
in, out	<i>geometrymergedberms</i>	geometry data with merged sequential berms
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 417 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.13 subroutine, public geometrymoduletooverlapping::removeberms ( type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(out) *geometryNoBerms*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

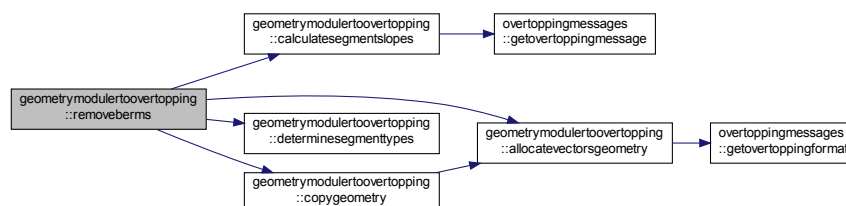
removeBerms: remove berms

## Parameters

in	<i>geometry</i>	structure with geometry data
out	<i>geometrynoberms</i>	geometry data without berms
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 615 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.4.1.14** subroutine, public geometrymodulertoovertopping::removedikesegments ( type (tpgeometry), intent(in) *geometry*, integer, intent(in) *index*, type (tpgeometry), intent(out) *geometryAdjusted*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

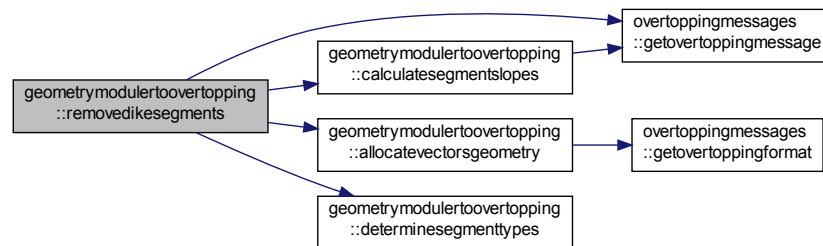
removeDikeSegments: remove dike segments

#### Parameters

in	<i>geometry</i>	structure with geometry data
in	<i>index</i>	index starting point new cross section
out	<i>geometryadjusted</i>	geometry data with removed dike segments
out	<i>succes</i>	flag for succes
out	<i>errorMessage</i>	error message

Definition at line 714 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**4.4.1.15** subroutine, public geometrymodulertoovertopping::splitcrosssection ( type (tpgeometry), intent(in) *geometry*, real(wp), intent(in) *L0*, integer, intent(out) *NwideBerms*, type (tpgeometry), intent(out) *geometrysectionB*, type (tpgeometry), intent(out) *geometrysectionF*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

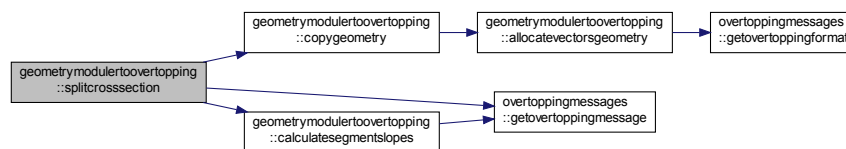
splitCrossSection: split a cross section

#### Parameters

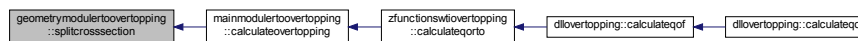
in	<i>geometry</i>	structure with geometry data
in	<i>l0</i>	wave length (m)
out	<i>nwideberms</i>	number of wide berms
out	<i>geometrysectionb</i>	geometry data with wide berms to ordinary berms
out	<i>geometrysectionf</i>	geometry data with wide berms to foreshores
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 778 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.5 mainmodulertooverlapping Module Reference

### Functions/Subroutines

- subroutine, public [calculateovertopping](#) (geometry, load, modelFactors, overtopping, succes, errorMessage)  
*calculateOvertopping: calculate the overtopping*
- subroutine, public [calculateovertoppingsection](#) (geometry, h, Hm0, Tm\_10, L0, gammaBeta\_z, gammaBeta\_o, modelFactors, overtopping, succes, errorMessage)  
*calculateOvertoppingSection: calculate the overtopping for a section*
- subroutine, public [calculatewaveovertopping](#) (geometry, h, Hm0, Tm\_10, z2, gammaBeta\_o, modelFactors, Qo, succes, errorMessage)  
*calculateWaveOvertopping: calculate wave overtopping*
- subroutine [calculateovertoppingnegativefreeboard](#) (load, geometry, overtopping, succes, errorMessage)  
*calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard*
- subroutine, public [interpolateresultssections](#) (geometry, L0, NwideBerms, overtoppingB, overtoppingF, overtopping, succes, errorMessage)  
*interpolateResultsSections: interpolate results for split cross sections*
- subroutine, public [checkinputdata](#) (geometry, load, modelFactors, succes, errorMessage)  
*checkInputdata: check the input data*
- subroutine, public [checkmodelfactors](#) (modelFactors, dimErrMessage, errorMessages, ierr)  
*checkModelFactors: check the input data*
- subroutine, public [convertovertoppinginput](#) (modelFactors, success, errorMessage)  
*convertOvertoppingInput: convert the model factors from C-like to Fortran*

### 4.5.1 Function/Subroutine Documentation

4.5.1.1 subroutine, public mainmodulertoovertopping::calculateovertopping ( type (tpgeometry), intent(in) *geometry*, type (tpload), intent(in) *load*, type (tpovertoppinginput), intent(in) *modelFactors*, type (tpovertopping), intent(out) *overtopping*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

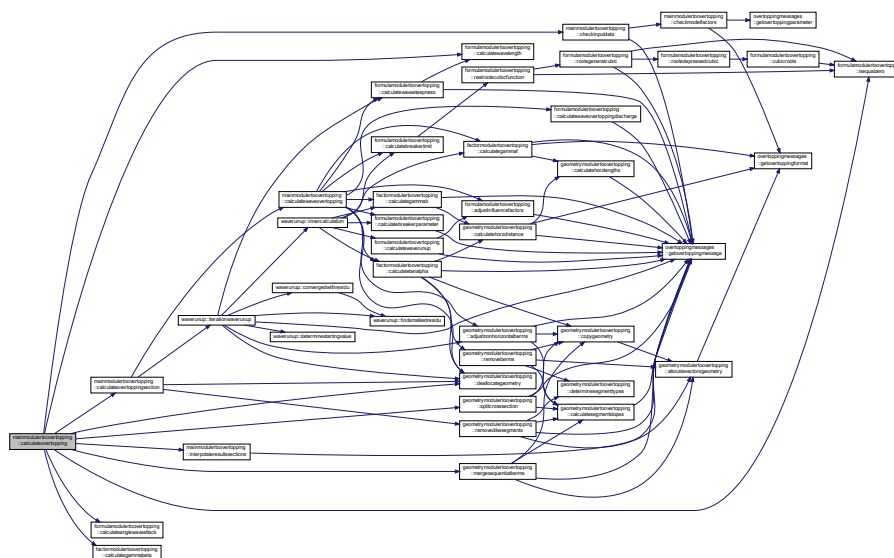
calculateOvertopping: calculate the overtopping

Parameters

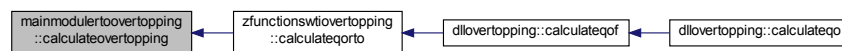
in	<i>geometry</i>	structure with geometry data
in	<i>load</i>	structure with load parameters
in	<i>modelfactors</i>	structure with model factors
out	<i>overtopping</i>	structure with overtopping results
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 37 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.2 subroutine mainmodulertoovertopping::calculateovertoppingnegativefreeboard ( type (tpload), intent(in) *load*, type (tpgeometry), intent(in) *geometry*, type (tpovertopping), intent(inout) *overtopping*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* ) [private]

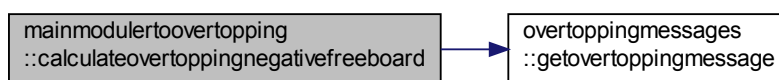
calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard

**Parameters**

in	<i>geometry</i>	structure with geometry data
in	<i>load</i>	structure with load parameters
in, out	<i>overtopping</i>	structure with overtopping results
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 479 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



**4.5.1.3** subroutine, public mainmodulertooverlapping::calculateovertoppingsection ( type (tpgeometry), intent(in) *geometry*, real(wp), intent(in) *h*, real(wp), intent(in) *Hm0*, real(wp), intent(in) *Tm\_10*, real(wp), intent(in) *L0*, real(wp), intent(inout) *gammaBeta\_z*, real(wp), intent(inout) *gammaBeta\_o*, type (tpovertoppinginput), intent(in) *modelFactors*, type (tpovertopping), intent(out) *overtopping*, logical, intent(out) *succes*, character(len=\*) , intent(out) *errorMessage* )

calculateOvertoppingSection: calculate the overtopping for a section

**Parameters**

in	<i>geometry</i>	structure with geometry data
in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>tm_10</i>	spectral wave period (s)
in	<i>l0</i>	wave length (m)
in, out	<i>gammabeta_z</i>	influence angle wave attack wave run-up
in, out	<i>gammabeta_o</i>	influence angle wave attack overtopping
in	<i>modelfactors</i>	structure with model factors
out	<i>overtopping</i>	structure with overtopping results
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 166 of file mainModuleRTOovertopping.f90.

The diagram is a directed graph with numerous nodes and edges. The nodes are rectangular boxes containing text labels. The labels are in German and appear to be related to a technical or organizational context. The graph shows a complex network of relationships, with many nodes having multiple incoming and outgoing edges. The nodes are arranged in a roughly circular pattern, with a central cluster of nodes and several peripheral nodes. The edges represent directed relationships between these entities.

```

graph LR
    A["mainmodulertoovtopping  
::calculateovertoppingsection"] --> B["mainmodulertoovtopping  
::calculateovertopping"]
    B --> C["zfunctionswwiovertopping  
::calculateqorto"]
    C --> D["dllovertopping::calculateqof"]
    D --> E["dllovertopping::calculateqof"]

```

calculateWaveOvertopping: calculate wave overtopping

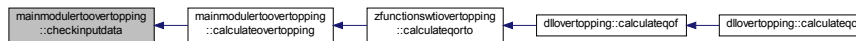
in	<i>geometry</i>	structure with geometry data
in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>tm_10</i>	spectral wave period (s)
in	<i>z2</i>	2% wave run-up (m)
in, out	<i>gammabeta_o</i>	influence angle wave attack overtopping
in	<i>modelfactors</i>	structure with model factors
out	<i>qo</i>	wave overtopping discharge (m3/m per s)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Generated on Fri Mar 25 2016 10:14:01 for Dikes Overtopping Kernel - Technical documentation by Doxygen





Here is the caller graph for this function:



4.5.1.6 subroutine, public mainmodulertoovertopping::checkmodelFactors ( type (tpovertoppinginput), intent(in) *modelFactors*, integer, intent(in) *dimErrMessage*, character(len=\*), dimension(dimerrmessage), intent(out) *errorMessages*, integer, intent(out) *ierr* )

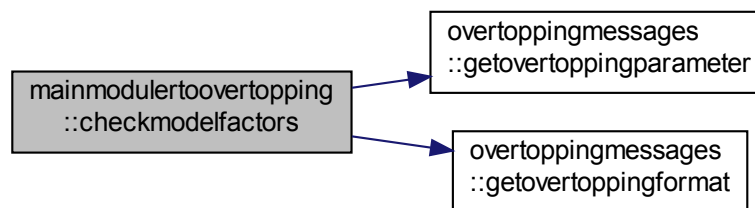
checkModelFactors: check the input data

#### Parameters

in	<i>modelFactors</i>	structure with model factors
in	<i>dimerrmessage</i>	max. number of error messages
out	<i>ierr</i>	number of errors found
out	<i>errormessages</i>	error message

Definition at line 655 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.7 subroutine, public mainmodulertoovertopping::convertovertoppinginput ( type (tpovertoppinginput), intent(inout) *modelFactors*, logical, intent(out) *success*, character(len=\*), intent(inout) *errorMessage* )

