

Dikes Overtopping Kernel - Technical documentation

Generated by Doxygen 1.8.9.1

Wed May 4 2016 12:34:15

Contents

1	Modules Index	1
1.1	Modules List	1
2	Data Type Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
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	omkeervariantf	11
4.1.2.7	setlanguage	12
4.1.2.8	validateinputc	12
4.1.2.9	validateinputf	13
4.1.2.10	versionnumber	13
4.2	factormodulertooverlapping Module Reference	13
4.2.1	Detailed Description	14
4.2.2	Function/Subroutine Documentation	14
4.2.2.1	calculategammab	14
4.2.2.2	calculategammabeta	15
4.2.2.3	calculategammaf	16
4.2.2.4	calculatetanalpha	17
4.3	formulamodulertooverlapping Module Reference	18
4.3.1	Detailed Description	19

4.3.2	Function/Subroutine Documentation	19
4.3.2.1	adjustinfluencefactors	19
4.3.2.2	calculateanglewaveattack	20
4.3.2.3	calculatebreakerlimit	20
4.3.2.4	calculatebreakerparameter	20
4.3.2.5	calculatewavelength	21
4.3.2.6	calculatewaveovertoppingdischarge	21
4.3.2.7	calculatewaverunup	22
4.3.2.8	calculatewavesteepness	23
4.3.2.9	cubicroots	24
4.3.2.10	isequalreal	24
4.3.2.11	isequalzero	24
4.3.2.12	realrootscubicfunction	25
4.3.2.13	rootsdepressedcubic	26
4.3.2.14	rootsgeneralcubic	27
4.4	geometrymodulertooverlapping Module Reference	28
4.4.1	Detailed Description	29
4.4.2	Function/Subroutine Documentation	29
4.4.2.1	adjustnonhorizontalberms	29
4.4.2.2	allocatevectorsgeometry	30
4.4.2.3	basicgeometrytest	31
4.4.2.4	calculatehorzdistance	32
4.4.2.5	calculatehorzlengths	32
4.4.2.6	calculatesegmentslopes	33
4.4.2.7	checkcrosssection	34
4.4.2.8	copygeometry	34
4.4.2.9	deallocategeometry	35
4.4.2.10	determinesegmenttypes	35
4.4.2.11	initializegeometry	36
4.4.2.12	mergesequentialberms	37
4.4.2.13	removeberms	38
4.4.2.14	removedikesegments	39
4.4.2.15	splitcrosssection	39
4.5	mainmodulertooverlapping Module Reference	40
4.5.1	Detailed Description	41
4.5.2	Function/Subroutine Documentation	41
4.5.2.1	calculateovertopping	41
4.5.2.2	calculateovertoppingnegativefreeboard	42
4.5.2.3	calculateovertoppingsection	43
4.5.2.4	calculatewaveovertopping	44

4.5.2.5	checkinputdata	45
4.5.2.6	checkmodelfactors	46
4.5.2.7	interpolateresultssections	46
4.6	modulelogging Module Reference	47
4.6.1	Detailed Description	47
4.6.2	Variable Documentation	48
4.6.2.1	currentlogging	48
4.6.2.2	maxfilenamelength	48
4.7	omkeervariantmodule Module Reference	48
4.7.1	Detailed Description	48
4.7.2	Function/Subroutine Documentation	48
4.7.2.1	iteratetogivendischarge	48
4.7.2.2	iteratetogivendischargevalidprofile	49
4.8	overtoppinginterface Module Reference	50
4.8.1	Detailed Description	50
4.8.2	Variable Documentation	51
4.8.2.1	varmodelfactorcriticalovertopping	51
4.9	overtoppingmessages Module Reference	51
4.9.1	Detailed Description	51
4.9.2	Function/Subroutine Documentation	51
4.9.2.1	getlanguage	51
4.9.2.2	getovertoppingformat	51
4.9.2.3	getovertoppingmessage	52
4.9.2.4	getovertoppingparameter	52
4.9.2.5	setlanguage	53
4.9.3	Variable Documentation	53
4.9.3.1	language	53
4.9.3.2	maxmsg	53
4.9.3.3	maxpar	53
4.10	typedefinitionsrtooverlapping Module Reference	53
4.10.1	Detailed Description	55
4.10.2	Variable Documentation	55
4.10.2.1	berm_max	55
4.10.2.2	berm_min	55
4.10.2.3	fb_max	55
4.10.2.4	fb_min	55
4.10.2.5	fn_max	55
4.10.2.6	fn_min	55
4.10.2.7	foreshore_max	55
4.10.2.8	foreshore_min	55