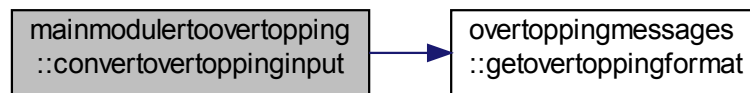
convertOvertoppingInput: convert the model factors from C-like to Fortran

#### Parameters

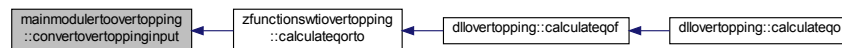
in, out	<i>modelfactors</i>	model factors and other input for overtopping
out	<i>success</i>	flag for success
in, out	<i>errormessage</i>	error message; only set when not successful

Definition at line 751 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.8 subroutine, public mainmodulertoovertopping::interpolateresultssections ( type (tpgeometry), intent(in) *geometry*, real(wp), intent(in) *L0*, integer, intent(in) *NwideBerms*, type (tpovertopping), intent(in) *overtoppingB*, type (tpovertopping), intent(in) *overtoppingF*, type (tpovertopping), intent(out) *overtopping*, logical, intent(out) *success*, character(len=\*) , intent(out) *errorMessage* )

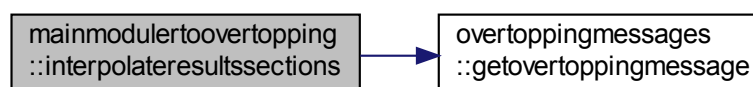
interpolateResultsSections: interpolate results for split cross sections

#### Parameters

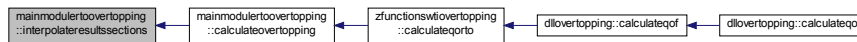
in	<i>geometry</i>	structure with geometry data
in	<i>l0</i>	wave length (m)
in	<i>nwideberms</i>	number of wide berms
in	<i>overtoppingb</i>	structure with overtopping results ordinary berms
in	<i>overtoppingf</i>	structure with overtopping results foreshores
out	<i>overtopping</i>	structure with combined overtopping results
out	<i>success</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 515 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.6 modulelogging Module Reference

### Data Types

- type `tlogging`  
*TLogging: structure for steering the logging.*

### Variables

- integer, parameter `maxfilenamelen` = 256  
*maximum length of filename*
- type(`tlogging`) `currentlogging`  
*copy of argument logging*

#### 4.6.1 Variable Documentation

##### 4.6.1.1 type(`tlogging`) `modulelogging::currentlogging`

copy of argument logging

Definition at line 21 of file `ModuleLogging.f90`.

##### 4.6.1.2 integer, parameter `modulelogging::maxfilenamelen` = 256

maximum length of filename

Definition at line 13 of file `ModuleLogging.f90`.

## 4.7 overtoppinginterface Module Reference

### Data Types

- type `overtoppinggeometrytype`
- type `overtoppinggeometrytypef`
- type `tpprofilecoordinate`

### Variables

- integer, parameter, public `varmodelfactorcriticalovertopping` = 8  
*Model factor critical overtopping.*

### 4.7.1 Variable Documentation

#### 4.7.1.1 integer, parameter, public overtoppinginterface::varmodelfactorcriticalovertopping = 8

Model factor critical overtopping.

Definition at line 17 of file overtoppingInterface.f90.

## 4.8 overtoppingmessages Module Reference

### Functions/Subroutines

- subroutine [setlanguage](#) (lang)  
*IDs for the strings in this module:*
- subroutine [getlanguage](#) (lang)  
*Subroutine that gets the language for error and validation messages.*
- character(len=[maxmsg](#)) function [getovertoppingmessage](#) (ID)  
*Subroutine that returns a message with the corresponding ID in the current language.*
- character(len=[maxmsg](#)) function [getovertoppingformat](#) (ID)  
*Subroutine that returns a Fortran format string with the corresponding ID in the current language.*
- character(len=[maxpar](#)) function [getovertoppingparameter](#) (ID)  
*Subroutine that returns the name of an input parameter with the corresponding ID in the current language.*

### Variables

- integer, parameter, private [maxmsg](#) = 128  
*Module for the messages in the overtopping dll, in Dutch or English.*
- integer, parameter, private [maxpar](#) = 32
- character(len=2), private [language](#) = 'NL'  
*default : Dutch*

### 4.8.1 Function/Subroutine Documentation

#### 4.8.1.1 subroutine overtoppingmessages::getlanguage ( character(len=\*), intent(out) lang )

Subroutine that gets the language for error and validation messages.

##### Parameters

out	lang	filled with current language ID
-----	------	---------------------------------

Definition at line 101 of file OvertoppingMessages.f90.

#### 4.8.1.2 character(len=[maxmsg](#)) function overtoppingmessages::getovertoppingformat ( integer, intent(in) ID )

Subroutine that returns a Fortran format string with the corresponding ID in the current language.

##### Parameters

in	id	identification number of string
----	----	---------------------------------

Definition at line 261 of file OvertoppingMessages.f90.



#### 4.8.1.5 subroutine overtoppingmessages::setlanguage ( character(len=\*), intent(in) lang )

IDs for the strings in this module:

Subroutine that sets the language for error and validation messages only strings 'NL' and 'UK' are recoqnized (lower and upper case)

##### Parameters

<i>in</i>	<i>lang</i>	new language ID to be used
-----------	-------------	----------------------------

Definition at line 83 of file OvertoppingMessages.f90.

### 4.8.2 Variable Documentation

#### 4.8.2.1 character(len=2), private overtoppingmessages::language = 'NL'

default : Dutch

Definition at line 17 of file OvertoppingMessages.f90.

#### 4.8.2.2 integer, parameter, private overtoppingmessages::maxmsg = 128

Module for the messages in the overtopping dll, in Dutch or English.

Definition at line 15 of file OvertoppingMessages.f90.

#### 4.8.2.3 integer, parameter, private overtoppingmessages::maxpar =32

Definition at line 15 of file OvertoppingMessages.f90.

## 4.9 typedefinitionsrtovertopping Module Reference

### Data Types

- type [tpgeometry](#)  
*tpGeometry: structure with geometry data*
- type [tpload](#)  
*tpLoad: structure with load parameters*
- type [tpovertopping](#)  
*tpOvertopping: structure with overtopping results*
- type [tpovertoppinginput](#)  
*OvertoppingModelFactors: C-structure with model factors.*

### Variables

- real(wp), parameter [xdiff\\_min](#) = 2.0d-2  
*minimal value distance between x-coordinates (m)*
- real(wp), parameter [margindiff](#) = 1.0d-14  
*margin for minimal distance (m)*
- real(wp), parameter [berm\\_min](#) = 0.0d0  
*minimal value gradient berm segment*
- real(wp), parameter [berm\\_max](#) = 1.0d0/15

- maximal value gradient berm segment*
- real(wp), parameter `slope_min` = 1.0d0/8
- minimal value gradient slope segment*
- real(wp), parameter `slope_max` = 1.0d0
- maximal value gradient slope segment*
- real(wp), parameter `margingrad` = 0.0025d0
- margin for minimal and maximal gradients*
- real(wp), parameter `rfactor_min` = 0.5d0
- minimal value roughness factor dike segments*
- real(wp), parameter `rfactor_max` = 1.0d0
- maximal value roughness factor dike segments*
- real(wp), parameter `mz2_min` = 0.0d0
- minimal value model factor of 2% runup height*
- real(wp), parameter `mz2_max` = huge(mz2\_max)
- maximal value model factor of 2% runup height*
- real(wp), parameter `frunup1_min` = 0.0d0
- minimal value model factor 1 for wave run-up*
- real(wp), parameter `frunup1_max` = huge(fRunup1\_max)
- maximal value model factor 1 for wave run-up*
- real(wp), parameter `frunup2_min` = 0.0d0
- minimal value model factor 2 for wave run-up*
- real(wp), parameter `frunup2_max` = huge(fRunup2\_max)
- maximal value model factor 2 for wave run-up*
- real(wp), parameter `frunup3_min` = 0.0d0
- minimal value model factor 3 for wave run-up*
- real(wp), parameter `frunup3_max` = huge(fRunup3\_max)
- maximal value model factor 3 for wave run-up*
- real(wp), parameter `fb_min` = 0.0d0
- minimal value model factor for breaking waves*
- real(wp), parameter `fb_max` = huge(fb\_max)
- maximal value model factor for breaking waves*
- real(wp), parameter `fn_min` = 0.0d0
- minimal value model factor for non-breaking waves*
- real(wp), parameter `fn_max` = huge(fn\_max)
- maximal value model factor for non-breaking waves*
- real(wp), parameter `fs_min` = 0.0d0
- minimal value model factor for shallow waves*
- real(wp), parameter `fs_max` = huge(fs\_max)
- maximal value model factor for shallow waves*
- real(wp), parameter `foreshore_min` = 0.3d0
- minimal value reduction factor foreshore*
- real(wp), parameter `foreshore_max` = 1.0d0
- maximal value reduction factor foreshore*
- integer, parameter `z2_iter_max1` = 49
- maximal number of iterations for calculation z2 part 1*
- integer, parameter `z2_iter_max2` = 70
- maximal number of iterations for calculation z2 part 1 & 2*
- real(wp), parameter `z2_margin` = 0.001d0
- margin for convergence criterium calculation z2*

### 4.9.1 Variable Documentation

#### 4.9.1.1 `real(wp), parameter typeDefinitionsRTOovertopping::berm_max = 1.0d0/15`

maximal value gradient berm segment

Definition at line 67 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.2 `real(wp), parameter typeDefinitionsRTOovertopping::berm_min = 0.0d0`

minimal value gradient berm segment

Definition at line 66 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.3 `real(wp), parameter typeDefinitionsRTOovertopping::fb_max = huge(fb_max)`

maximal value model factor for breaking waves

Definition at line 82 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.4 `real(wp), parameter typeDefinitionsRTOovertopping::fb_min = 0.0d0`

minimal value model factor for breaking waves

Definition at line 81 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.5 `real(wp), parameter typeDefinitionsRTOovertopping::fn_max = huge(fn_max)`

maximal value model factor for non-breaking waves

Definition at line 84 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.6 `real(wp), parameter typeDefinitionsRTOovertopping::fn_min = 0.0d0`

minimal value model factor for non-breaking waves

Definition at line 83 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.7 `real(wp), parameter typeDefinitionsRTOovertopping::foreshore_max = 1.0d0`

maximal value reduction factor foreshore

Definition at line 88 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.8 `real(wp), parameter typeDefinitionsRTOovertopping::foreshore_min = 0.3d0`

minimal value reduction factor foreshore

Definition at line 87 of file `typeDefinitionsRTOovertopping.f90`.

#### 4.9.1.9 `real(wp), parameter typeDefinitionsRTOovertopping::frunup1_max = huge(fRunup1_max)`

maximal value model factor 1 for wave run-up

Definition at line 76 of file `typeDefinitionsRTOovertopping.f90`.



**4.9.1.10** `real(wp), parameter typedefinitionsrtoovertopping::frunup1_min = 0.0d0`

minimal value model factor 1 for wave run-up

Definition at line 75 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.11** `real(wp), parameter typedefinitionsrtoovertopping::frunup2_max = huge(fRunup2_max)`

maximal value model factor 2 for wave run-up

Definition at line 78 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.12** `real(wp), parameter typedefinitionsrtoovertopping::frunup2_min = 0.0d0`

minimal value model factor 2 for wave run-up

Definition at line 77 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.13** `real(wp), parameter typedefinitionsrtoovertopping::frunup3_max = huge(fRunup3_max)`

maximal value model factor 3 for wave run-up

Definition at line 80 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.14** `real(wp), parameter typedefinitionsrtoovertopping::frunup3_min = 0.0d0`

minimal value model factor 3 for wave run-up

Definition at line 79 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.15** `real(wp), parameter typedefinitionsrtoovertopping::fs_max = huge(fS_max)`

maximal value model factor for shallow waves

Definition at line 86 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.16** `real(wp), parameter typedefinitionsrtoovertopping::fs_min = 0.0d0`

minimal value model factor for shallow waves

Definition at line 85 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.17** `real(wp), parameter typedefinitionsrtoovertopping::margindiff = 1.0d-14`

margin for minimal distance (m)

Definition at line 65 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.18** `real(wp), parameter typedefinitionsrtoovertopping::margingrad = 0.0025d0`

margin for minimal and maximal gradients

Definition at line 70 of file typeDefinitionsRTOovertopping.f90.

**4.9.1.19 real(wp), parameter typedefinitionsrtovertopping::mz2\_max = huge(mz2\_max)**

maximal value model factor of 2% runup height

Definition at line 74 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.20 real(wp), parameter typedefinitionsrtovertopping::mz2\_min = 0.0d0**

minimal value model factor of 2% runup height

Definition at line 73 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.21 real(wp), parameter typedefinitionsrtovertopping::rfactor\_max = 1.0d0**

maximal value roughness factor dike segments

Definition at line 72 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.22 real(wp), parameter typedefinitionsrtovertopping::rfactor\_min = 0.5d0**

minimal value roughness factor dike segments

Definition at line 71 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.23 real(wp), parameter typedefinitionsrtovertopping::slope\_max = 1.0d0**

maximal value gradient slope segment

Definition at line 69 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.24 real(wp), parameter typedefinitionsrtovertopping::slope\_min = 1.0d0/8**

minimal value gradient slope segment

Definition at line 68 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.25 real(wp), parameter typedefinitionsrtovertopping::xdiff\_min = 2.0d-2**

minimal value distance between x-coordinates (m)

Definition at line 64 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.26 integer, parameter typedefinitionsrtovertopping::z2\_iter\_max1 = 49**

maximal number of iterations for calculation z2 part 1

Definition at line 89 of file typeDefinitionsRTOvertopping.f90.

**4.9.1.27 integer, parameter typedefinitionsrtovertopping::z2\_iter\_max2 = 70**

maximal number of iterations for calculation z2 part 1 & 2

Definition at line 90 of file typeDefinitionsRTOvertopping.f90.

4.9.1.28 `real(wp), parameter typedefinitionsrtovertopping::z2_margin = 0.001d0`

margin for convergence criterium calculation z2

Definition at line 91 of file typeDefinitionsRTOOvertopping.f90.

## 4.10 waverunup Module Reference

### Functions/Subroutines

- subroutine, public [iterationwaverunup](#) (geometry, h, Hm0, Tm\_10, gammaBeta\_z, modelFactors, z2, succes, errorMessage)  
*iterationWaveRunup: iteration for the wave runup*
- real(kind=wp) function [innercalculation](#) (geometry, h, Hm0, gammaBeta\_z, modelFactors, z2, s0, geometry↔ FlatBerms, succes, errorMessage)
- real(kind=wp) function [determinestartingvalue](#) (i, relaxationFactor, z2\_start, z2\_end, Hm0)
- integer function [findsmallestresidu](#) (z2\_start, z2\_end, n)
- subroutine [convergedwithresidu](#) (z2\_start, z2\_end)

#### 4.10.1 Function/Subroutine Documentation

4.10.1.1 subroutine `waverunup::convergedwithresidu` ( `real(kind=wp), dimension(:), intent(in) z2_start`, `real(kind=wp), dimension(:), intent(inout) z2_end` ) [`private`]

Definition at line 317 of file waveRunup.f90.

Here is the call graph for this function:



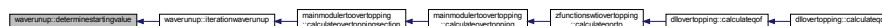
Here is the caller graph for this function:



4.10.1.2 `real(kind=wp) function waverunup::determinestartingvalue` ( `integer, intent(in) i`, `real(kind=wp), intent(in) relaxationFactor`, `real(kind=wp), dimension(:), intent(in) z2_start`, `real(kind=wp), dimension(:), intent(in) z2_end`, `real(kind=wp), intent(in) Hm0` ) [`private`]

Definition at line 266 of file waveRunup.f90.

Here is the caller graph for this function:







## Functions/Subroutines

- subroutine, public `calculateqorto` (diHeight, modelFactors, overtopping, load, geometry, succes, error↔Message)  
*Subroutine to calculate the overtopping discharge with the RTO-overtopping dll.*
- subroutine, public `profileinstructure` (nrCoordinates, xcoordinates, ycoordinates, diHeight, nrCoords↔Adjusted, xCoordsAdjusted, zCoordsAdjusted, succes, errorMessage)  
*Subroutine to fill the profile in a structure and call the adjustment function of the profile due to a desired dike height.*
- subroutine `adjustprofile` (nrCoordinates, coordinates, diHeight, nrCoordsAdjusted, xCoordsAdjusted, z↔CoordsAdjusted, succes, errorMessage)  
*Subroutine adjust the profile due to a desired dike height.*
- real(kind=wp) function, public `zfunclogratios` (qo, qc, mqo, mqc, success, errorMessage)  
*Routine to compute the limit state value by using the logs of the overtopping discharges (computed and desired)*

### 4.11.1 Detailed Description

Module for the Limit State Functions (Z-functions) for wave overtopping.

### 4.11.2 Function/Subroutine Documentation

- 4.11.2.1 subroutine `zfunctionswtiovertopping::adjustprofile` ( integer, intent(in) *nrCoordinates*, type(tpprofilecoordinate), dimension(nrcoordinates), intent(in) *coordinates*, real(kind=wp), intent(in) *diHeight*, integer, intent(out) *nrCoordsAdjusted*, real(kind=wp), dimension(:), pointer *xCoordsAdjusted*, real(kind=wp), dimension(:), pointer *zCoordsAdjusted*, logical, intent(out) *succes*, character(len=\*) intent(out) *errorMessage* ) [private]

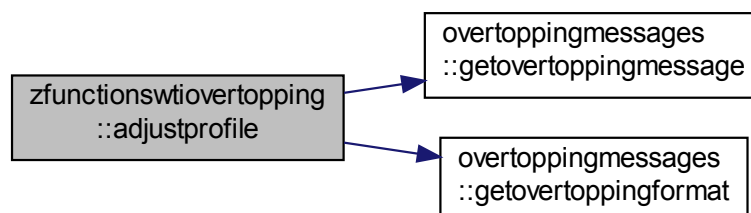
Subroutine adjust the profile due to a desired dike height.

#### Parameters

in	<i>nrcoordinates</i>	number of coordinates of the profile
in	<i>coordinates</i>	structure for the profile
in	<i>dikeheight</i>	dike height
out	<i>nrcoordsadjusted</i>	number of coordinates in the adjusted profile
	<i>xcoordsadjusted</i>	vector with x-coordinates of the adjusted profile
	<i>zcoordsadjusted</i>	vector with y-coordinates of the adjusted profile
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 110 of file `zFunctionsWTIOvertopping.f90`.

Here is the call graph for this function:



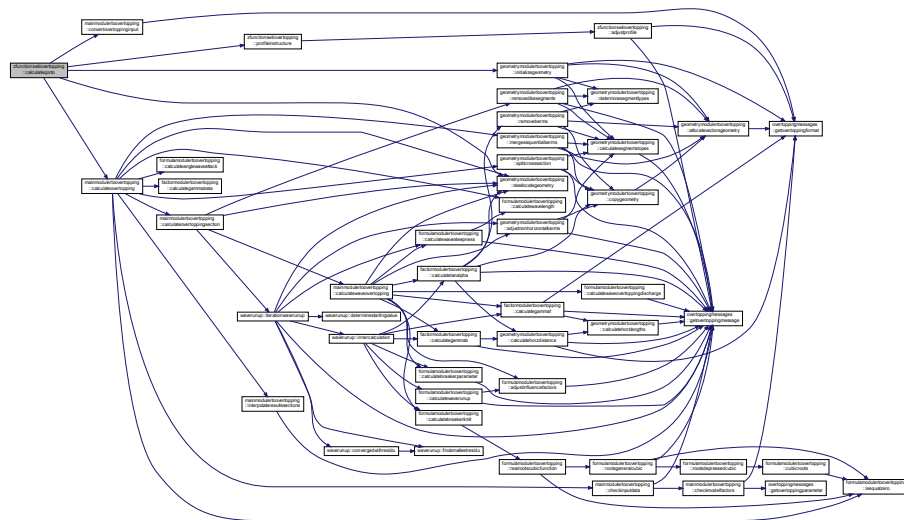
```

graph LR
    A[dioverlapping::validateinput] --> B[dioverlapping::validateinput]
    A --> C[zfunctions::wtoverlapping::profilestructure]
    B --> C
    C --> D[zfunctions::wtoverlapping::adjustprofile]
    E[dioverlapping::calculateqof] --> F[dioverlapping::calculateqof]
    E --> G[zfunctions::wtoverlapping::calculateqof]
    F --> G
    G --> H[dioverlapping::calculateqof]

```

## Parameters

Here is the call graph for this function:



```
graph RL
    A["dllovertopping::calculateqo"] --> B["dllovertopping::calculateqof"]
    B --> C["zfunctions::calculateqorto"]
```

4.11.2.3 subroutine, public zfunctionswtiovertopping::profileinstructure ( integer, intent(in) *nrCoordinates*, real(kind=wp), dimension(nrcoordinates), intent(in) *xcoordinates*, real(kind=wp), dimension(nrcoordinates), intent(in) *ycoordinates*, real(kind=wp), intent(in) *dikeHeight*, integer, intent(out) *nrCoordsAdjusted*, real(kind=wp), dimension(:), pointer *xCoordsAdjusted*, real(kind=wp), dimension(:), pointer *zCoordsAdjusted*, logical, intent(out) *succes*, character(len=\*), intent(out) *errorMessage* )

Subroutine to fill the profile in a structure and call the adjustment function of the profile due to a desired dike height.

#### Parameters

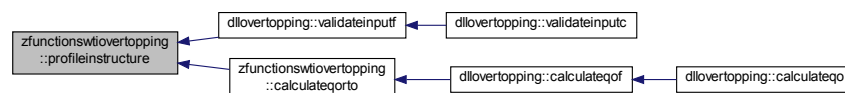
in	<i>nrcoordinates</i>	number of coordinates of the profile
in	<i>xcoordinates</i>	vector with x-coordinates of the profile
in	<i>ycoordinates</i>	vector with y-coordinates of the profile
in	<i>dikeheight</i>	dike height
out	<i>nrcoordsadjusted</i>	number of coordinates in the adjusted profile
	<i>xcoordsadjusted</i>	vector with x-coordinates of the adjusted profile
	<i>zcoordsadjusted</i>	vector with y-coordinates of the adjusted profile
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 85 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.4 real (kind=wp) function, public zfunctionswtiovertopping::zfunclogratios ( real (kind=wp), intent(in) *qo*, real (kind=wp), intent(in) *qc*, real (kind=wp), intent(in) *mgo*, real (kind=wp), intent(in) *mqc*, logical, intent(out) *success*, character(len=\*), intent(out) *errorMessage* )

Routine to compute the limit state value by using the logs of the overtopping discharges (computed and desired)

#### Parameters

in	<i>qo</i>	computed overtopping discharge
----	-----------	--------------------------------



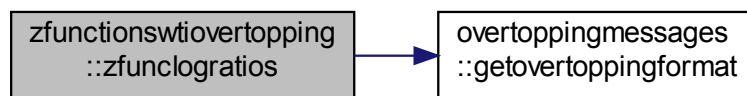
in	<i>qc</i>	Critical overtopping discharge
in	<i>m<sub>qo</sub></i>	Model factor computed overtopping discharge
in	<i>m<sub>qc</sub></i>	Model factor Critical overtopping discharge
out	<i>success</i>	Flag for succes
out	<i>errormessage</i>	error message, only set if not successful