4.10.2.9	frunup1	56
4.10.2.10	frunup2	56
4.10.2.11	frunup3	56
4.10.2.12	fs_max	56
4.10.2.13	fs_min	56
4.10.2.14	margindiff	56
4.10.2.15	margingrad	56
4.10.2.16	mz2_max	56
4.10.2.17	mz2_min	56
4.10.2.18	rfactor_max	56
4.10.2.19	rfactor_min	57
4.10.2.20	slope_max	57
4.10.2.21	slope_min	57
4.10.2.22	xdiff_min	57
4.10.2.23	z2_iter_max1	57
4.10.2.24	z2_iter_max2	57
4.10.2.25	z2_margin	57
4.11	waverunup Module Reference	57
4.11.1	Detailed Description	58
4.11.2	Function/Subroutine Documentation	58
4.11.2.1	convergedwithresidu	58
4.11.2.2	determinestartingvalue	58
4.11.2.3	findsmallestresidu	58
4.11.2.4	innercalculation	59
4.11.2.5	iterationwaverunup	59
4.12	zfunctionswtiovertopping Module Reference	60
4.12.1	Detailed Description	61
4.12.2	Function/Subroutine Documentation	61
4.12.2.1	adjustprofile	61
4.12.2.2	calculateqorto	62
4.12.2.3	profileinstructure	63
4.12.2.4	zfunclogratios	64
5	Data Type Documentation	67
5.1	overtoppinginterface::overtoppinggeometrytype Type Reference	67
5.1.1	Detailed Description	68
5.1.2	Member Data Documentation	68
5.1.2.1	normal	68
5.1.2.2	npoints	68
5.1.2.3	roughness	68

5.1.2.4	xcoords	68
5.1.2.5	ycoords	68
5.2	overtoppinginterface::overtoppinggeometrytypef Type Reference	69
5.2.1	Detailed Description	69
5.2.2	Member Data Documentation	69
5.2.2.1	normal	69
5.2.2.2	npoints	69
5.2.2.3	roughness	70
5.2.2.4	xcoords	70
5.2.2.5	ycoords	70
5.3	modulelogging::tlogging Type Reference	70
5.3.1	Detailed Description	70
5.3.2	Member Data Documentation	71
5.3.2.1	filename	71
5.3.2.2	verbosity	71
5.4	typedefinitionsrtooverlapping::tpgeometry Type Reference	71
5.4.1	Detailed Description	72
5.4.2	Member Data Documentation	72
5.4.2.1	nbermsements	72
5.4.2.2	ncoordinates	72
5.4.2.3	psi	72
5.4.2.4	roughnessfactors	72
5.4.2.5	segmentslopes	73
5.4.2.6	segmenttypes	73
5.4.2.7	xcoorddiff	73
5.4.2.8	xcoordinates	73
5.4.2.9	ycoorddiff	73
5.4.2.10	ycoordinates	73
5.5	typedefinitionsrtooverlapping::tpload Type Reference	73
5.5.1	Detailed Description	74
5.5.2	Member Data Documentation	74
5.5.2.1	h	74
5.5.2.2	hm0	75
5.5.2.3	phi	75
5.5.2.4	tm_10	75
5.6	typedefinitionsrtooverlapping::tpoverlapping Type Reference	75
5.6.1	Detailed Description	76
5.6.2	Member Data Documentation	76
5.6.2.1	qo	76
5.6.2.2	z2	76