**Returns**

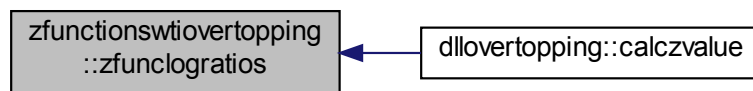
Value z-function

Definition at line 212 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



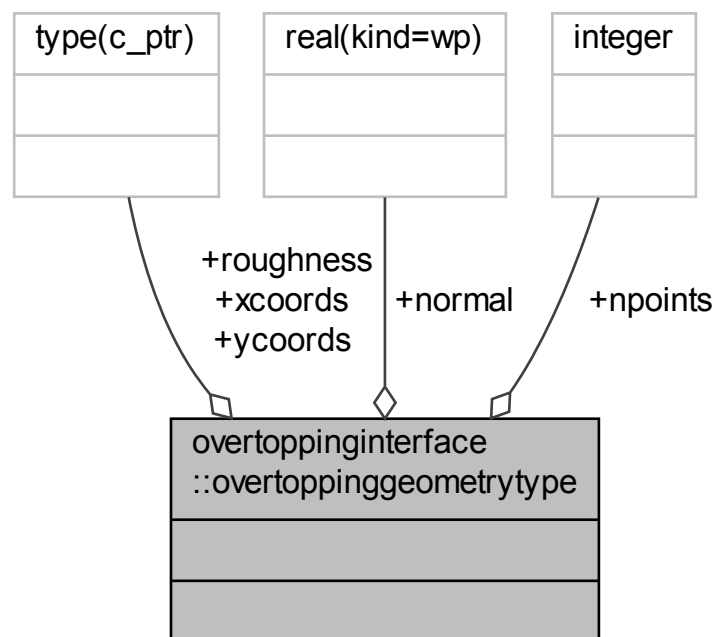


## Chapter 5

# Data Type Documentation

### 5.1 overtoppinginterface::overtoppinggeometrytype Type Reference

Collaboration diagram for overtoppinginterface::overtoppinggeometrytype:



#### Public Attributes

- `real(kind=wp)` `normal`
- `integer` `npoints`
- `type(c_ptr)` `xcoords`
- `type(c_ptr)` `ycoords`
- `type(c_ptr)` `roughness`

### 5.1.1 Detailed Description

Definition at line 25 of file overtoppingInterface.f90.

### 5.1.2 Member Data Documentation

#### 5.1.2.1 `real(kind=wp) overtoppinginterface::overtoppinggeometrytype::normal`

Definition at line 26 of file overtoppingInterface.f90.

#### 5.1.2.2 `integer overtoppinginterface::overtoppinggeometrytype::npoints`

Definition at line 27 of file overtoppingInterface.f90.

#### 5.1.2.3 `type(c_ptr) overtoppinginterface::overtoppinggeometrytype::roughness`

Definition at line 30 of file overtoppingInterface.f90.

#### 5.1.2.4 `type(c_ptr) overtoppinginterface::overtoppinggeometrytype::xcoords`

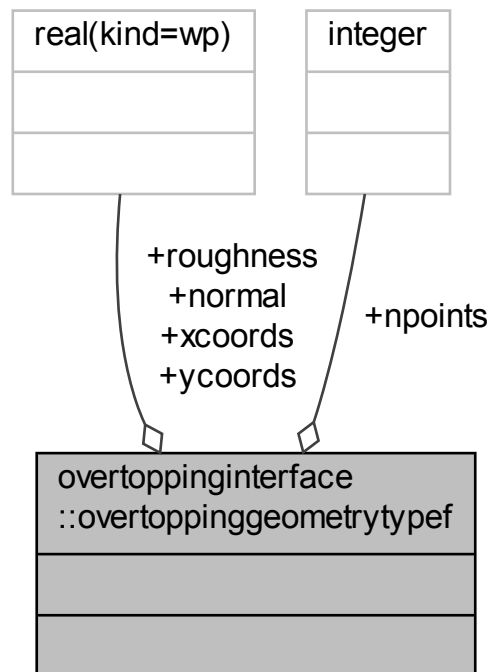
Definition at line 28 of file overtoppingInterface.f90.

#### 5.1.2.5 `type(c_ptr) overtoppinginterface::overtoppinggeometrytype::ycoords`

Definition at line 29 of file overtoppingInterface.f90.

## 5.2 overtoppinginterface::overtoppinggeometrytypef Type Reference

Collaboration diagram for overtoppinginterface::overtoppinggeometrytypef:



### Public Attributes

- real(kind=wp) [normal](#)
- integer [npoints](#)
- real(kind=wp), dimension(:), pointer [xcoords](#)
- real(kind=wp), dimension(:), pointer [ycoords](#)
- real(kind=wp), dimension(:), pointer [roughness](#)

### 5.2.1 Detailed Description

Definition at line 33 of file overtoppingInterface.f90.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 real(kind=wp) overtoppinginterface::overtoppinggeometrytypef::normal

Definition at line 34 of file overtoppingInterface.f90.

#### 5.2.2.2 integer overtoppinginterface::overtoppinggeometrytypef::npoints

Definition at line 35 of file overtoppingInterface.f90.

#### 5.2.2.3 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::roughness`

Definition at line 38 of file `overtoppingInterface.f90`.

#### 5.2.2.4 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::xcoords`

Definition at line 36 of file `overtoppingInterface.f90`.

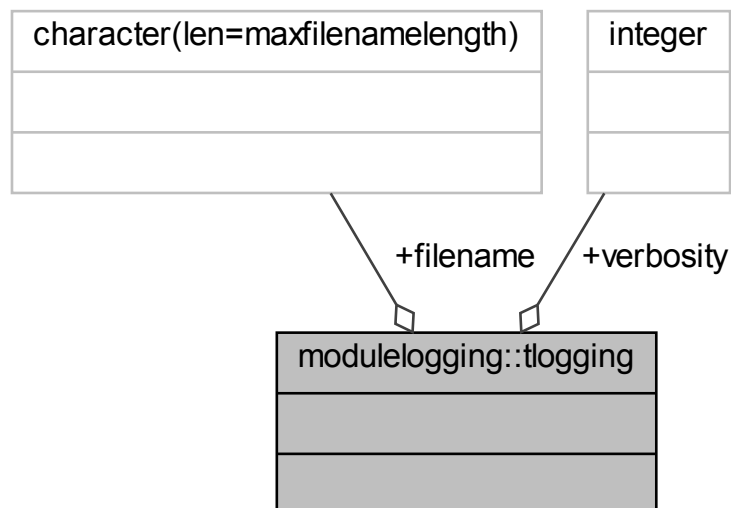
#### 5.2.2.5 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::ycoords`

Definition at line 37 of file `overtoppingInterface.f90`.

## 5.3 `modulelogging::tlogging` Type Reference

TLogging: structure for steering the logging.

Collaboration diagram for `modulelogging::tlogging`:



### Public Attributes

- integer `verbosity` = `verboseNone`  
*level of verbosity: one of `verboseNone`, `verboseBasic`, `verboseDetailed`, `verboseDebugging`*
- character(len=`maxfilenamelength`) `filename` = ''  
*filename of logging*

#### 5.3.1 Detailed Description

TLogging: structure for steering the logging.

Definition at line 16 of file `ModuleLogging.f90`.

### 5.3.2 Member Data Documentation

#### 5.3.2.1 `character(len=maxfilenamelength) modulelogging::tlogging::filename = ''`

filename of logging

Definition at line 18 of file ModuleLogging.f90.

#### 5.3.2.2 `integer modulelogging::tlogging::verbosity = verboseNone`

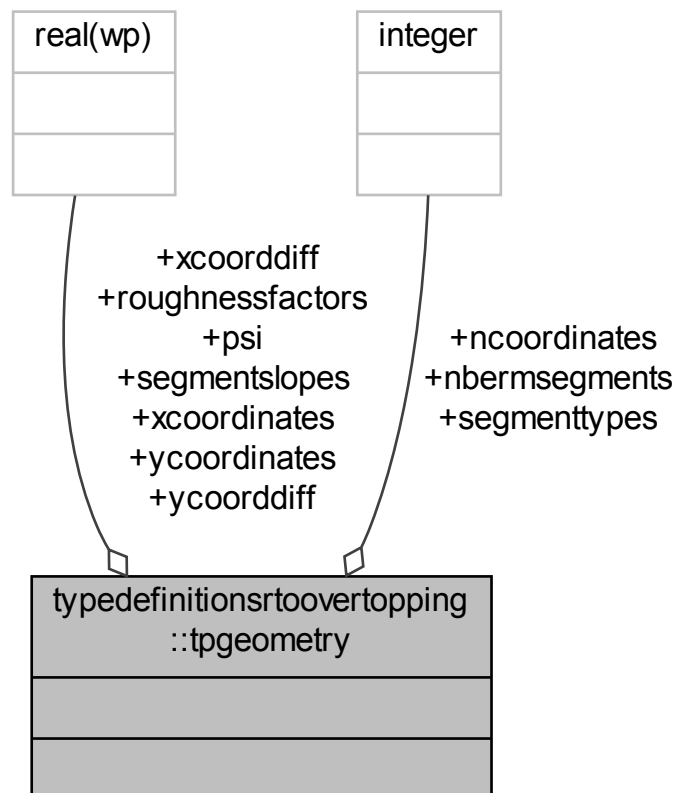
level of verbosity: one of verboseNone, verboseBasic, verboseDetailed, verboseDebugging

Definition at line 17 of file ModuleLogging.f90.

## 5.4 typedefinitionsrtoovertopping::tpgeometry Type Reference

tpGeometry: structure with geometry data

Collaboration diagram for typedefinitionsrtoovertopping::tpgeometry:



### Public Attributes

- `real(wp)` [psi](#)

- dike normal (degrees)*
- integer [ncoordinates](#)
  - number of coordinates cross section*
- real(wp), dimension(:), pointer [xcoordinates](#)
  - vector with x-coordinates cross section (m)*
- real(wp), dimension(:), pointer [ycoordinates](#)
  - vector with y-coordinates cross section (m+NAP)*
- real(wp), dimension(:), pointer [roughnessfactors](#)
  - vector with roughness factors cross section*
- real(wp), dimension(:), pointer [xcooordiff](#)
  - vector with differences in x-coordinates (m)*
- real(wp), dimension(:), pointer [ycooordiff](#)
  - vector with differences in y-coordinates (m)*
- real(wp), dimension(:), pointer [segmentslopes](#)
  - vector with slopes dike segments*
- integer, dimension(:), pointer [segmenttypes](#)
  - vector with segment types (1=slope,2=berm,3=other)*
- integer [nbermsements](#)
  - number of berm segments*

#### 5.4.1 Detailed Description

tpGeometry: structure with geometry data

Definition at line 18 of file typeDefinitionsRTOovertopping.f90.

#### 5.4.2 Member Data Documentation

##### 5.4.2.1 integer typedefinitionsrtooverlapping::tpgeometry::nbermsements

number of berm segments

Definition at line 28 of file typeDefinitionsRTOovertopping.f90.

##### 5.4.2.2 integer typedefinitionsrtooverlapping::tpgeometry::ncoordinates

number of coordinates cross section

Definition at line 20 of file typeDefinitionsRTOovertopping.f90.

##### 5.4.2.3 real(wp) typedefinitionsrtooverlapping::tpgeometry::psi

dike normal (degrees)

Definition at line 19 of file typeDefinitionsRTOovertopping.f90.

##### 5.4.2.4 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::roughnessfactors

vector with roughness factors cross section

Definition at line 23 of file typeDefinitionsRTOovertopping.f90.



**5.4.2.5 real(wp), dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::segmentslopes**

vector with slopes dike segments

Definition at line 26 of file typeDefinitionsRTOover topping.f90.

**5.4.2.6 integer, dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::segmenttypes**

vector with segment types (1=slope,2=berm,3=other)

Definition at line 27 of file typeDefinitionsRTOover topping.f90.

**5.4.2.7 real(wp), dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::xcoorddiff**

vector with differences in x-coordinates (m)

Definition at line 24 of file typeDefinitionsRTOover topping.f90.

**5.4.2.8 real(wp), dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::xcoordinates**

vector with x-coordinates cross section (m)

Definition at line 21 of file typeDefinitionsRTOover topping.f90.

**5.4.2.9 real(wp), dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::ycoorddiff**

vector with differences in y-coordinates (m)

Definition at line 25 of file typeDefinitionsRTOover topping.f90.

**5.4.2.10 real(wp), dimension(:), pointer typedefinitionsrtoover topping::tpgeometry::ycoordinates**

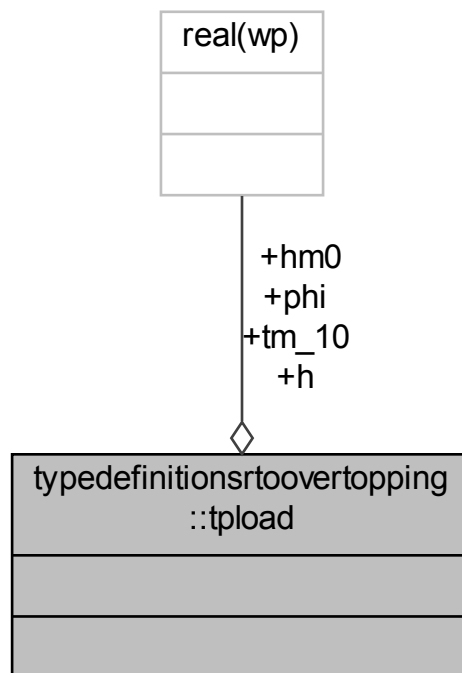
vector with y-coordinates cross section (m+NAP)

Definition at line 22 of file typeDefinitionsRTOover topping.f90.

**5.5 typedefinitionsrtoover topping::tpload Type Reference**

tpLoad: structure with load parameters

Collaboration diagram for `typedefinitionsrtoovertopping::tpload`:



## Public Attributes

- `real(wp) h`  
*local water level (m+NAP)*
- `real(wp) hm0`  
*significant wave height (m)*
- `real(wp) tm_10`  
*spectral wave period (s)*
- `real(wp) phi`  
*wave direction (degrees)*

## 5.5.1 Detailed Description

`tpLoad`: structure with load parameters

Definition at line 32 of file `typeDefinitionsRTOovertopping.f90`.

## 5.5.2 Member Data Documentation

### 5.5.2.1 `real(wp) typedefinitionsrtoovertopping::tpload::h`

local water level (m+NAP)

Definition at line 33 of file `typeDefinitionsRTOovertopping.f90`.

5.5.2.2 `real(wp) typedefinitionsrtooverlapping::tpload::hm0`

significant wave height (m)

Definition at line 34 of file `typeDefinitionsRTOoverlapping.f90`.

5.5.2.3 `real(wp) typedefinitionsrtooverlapping::tpload::phi`

wave direction (degrees)

Definition at line 36 of file `typeDefinitionsRTOoverlapping.f90`.

5.5.2.4 `real(wp) typedefinitionsrtooverlapping::tpload::tm_10`

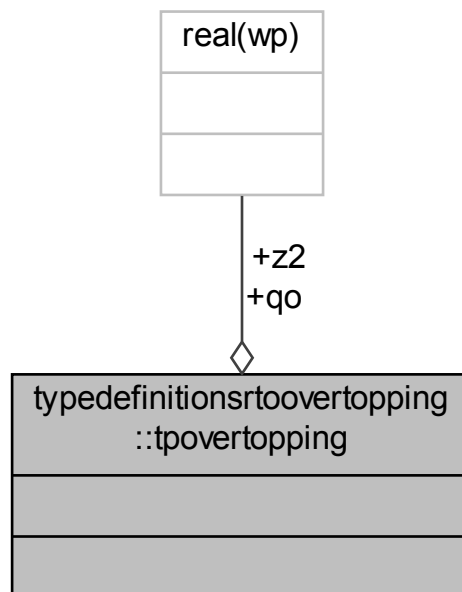
spectral wave period (s)

Definition at line 35 of file `typeDefinitionsRTOoverlapping.f90`.

5.6 `typedefinitionsrtooverlapping::tpoverlapping` Type Reference

`tpOverlapping`: structure with overtopping results

Collaboration diagram for `typedefinitionsrtooverlapping::tpoverlapping`:



## Public Attributes

- `real(wp)` [z2](#)  
*2% wave run-up (m)*
- `real(wp)` [qo](#)

*wave overtopping discharge (m<sup>3</sup>/m per s)*

### 5.6.1 Detailed Description

tpOvertopping: structure with overtopping results

Definition at line 56 of file typeDefinitionsRTOovertopping.f90.

### 5.6.2 Member Data Documentation

#### 5.6.2.1 `real(wp) typedefinitionsrtooveropping::tpoveropping::qo`

wave overtopping discharge (m<sup>3</sup>/m per s)

Definition at line 58 of file typeDefinitionsRTOovertopping.f90.

#### 5.6.2.2 `real(wp) typedefinitionsrtooveropping::tpoveropping::z2`

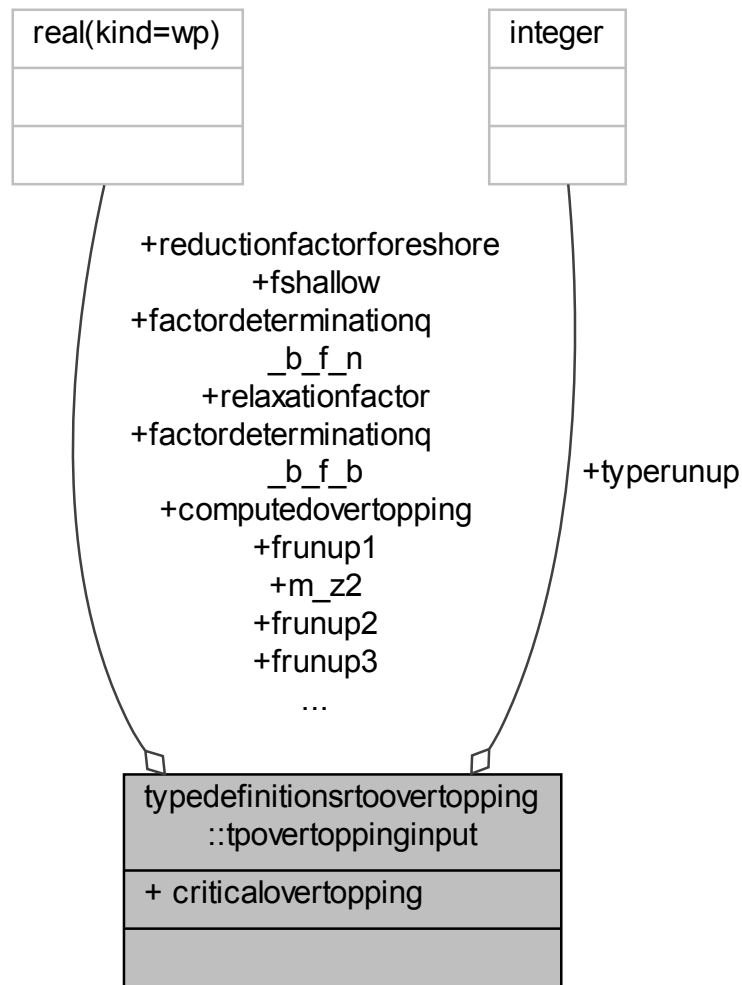
2% wave run-up (m)

Definition at line 57 of file typeDefinitionsRTOovertopping.f90.

## 5.7 `typedefinitionsrtooveropping::tpoveroppinginput` Type Reference

OvertoppingModelFactors: C-structure with model factors.

Collaboration diagram for typedefinitionsrtovertopping::tpovertoppinginput:



## Public Attributes

- `real(kind=wp)` [factordeterminationq\\_b\\_f\\_n](#)  
*model factor for non-breaking waves*
- `real(kind=wp)` [factordeterminationq\\_b\\_f\\_b](#)  
*model factor for breaking waves*
- `real(kind=wp)` [m\\_z2](#)  
*model factor describing the uncertainty of 2% runup height*
- `real(kind=wp)` [frunup1](#)  
*model factor 1 for wave run-up (for backwards compatability)*
- `real(kind=wp)` [frunup2](#)  
*model factor 2 for wave run-up (idem)*
- `real(kind=wp)` [frunup3](#)  
*model factor 3 for wave run-up (idem)*
- `real(kind=wp)` [fshallow](#)

- model factor for shallow waves*
- `real(kind=wp) computedovertopping`  
*model factor computed overtopping*
- `real(kind=wp) criticalovertopping`  
*model factor critical overtopping*
- `integer typerunup`  
*0: fRunup1, 2 and 3 are given; 1: m\_z2 is given*
- `real(kind=wp) relaxationfactor`  
*relaxation factor iteration procedure wave runup*
- `real(kind=wp) reductionfactorforeshore = 0.5_wp`  
*reduction factor foreshore*

### 5.7.1 Detailed Description

OvertoppingModelFactors: C-structure with model factors.

Definition at line 40 of file typeDefinitionsRTOovertopping.f90.

### 5.7.2 Member Data Documentation

#### 5.7.2.1 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::computedovertopping`

*model factor computed overtopping*

Definition at line 48 of file typeDefinitionsRTOovertopping.f90.

#### 5.7.2.2 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::criticalovertopping`

*model factor critical overtopping*

Definition at line 49 of file typeDefinitionsRTOovertopping.f90.

#### 5.7.2.3 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::factordeterminationq_b_f_b`

*model factor for breaking waves*

Definition at line 42 of file typeDefinitionsRTOovertopping.f90.

#### 5.7.2.4 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::factordeterminationq_b_f_n`

*model factor for non-breaking waves*

Definition at line 41 of file typeDefinitionsRTOovertopping.f90.

#### 5.7.2.5 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::frunup1`

*model factor 1 for wave run-up (for backwards compatability)*

Definition at line 44 of file typeDefinitionsRTOovertopping.f90.

#### 5.7.2.6 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::frunup2`

*model factor 2 for wave run-up (idem)*

Definition at line 45 of file typeDefinitionsRTOovertopping.f90.

**5.7.2.7 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::frunup3**

model factor 3 for wave run-up (idem)

Definition at line 46 of file typeDefinitionsRTOovertopping.f90.

**5.7.2.8 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::fshallow**

model factor for shallow waves

Definition at line 47 of file typeDefinitionsRTOovertopping.f90.

**5.7.2.9 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::m\_z2**

model factor describing the uncertainty of 2% runup height

Definition at line 43 of file typeDefinitionsRTOovertopping.f90.

**5.7.2.10 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::reductionfactorforeshore = 0.5\_wp**

reduction factor foreshore

Definition at line 52 of file typeDefinitionsRTOovertopping.f90.

**5.7.2.11 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::relaxationfactor**

relaxation factor iteration procedure wave runup

Definition at line 51 of file typeDefinitionsRTOovertopping.f90.

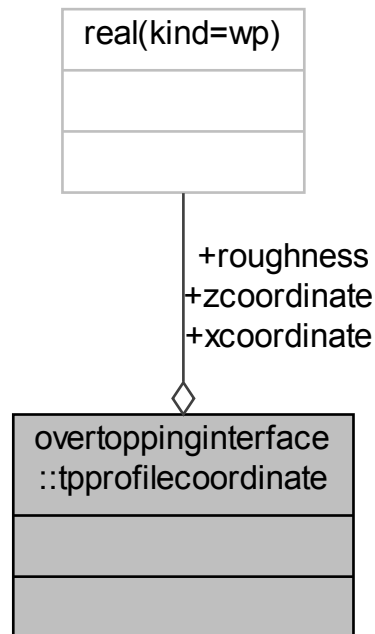
**5.7.2.12 integer typedefinitionsrtoovertopping::tpovertoppinginput::typerunup**

0: fRunup1, 2 and 3 are given; 1: m\_z2 is given

Definition at line 50 of file typeDefinitionsRTOovertopping.f90.

## 5.8 overtoppinginterface::tpprofilecoordinate Type Reference

Collaboration diagram for overtoppinginterface::tpprofilecoordinate:



### Public Attributes

- `real(kind=wp)` `xcoordinate`  
*X-coordinate foreland profile.*
- `real(kind=wp)` `zcoordinate`  
*Z-coordinate foreland profile.*
- `real(kind=wp)` `roughness`  
*Roughness of the area between two points.*

### 5.8.1 Detailed Description

Definition at line 19 of file `overtoppingInterface.f90`.

### 5.8.2 Member Data Documentation

#### 5.8.2.1 `real(kind=wp)` `overtoppinginterface::tpprofilecoordinate::roughness`

Roughness of the area between two points.

Definition at line 22 of file `overtoppingInterface.f90`.



**5.8.2.2 `real(kind=wp) overtoppinginterface::tpprofilecoordinate::xcoordinate`**

X-coordinate foreland profile.

Definition at line 20 of file overtoppingInterface.f90.

**5.8.2.3 `real(kind=wp) overtoppinginterface::tpprofilecoordinate::zcoordinate`**

Z-coordinate foreland profile.

Definition at line 21 of file overtoppingInterface.f90.



## Chapter 6

# File Documentation

### 6.1 dllOvertopping.f90 File Reference

Main entry for the dll DikesOvertopping FUNCTIONS/SUBROUTINES exported from dllOvertopping.dll:

#### Modules

- module [dllovertopping](#)  
*Calculate one type of overtopping.*

#### Functions/Subroutines

- subroutine, public [dllovertopping::calculateqo](#) (load, geometryInput, dikeHeight, modelFactors, overtopping, success, errorText, verbosity, logFile)  
*Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF↔ : convert C-like input structures to Fortran input structures.*
- subroutine, public [dllovertopping::calculateqof](#) (load, geometryF, dikeHeight, modelFactors, overtopping, success, errorText, logging)  
*Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.*
- subroutine, public [dllovertopping::calczvalue](#) (criticalOvertoppingRate, modelFactors, Qo, z, success, error↔ Message)  
*Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.*
- subroutine, public [dllovertopping::validateinputc](#) (geometryInput, dikeHeight, modelFactors, success, error↔ Text)  
*Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.*
- subroutine, public [dllovertopping::validateinputf](#) (geometryF, dikeHeight, modelFactors, errorStruct)  
*Subroutine that validates the geometry.*
- subroutine, public [dllovertopping::setlanguage](#) (lang)  
*Subroutine that sets the language for error and validation messages.*
- subroutine, public [dllovertopping::getlanguage](#) (lang)  
*Subroutine that gets the language for error and validation messages.*
- subroutine, public [dllovertopping::versionnumber](#) (version)  
*Subroutine that delivers the version number.*
- type(overtoppinggeometrytypef) function [dllovertopping::geometry\\_c\\_f](#) (geometryInput)  
*Private subroutine that converts geometry from c-pointer to fortran struct.*

### 6.1.1 Detailed Description

Main entry for the dll DikesOvertopping FUNCTIONS/SUBROUTINES exported from dllOvertopping.dll:

- calcZValue
- calculateQo
- calculateQoF
- ValidateInputC
- ValidateInputF
- SetLanguage
- GetLanguage
- versionNumber

## 6.2 factorModuleRTOovertopping.f90 File Reference

This file contains a module with functions for the slope angle and influence factors.

### Modules

- module [factormodulertooverlapping](#)

### Functions/Subroutines

- subroutine, public [factormodulertooverlapping::calculatetanalpha](#) (h, Hm0, z2, geometry, tanAlpha, succes, errorMessage)  
*calculateTanAlpha representative slope angle*
- subroutine, public [factormodulertooverlapping::calculategammabeta](#) (Hm0, Tm\_10, beta, gammaBeta\_↔z, gammaBeta\_o)  
*calculateGammaBeta influence factor angle of wave attack*
- subroutine, public [factormodulertooverlapping::calculategammaf](#) (h, ksi0, ksi0Limit, gammaB, z2, geometry, gammaF, succes, errorMessage)  
*calculateGammaF influence factor roughness*
- subroutine, public [factormodulertooverlapping::calculategammab](#) (h, Hm0, z2, geometry, gammaB, succes, errorMessage)  
*calculateGammaB influence factor berms*

### 6.2.1 Detailed Description

This file contains a module with functions for the slope angle and influence factors.

## 6.3 formulaModuleRTOovertopping.f90 File Reference

This file contains a module with the core computations for Dikes Overtopping.

## Modules

- module [formulamodulertooverlapping](#)

## Functions/Subroutines

- subroutine, public [formulamodulertooverlapping::calculatewaverunup](#) (Hm0, ksi0, ksi0Limit, gammaB, gammaF, gammaBeta, modelFactors, z2, succes, errorMessage)  
*calculateWaveRunup: calculate wave runup*
- subroutine, public [formulamodulertooverlapping::calculatewaveovertoppingdischarge](#) (h, Hm0, tanAlpha, gammaB, gammaF, gammaBeta, ksi0, hCrest, modelFactors, Qo, succes, errorMessage)  
*calculateWaveOvertoppingDischarge: calculate the wave overtopping discharge*
- subroutine, public [formulamodulertooverlapping::calculatewavelength](#) (Tm\_10, L0)  
*calculateWaveLength: calculate the wave length*
- subroutine, public [formulamodulertooverlapping::calculatewavesteepness](#) (Hm0, Tm\_10, s0, succes, errorMessage)  
*calculateWaveSteepness: calculate the wave steepness*
- subroutine, public [formulamodulertooverlapping::calculatebreakerparameter](#) (tanAlpha, s0, ksi0, succes, errorMessage)  
*calculateBreakerParameter: calculate the breaker parameter*
- subroutine, public [formulamodulertooverlapping::calculateanglewaveattack](#) (phi, psi, beta)  
*calculateAngleWaveAttack: calculate the angle of wave attack*
- subroutine, public [formulamodulertooverlapping::calculatebreakerlimit](#) (modelFactors, gammaB, ksi0Limit, succes, errorMessage)  
*calculateBreakerLimit: calculate the breaker limit*
- subroutine, public [formulamodulertooverlapping::adjustinfluencefactors](#) (gammaB, gammaF, gammaBeta, gammaBetaType, ksi0, ksi0Limit, succes, errorMessage)  
*adjustInfluenceFactors: adjust the influence factors*
- subroutine [formulamodulertooverlapping::realrootscubicfunction](#) (a, b, c, d, N, x, succes, errorMessage)  
*realRootsCubicFunction: calculate the roots of a cubic function*
- subroutine [formulamodulertooverlapping::rootsgeneralcubic](#) (a, b, c, d, z, succes, errorMessage)  
*rootsGeneralCubic: calculate the roots of a generic cubic function*
- subroutine [formulamodulertooverlapping::rootsdepressedcubic](#) (p, q, z)  
*rootsDepressedCubic: calculate the roots of a depressed cubic function*
- subroutine [formulamodulertooverlapping::cubicroots](#) (z, roots)  
*cubicRoots: calculate the roots of a cubic function*
- logical function, public [formulamodulertooverlapping::isequalreal](#) (x1, x2)  
*isEqualReal: are two reals (almost) equal*
- logical function, public [formulamodulertooverlapping::isequalzero](#) (x)  
*isEqualZero: is a real (almost) zero*

### 6.3.1 Detailed Description

This file contains a module with the core computations for Dikes Overtopping.

## 6.4 geometryModuleRTOovertopping.f90 File Reference

This file contains a module with the core computations for Dikes Overtopping related to the geometry.

## Modules

- module [geometrymodulertovertopping](#)

## Functions/Subroutines

- subroutine, public [geometrymodulertovertopping::checkcrosssection](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, succes, errorMessage)  
*checkCrossSection: check cross section*
- subroutine, public [geometrymodulertovertopping::initializegeometry](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, geometry, succes, errorMessage)  
*initializeGeometry: initialize the geometry*
- subroutine, public [geometrymodulertovertopping::allocatevectorsgeometry](#) (nCoordinates, geometry, succes, errorMessage)  
*allocateVectorsGeometry: allocate the geometry vectors*
- subroutine, public [geometrymodulertovertopping::deallocategemetry](#) (geometry)  
*deallocateGeometry: deallocate the geometry vectors*
- subroutine, public [geometrymodulertovertopping::calculatesegmentsslopes](#) (geometry, succes, errorMessage)  
*calculateSegmentSlopes: calculate the segment slopes*
- subroutine, public [geometrymodulertovertopping::determinesegmenttypes](#) (geometry)  
*determineSegmentTypes: determine the segment types*
- subroutine, public [geometrymodulertovertopping::copygeometry](#) (geometry, geometryCopy, succes, errorMessage)  
*copyGeometry: copy a geometry structure*
- subroutine, public [geometrymodulertovertopping::mergesquentialberms](#) (geometry, geometryMergedBerms, succes, errorMessage)  
*mergeSequentialBerms: merge sequential berms*
- subroutine, public [geometrymodulertovertopping::adjustnonhorizontalberms](#) (geometry, geometryFlatBerms, succes, errorMessage)  
*adjustNonHorizontalBerms: adjust non-horizontal berms*
- subroutine, public [geometrymodulertovertopping::removeberms](#) (geometry, geometryNoBerms, succes, errorMessage)  
*removeBerms: remove berms*
- subroutine, public [geometrymodulertovertopping::removedikesegments](#) (geometry, index, geometryAdjusted, succes, errorMessage)  
*removeDikeSegments: remove dike segments*
- subroutine, public [geometrymodulertovertopping::splitcrosssection](#) (geometry, L0, NwideBerms, geometrysectionB, geometrysectionF, succes, errorMessage)  
*splitCrossSection: split a cross section*
- subroutine, public [geometrymodulertovertopping::calculatehorzlengths](#) (geometry, yLower, yUpper, horzLengths, succes, errorMessage)  
*calculateHorzLengths: calculate horizontal lengths*
- subroutine, public [geometrymodulertovertopping::calculatehorzdistance](#) (geometry, yLower, yUpper, dx, succes, errorMessage)  
*calculateHorzDistance: calculate horizontal distance*
- subroutine, public [geometrymodulertovertopping::basicgeometrytest](#) (geometryF, success, errorStruct)  
*basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)*

### 6.4.1 Detailed Description

This file contains a module with the core computations for Dikes Overtopping related to the geometry.

## 6.5 mainModuleRTOovertopping.f90 File Reference

This file contains a module with the core computations for Dikes Overtopping.

### Modules

- module [mainmodulertooveropping](#)

### Functions/Subroutines

- subroutine, public [mainmodulertooveropping::calculateovertopping](#) (geometry, load, modelFactors, overtopping, succes, errorMessage)  
*calculateOvertopping: calculate the overtopping*
- subroutine, public [mainmodulertooveropping::calculateovertoppingsection](#) (geometry, h, Hm0, Tm\_10, L0, gammaBeta\_z, gammaBeta\_o, modelFactors, overtopping, succes, errorMessage)  
*calculateOvertoppingSection: calculate the overtopping for a section*
- subroutine, public [mainmodulertooveropping::calculatewaveovertopping](#) (geometry, h, Hm0, Tm\_10, z2, gammaBeta\_o, modelFactors, Qo, succes, errorMessage)  
*calculateWaveOvertopping: calculate wave overtopping*
- subroutine [mainmodulertooveropping::calculateovertoppingnegativefreeboard](#) (load, geometry, overtopping, succes, errorMessage)  
*calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard*
- subroutine, public [mainmodulertooveropping::interpolateresultssections](#) (geometry, L0, NwideBerms, overtoppingB, overtoppingF, overtopping, succes, errorMessage)  
*interpolateResultsSections: interpolate results for split cross sections*
- subroutine, public [mainmodulertooveropping::checkinputdata](#) (geometry, load, modelFactors, succes, errorMessage)  
*checkInputdata: check the input data*
- subroutine, public [mainmodulertooveropping::checkmodelFactors](#) (modelFactors, dimErrMsg, error←Messages, ierr)  
*checkModelFactors: check the input data*
- subroutine, public [mainmodulertooveropping::converttooveroppinginput](#) (modelFactors, success, error←Message)  
*convertOvertoppingInput: convert the model factors from C-like to Fortran*

### 6.5.1 Detailed Description

This file contains a module with the core computations for Dikes Overtopping.

## 6.6 ModuleLogging.f90 File Reference

Module for steering the extra logging.

### Data Types

- type [modulelogging::tlogging](#)  
*TLogging: structure for steering the logging.*

## Modules

- module [modulelogging](#)

## Variables

- integer, parameter [modulelogging::maxfilenamelength](#) = 256  
*maximum length of filename*
- type(tlogging) [modulelogging::currentlogging](#)  
*copy of argument logging*

### 6.6.1 Detailed Description

Module for steering the extra logging.

## 6.7 [overtoppingInterface.f90](#) File Reference

This file contains the parameters and types (structs) as part of the interface to and from dllOvertopping.

## Data Types

- type [overtoppinginterface::tpprofilecoordinate](#)
- type [overtoppinginterface::overtoppinggeometrytype](#)
- type [overtoppinginterface::overtoppinggeometrytypef](#)

## Modules

- module [overtoppinginterface](#)

## Variables

- integer, parameter, public [overtoppinginterface::varmodelfactorcriticalovertopping](#) = 8  
*Model factor critical overtopping.*

### 6.7.1 Detailed Description

This file contains the parameters and types (structs) as part of the interface to and from dllOvertopping.

## 6.8 [OvertoppingMessages.f90](#) File Reference

This file contains the messages in the overtopping dll, in Dutch or English.

## Modules

- module [overtoppingmessages](#)



## Functions/Subroutines

- subroutine [overtoppingmessages::setlanguage](#) (lang)  
*IDs for the strings in this module:*
- subroutine [overtoppingmessages::getlanguage](#) (lang)  
*Subroutine that gets the language for error and validation messages.*
- character(len=maxmsg) function [overtoppingmessages::getovertoppingmessage](#) (ID)  
*Subroutine that returns a message with the corresponding ID in the current language.*
- character(len=maxmsg) function [overtoppingmessages::getovertoppingformat](#) (ID)  
*Subroutine that returns a Fortran format string with the corresponding ID in the current language.*
- character(len=maxpar) function [overtoppingmessages::getovertoppingparameter](#) (ID)  
*Subroutine that returns the name of an input parameter with the corresponding ID in the current language.*

## Variables

- integer, parameter, private [overtoppingmessages::maxmsg](#) = 128  
*Module for the messages in the overtopping dll, in Dutch or English.*
- integer, parameter, private [overtoppingmessages::maxpar](#) = 32
- character(len=2), private [overtoppingmessages::language](#) = 'NL'  
*default : Dutch*

### 6.8.1 Detailed Description

This file contains the messages in the overtopping dll, in Dutch or English.

## 6.9 typeDefinitionsRTOovertopping.f90 File Reference

This file contains a module with the type definitions for Dikes Overtopping.

## Data Types

- type [typedefinitionsrtooveropping::tpgeometry](#)  
*tpGeometry: structure with geometry data*
- type [typedefinitionsrtooveropping::tpload](#)  
*tpLoad: structure with load parameters*
- type [typedefinitionsrtooveropping::tpovertoppinginput](#)  
*OvertoppingModelFactors: C-structure with model factors.*
- type [typedefinitionsrtooveropping::tpovertopping](#)  
*tpOvertopping: structure with overtopping results*

## Modules

- module [typedefinitionsrtooveropping](#)

## Variables

- `real(wp)`, parameter `typedefinitionsrtoovertopping::xdiff_min = 2.0d-2`  
*minimal value distance between x-coordinates (m)*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::margindiff = 1.0d-14`  
*margin for minimal distance (m)*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::berm_min = 0.0d0`  
*minimal value gradient berm segment*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::berm_max = 1.0d0/15`  
*maximal value gradient berm segment*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::slope_min = 1.0d0/8`  
*minimal value gradient slope segment*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::slope_max = 1.0d0`  
*maximal value gradient slope segment*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::maringrad = 0.0025d0`  
*margin for minimal and maximal gradients*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::rfactor_min = 0.5d0`  
*minimal value roughness factor dike segments*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::rfactor_max = 1.0d0`  
*maximal value roughness factor dike segments*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::mz2_min = 0.0d0`  
*minimal value model factor of 2% runup height*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::mz2_max = huge(mz2_max)`  
*maximal value model factor of 2% runup height*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup1_min = 0.0d0`  
*minimal value model factor 1 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup1_max = huge(fRunup1_max)`  
*maximal value model factor 1 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup2_min = 0.0d0`  
*minimal value model factor 2 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup2_max = huge(fRunup2_max)`  
*maximal value model factor 2 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup3_min = 0.0d0`  
*minimal value model factor 3 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::frunup3_max = huge(fRunup3_max)`  
*maximal value model factor 3 for wave run-up*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fb_min = 0.0d0`  
*minimal value model factor for breaking waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fb_max = huge(fb_max)`  
*maximal value model factor for breaking waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fn_min = 0.0d0`  
*minimal value model factor for non-breaking waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fn_max = huge(fn_max)`  
*maximal value model factor for non-breaking waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fs_min = 0.0d0`  
*minimal value model factor for shallow waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::fs_max = huge(fs_max)`  
*maximal value model factor for shallow waves*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::foreshore_min = 0.3d0`  
*minimal value reduction factor foreshore*
- `real(wp)`, parameter `typedefinitionsrtoovertopping::foreshore_max = 1.0d0`

*maximal value reduction factor foreshore*

- integer, parameter `typedefinitionsrtoovertopping::z2_iter_max1` = 49  
*maximal number of iterations for calculation z2 part 1*
- integer, parameter `typedefinitionsrtoovertopping::z2_iter_max2` = 70  
*maximal number of iterations for calculation z2 part 1 & 2*
- real(wp), parameter `typedefinitionsrtoovertopping::z2_margin` = 0.001d0  
*margin for convergence criterium calculation z2*

### 6.9.1 Detailed Description

This file contains a module with the type definitions for Dikes Overtopping.

## 6.10 waveRunup.f90 File Reference

This file contains a module with the core computations for Dikes Overtopping.

### Modules

- module `waverunup`

### Functions/Subroutines

- subroutine, public `waverunup::iterationwaverunup` (geometry, h, Hm0, Tm\_10, gammaBeta\_z, modelFactors, z2, succes, errorMessage)  
*iterationWaveRunup: iteration for the wave runup*
- real(kind=wp) function `waverunup::innercalculation` (geometry, h, Hm0, gammaBeta\_z, modelFactors, z2, s0, geometryFlatBerms, succes, errorMessage)
- real(kind=wp) function `waverunup::determinestartingvalue` (i, relaxationFactor, z2\_start, z2\_end, Hm0)
- integer function `waverunup::findsmallestresidu` (z2\_start, z2\_end, n)
- subroutine `waverunup::convergedwithresidu` (z2\_start, z2\_end)

### 6.10.1 Detailed Description

This file contains a module with the core computations for Dikes Overtopping.

## 6.11 zFunctionsWTIOvertopping.f90 File Reference

This file contains the limit state functions for wave overtopping within WTI.

### Modules

- module `zfunctionswtiovertopping`  
*Module for the Limit State Functions (Z-functions) for wave overtopping.*

## Functions/Subroutines

- subroutine, public `zfunctionswtiovertopping::calculateqorto` (dikeHeight, modelFactors, overtopping, load, geometry, succes, errorMessage)

*Subroutine to calculate the overtopping discharge with the RTO-overtopping dll.*

- subroutine, public `zfunctionswtiovertopping::profileinstructure` (nrCoordinates, xcoordinates, ycoordinates, dikeHeight, nrCoordsAdjusted, xCoordsAdjusted, zCoordsAdjusted, succes, errorMessage)

*Subroutine to fill the profile in a structure and call the adjustment function of the profile due to a desired dike height.*

- subroutine `zfunctionswtiovertopping::adjustprofile` (nrCoordinates, coordinates, dikeHeight, nrCoordsAdjusted, xCoordsAdjusted, zCoordsAdjusted, succes, errorMessage)

*Subroutine adjust the profile due to a desired dike height.*

- real(kind=wp) function, public `zfunctionswtiovertopping::zfunclogratios` (qo, qc, mqo, mqc, success, errorMessage)

*Routine to compute the limit state value by using the logs of the overtopping discharges (computed and desired)*

### 6.11.1 Detailed Description

This file contains the limit state functions for wave overtopping within WTI.

# Index

- adjustinfluencefactors
  - formulamodulertooverlapping, 17
- adjustnonhorizontalberms
  - geometrymodulertooverlapping, 26
- adjustprofile
  - zfunctionswtioverlapping, 54
- allocatevectorsgeometry
  - geometrymodulertooverlapping, 26
- basicgeometrytest
  - geometrymodulertooverlapping, 27
- berm\_max
  - typedefinitionsrtooverlapping, 48
- berm\_min
  - typedefinitionsrtooverlapping, 48
- calculateanglewaveattack
  - formulamodulertooverlapping, 17
- calculatebreakerlimit
  - formulamodulertooverlapping, 18
- calculatebreakerparameter
  - formulamodulertooverlapping, 18
- calculategammab
  - factormodulertooverlapping, 13
- calculategammabeta
  - factormodulertooverlapping, 14
- calculategammalfactormodulertooverlapping, 14
- calculatehorzdistance
  - geometrymodulertooverlapping, 28
- calculatehorzlengths
  - geometrymodulertooverlapping, 28
- calculateoverlapping
  - mainmodulertooverlapping, 37
- calculateoverlappingnegativefreeboard
  - mainmodulertooverlapping, 37
- calculateoverlappingsection
  - mainmodulertooverlapping, 38
- calculateqo
  - dlllooverlapping, 8
- calculateqof
  - dlllooverlapping, 9
- calculateqorto
  - zfunctionswtioverlapping, 55
- calculatesegmentslopes
  - geometrymodulertooverlapping, 29
- calculatetanalpha
  - factormodulertooverlapping, 15
- calculatewavelength
  - formulamodulertooverlapping, 19
- calculatewaveoverlapping
  - mainmodulertooverlapping, 39
- calculatewaveoverlappingdischarge
  - formulamodulertooverlapping, 19
- calculatewaverunup
  - formulamodulertooverlapping, 20
- calculatewavesteepness
  - formulamodulertooverlapping, 21
- calczvalue
  - dlllooverlapping, 10
- checkcrosssection
  - geometrymodulertooverlapping, 30
- checkinputdata
  - mainmodulertooverlapping, 40
- checkmodelfactors
  - mainmodulertooverlapping, 41
- computedoverlapping
  - typedefinitionsrtooverlapping::tpoverlappinginput, 70
- convergedwithresidu
  - waverunup, 51
- convertoverlappinginput
  - mainmodulertooverlapping, 41
- copygeometry
  - geometrymodulertooverlapping, 30
- criticaloverlapping
  - typedefinitionsrtooverlapping::tpoverlappinginput, 70
- cubicroots
  - formulamodulertooverlapping, 21
- currentlogging
  - modulelogging, 43
- deallocategeometry
  - geometrymodulertooverlapping, 31
- determinesegmenttypes
  - geometrymodulertooverlapping, 31
- determinestartingvalue
  - waverunup, 51
- dllOverlapping.f90, 75
- dlllooverlapping, 7
  - calculateqo, 8
  - calculateqof, 9
  - calczvalue, 10
  - geometry\_c\_f, 10
  - getlanguage, 11
  - setlanguage, 11
  - validateinputc, 11
  - validateinputf, 12
  - versionnumber, 13

- factorModuleRTOovertopping.f90, 76
- factordeterminationq\_b\_f\_b
  - typedefinitionsrtooverlapping::tpovertoppinginput, 70
- factordeterminationq\_b\_f\_n
  - typedefinitionsrtooverlapping::tpovertoppinginput, 70
- factormodulertooverlapping, 13
  - calculategammab, 13
  - calculategammabeta, 14
  - calculategammaf, 14
  - calculatetalpha, 15
- fb\_max
  - typedefinitionsrtooverlapping, 48
- fb\_min
  - typedefinitionsrtooverlapping, 48
- filename
  - modulelogging::tlogging, 63
- findsmallestresidu
  - waverunup, 51
- fn\_max
  - typedefinitionsrtooverlapping, 48
- fn\_min
  - typedefinitionsrtooverlapping, 48
- foreshore\_max
  - typedefinitionsrtooverlapping, 48
- foreshore\_min
  - typedefinitionsrtooverlapping, 48
- formulaModuleRTOovertopping.f90, 76
- formulamodulertooverlapping, 16
  - adjustinfluencefactors, 17
  - calculateanglewaveattack, 17
  - calculatebreakerlimit, 18
  - calculatebreakerparameter, 18
  - calculatewavelength, 19
  - calculatewaveovertoppingdischarge, 19
  - calculatewaverunup, 20
  - calculatewavesteepness, 21
  - cubicroots, 21
  - isequalreal, 22
  - isequalzero, 22
  - realrootscubicfunction, 22
  - rootsdepressedcubic, 23
  - rootsgeneralcubic, 23
- frunup1
  - typedefinitionsrtooverlapping::tpovertoppinginput, 70
- frunup1\_max
  - typedefinitionsrtooverlapping, 48
- frunup1\_min
  - typedefinitionsrtooverlapping, 48
- frunup2
  - typedefinitionsrtooverlapping::tpovertoppinginput, 70
- frunup2\_max
  - typedefinitionsrtooverlapping, 49
- frunup2\_min
  - typedefinitionsrtooverlapping, 49
- frunup3
  - typedefinitionsrtooverlapping::tpovertoppinginput, 70
- frunup3\_max
  - typedefinitionsrtooverlapping, 49
- frunup3\_min
  - typedefinitionsrtooverlapping, 49
- fs\_max
  - typedefinitionsrtooverlapping, 49
- fs\_min
  - typedefinitionsrtooverlapping, 49
- fshallow
  - typedefinitionsrtooverlapping::tpovertoppinginput, 71
- geometry\_c\_f
  - dllovertopping, 10
- geometryModuleRTOovertopping.f90, 77
- geometrymodulertooverlapping, 25
  - adjustnonhorizontalberms, 26
  - allocatevectorsgeometry, 26
  - basicgeometrytest, 27
  - calculatehorzdistance, 28
  - calculatehorzlengths, 28
  - calculatesegmentslopes, 29
  - checkcrosssection, 30
  - copygeometry, 30
  - deallocategeometry, 31
  - determinesegmenttypes, 31
  - initializegeometry, 32
  - mergesequentialberms, 32
  - removeberms, 34
  - removedikesegments, 35
  - splitcrosssection, 35
- getlanguage
  - dllovertopping, 11
  - overtoppingmessages, 44
- getovertoppingformat
  - overtoppingmessages, 44
- getovertoppingmessage
  - overtoppingmessages, 45
- getovertoppingparameter
  - overtoppingmessages, 45
- h
  - typedefinitionsrtooverlapping::tpload, 66
- hm0
  - typedefinitionsrtooverlapping::tpload, 66
- initializegeometry
  - geometrymodulertooverlapping, 32
- innercalculation
  - waverunup, 52
- interpolateresultssections
  - mainmodulertooverlapping, 42
- isequalreal
  - formulamodulertooverlapping, 22
- isequalzero
  - formulamodulertooverlapping, 22

- iterationwaverunup
  - waverunup, 53
- language
  - overtoppingmessages, 46
- m\_z2
  - typedefinitionsrtooverlapping::tpoverlappinginput, 71
- mainModuleRTOoverlapping.f90, 79
- mainmodulertooverlapping, 36
  - calculateoverlapping, 37
  - calculateoverlappingnegativefreeboard, 37
  - calculateoverlappingsection, 38
  - calculatewaveoverlapping, 39
  - checkinputdata, 40
  - checkmodelfactors, 41
  - converttooverlappinginput, 41
  - interpolateresultssections, 42
- margindiff
  - typedefinitionsrtooverlapping, 49
- margingrad
  - typedefinitionsrtooverlapping, 49
- maxfilenamelenh
  - modulelogging, 43
- maxmsg
  - overtoppingmessages, 46
- maxpar
  - overtoppingmessages, 46
- mergesquentialberms
  - geometrymodulertooverlapping, 32
- ModuleLogging.f90, 79
- modulelogging, 43
  - currentlogging, 43
  - maxfilenamelenh, 43
- modulelogging::tlogging, 62
  - filename, 63
  - verbosity, 63
- mz2\_max
  - typedefinitionsrtooverlapping, 49
- mz2\_min
  - typedefinitionsrtooverlapping, 50
- nbermssegments
  - typedefinitionsrtooverlapping::tpgeometry, 64
- ncoordinates
  - typedefinitionsrtooverlapping::tpgeometry, 64
- normal
  - overtoppinginterface::overtoppinggeometrytype, 60
  - overtoppinginterface::overtoppinggeometrytypef, 61
- npoints
  - overtoppinginterface::overtoppinggeometrytype, 60
  - overtoppinginterface::overtoppinggeometrytypef, 61
- overtoppingInterface.f90, 80
- OvertoppingMessages.f90, 80
- overtoppinginterface, 43
  - varmodelfactorcriticaloverlapping, 44
- overtoppinginterface::overtoppinggeometrytype, 59
  - normal, 60
  - npoints, 60
  - roughness, 60
  - xcoords, 60
  - ycoords, 60
- overtoppinginterface::overtoppinggeometrytypef, 61
  - normal, 61
  - npoints, 61
  - roughness, 61
  - xcoords, 62
  - ycoords, 62
- overtoppinginterface::tpprofilecoordinate, 72
  - roughness, 72
  - xcoordinate, 72
  - zcoordinate, 73
- overtoppingmessages, 44
  - getlanguage, 44
  - getoverlappingformat, 44
  - getoverlappingmessage, 45
  - getoverlappingparameter, 45
  - language, 46
  - maxmsg, 46
  - maxpar, 46
  - setlanguage, 45
- phi
  - typedefinitionsrtooverlapping::tpload, 67
- profileinstructure
  - zfunctionswtiooverlapping, 55
- psi
  - typedefinitionsrtooverlapping::tpgeometry, 64
- qo
  - typedefinitionsrtooverlapping::tpoverlapping, 68
- realrootscubicfunction
  - formulamodulertooverlapping, 22
- reductionfactorforeshore
  - typedefinitionsrtooverlapping::tpoverlappinginput, 71
- relaxationfactor
  - typedefinitionsrtooverlapping::tpoverlappinginput, 71
- removeberms
  - geometrymodulertooverlapping, 34
- removedikeselements
  - geometrymodulertooverlapping, 35
- rfactor\_max
  - typedefinitionsrtooverlapping, 50
- rfactor\_min
  - typedefinitionsrtooverlapping, 50
- rootsdepressedcubic
  - formulamodulertooverlapping, 23
- rootsgeneralcubic
  - formulamodulertooverlapping, 23

- roughness
  - overtoppinginterface::overtoppinggeometrytype, 60
  - overtoppinginterface::overtoppinggeometrytypef, 61
  - overtoppinginterface::tpprofilecoordinate, 72
- roughnessfactors
  - typedefinitionsrtooverlapping::tpgeometry, 64
- segmentslopes
  - typedefinitionsrtooverlapping::tpgeometry, 64
- segmenttypes
  - typedefinitionsrtooverlapping::tpgeometry, 65
- setlanguage
  - dllovertopping, 11
  - overtoppingmessages, 45
- slope\_max
  - typedefinitionsrtooverlapping, 50
- slope\_min
  - typedefinitionsrtooverlapping, 50
- splitcrosssection
  - geometrymodulertooverlapping, 35
- tm\_10
  - typedefinitionsrtooverlapping::tpload, 67
- typeDefinitionsRTOoverlapping.f90, 81
- typedefinitionsrtooverlapping, 46
  - berm\_max, 48
  - berm\_min, 48
  - fb\_max, 48
  - fb\_min, 48
  - fn\_max, 48
  - fn\_min, 48
  - foreshore\_max, 48
  - foreshore\_min, 48
  - frunup1\_max, 48
  - frunup1\_min, 48
  - frunup2\_max, 49
  - frunup2\_min, 49
  - frunup3\_max, 49
  - frunup3\_min, 49
  - fs\_max, 49
  - fs\_min, 49
  - margindiff, 49
  - margingrad, 49
  - mz2\_max, 49
  - mz2\_min, 50
  - rfactor\_max, 50
  - rfactor\_min, 50
  - slope\_max, 50
  - slope\_min, 50
  - xdiff\_min, 50
  - z2\_iter\_max1, 50
  - z2\_iter\_max2, 50
  - z2\_margin, 50
- typedefinitionsrtooverlapping::tpgeometry, 63
  - nbermsegments, 64
  - ncoordinates, 64
  - psi, 64
  - roughnessfactors, 64
  - segmentslopes, 64
  - segmenttypes, 65
  - xcoorddiff, 65
  - xcoordinates, 65
  - ycoorddiff, 65
  - ycoordinates, 65
- typedefinitionsrtooverlapping::tpload, 65
  - h, 66
  - hm0, 66
  - phi, 67
  - tm\_10, 67
- typedefinitionsrtooverlapping::tpoverlapping, 67
  - qo, 68
  - z2, 68
- typedefinitionsrtooverlapping::tpoverlappinginput, 68
  - computedoverlapping, 70
  - criticaloverlapping, 70
  - factordeterminationq\_b\_f\_b, 70
  - factordeterminationq\_b\_f\_n, 70
  - frunup1, 70
  - frunup2, 70
  - frunup3, 70
  - fshallow, 71
  - m\_z2, 71
  - reductionfactorforeshore, 71
  - relaxationfactor, 71
  - typerunup, 71
- typerunup
  - typedefinitionsrtooverlapping::tpoverlappinginput, 71
- validateinputc
  - dllovertopping, 11
- validateinputf
  - dllovertopping, 12
- varmodelfactorcriticaloverlapping
  - overtoppinginterface, 44
- verbosity
  - modulelogging::tlogging, 63
- versionnumber
  - dllovertopping, 13
- waveRunup.f90, 83
- waverunup, 51
  - convergedwithresidu, 51
  - determinestartingvalue, 51
  - findsmallestresidu, 51
  - innercalculation, 52
  - iterationwaverunup, 53
- xcoorddiff
  - typedefinitionsrtooverlapping::tpgeometry, 65
- xcoordinate
  - overtoppinginterface::tpprofilecoordinate, 72
- xcoordinates
  - typedefinitionsrtooverlapping::tpgeometry, 65
- xcoords



- overtoppinginterface::overtoppinggeometrytype,  
[60](#)
  - overtoppinginterface::overtoppinggeometrytypef,  
[62](#)
- xdiff\_min
  - typedefinitionsrtoovertopping, [50](#)
- ycoorddiff
  - typedefinitionsrtoovertopping::tpgeometry, [65](#)
- ycoordinates
  - typedefinitionsrtoovertopping::tpgeometry, [65](#)
- ycoords
  - overtoppinginterface::overtoppinggeometrytype,  
[60](#)
  - overtoppinginterface::overtoppinggeometrytypef,  
[62](#)
- z2
  - typedefinitionsrtoovertopping::tpovertopping, [68](#)
- z2\_iter\_max1
  - typedefinitionsrtoovertopping, [50](#)
- z2\_iter\_max2
  - typedefinitionsrtoovertopping, [50](#)
- z2\_margin
  - typedefinitionsrtoovertopping, [50](#)
- zFunctionsWTIOvertopping.f90, [83](#)
- zcoordinate
  - overtoppinginterface::tpprofilecoordinate, [73](#)
- zfunclogratios
  - zfunctionswtiovertopping, [56](#)
- zfunctionswtiovertopping, [53](#)
  - adjustprofile, [54](#)
  - calculateqorto, [55](#)
  - profileinstructure, [55](#)
  - zfunclogratios, [56](#)