5.7	typedefinitionsrtovertopping::tpvertoppinginput Type Reference	76
5.7.1	Detailed Description	78
5.7.2	Member Data Documentation	78
5.7.2.1	computedvertopping	78
5.7.2.2	criticalvertopping	78
5.7.2.3	factordeterminationq_b_f_b	78
5.7.2.4	factordeterminationq_b_f_n	78
5.7.2.5	fshallow	78
5.7.2.6	m_z2	78
5.7.2.7	reductionfactorforeshore	78
5.7.2.8	relaxationfactor	79
5.8	vertoppinginterface::tpprofilecoordinate Type Reference	79
5.8.1	Detailed Description	79
5.8.2	Member Data Documentation	79
5.8.2.1	roughness	80
5.8.2.2	xcoordinate	80
5.8.2.3	zcoordinate	80
6	File Documentation	81
6.1	dllOvertopping.f90 File Reference	81
6.1.1	Detailed Description	82
6.2	factorModuleRTOvertopping.f90 File Reference	82
6.2.1	Detailed Description	82
6.3	formulaModuleRTOvertopping.f90 File Reference	82
6.3.1	Detailed Description	83
6.4	geometryModuleRTOvertopping.f90 File Reference	83
6.4.1	Detailed Description	84
6.5	mainModuleRTOvertopping.f90 File Reference	85
6.5.1	Detailed Description	85
6.6	ModuleLogging.f90 File Reference	85
6.6.1	Detailed Description	86
6.7	omkeerVariantModule.f90 File Reference	86
6.7.1	Detailed Description	86
6.8	vertoppingInterface.f90 File Reference	86
6.8.1	Detailed Description	87
6.9	OvertoppingMessages.f90 File Reference	87
6.9.1	Detailed Description	87
6.10	typeDefinitionsRTOvertopping.f90 File Reference	87
6.10.1	Detailed Description	89
6.11	waveRunup.f90 File Reference	89

6.11.1 Detailed Description	89
6.12 zFunctionsWTIOvertopping.f90 File Reference	89
6.12.1 Detailed Description	90
Index	91

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

dllovertopping	Main entry for the dll DikesOvertopping	7
factormodulertoovertopping	Functions for the slope angle and influence factors	13
formulamodulertoovertopping	Core computations for Dikes Overtopping	18
geometrymodulertoovertopping	Core computations related to the geometry	28
mainmodulertoovertopping	Core computations for Dikes Overtopping	40
modulelogging	Steering the extra logging	47
omkeervariantmodule	Module for the 'omkeerVariant'	48
overtoppinginterface	Module for the interface of dllOvertopping	50
overtoppingmessages	Module for the messages in the overtopping dll, in Dutch or English	51
typedefinitionsrtoovertopping	Type definitions for Dikes Overtopping	53
waverunup	Iteration procedure for 2% wave runup	57
zfunctionswtiovertopping	Module for the Limit State Functions (Z-functions) for wave overtopping	60

Chapter 2

Data Type Index

2.1 Class List

Here are the data types with brief descriptions:

overtoppinginterface::overtoppinggeometrytype	67
overtoppinginterface::overtoppinggeometrytypef	69
modulelogging::tlogging	
TLogging: structure for steering the logging	70
typedefinitionsrtoovertopping::tpgeometry	
TpGeometry: structure with geometry data	71
typedefinitionsrtoovertopping::tpload	
TpLoad: structure with load parameters	73
typedefinitionsrtoovertopping::tpovertopping	
TpOvertopping: structure with overtopping results	75
typedefinitionsrtoovertopping::tpovertoppinginput	
OvertoppingModelFactors: C-structure with model factors	76
overtoppinginterface::tpprofilecoordinate	79

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

dllOvertopping.f90	Main entry for the dll DikesOvertopping FUNCTIONS/SUBROUTINES exported from dll↔ Overtopping.dll:	81
factorModuleRTOvertopping.f90	This file contains a module with functions for the slope angle and influence factors	82
formulaModuleRTOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping	82
geometryModuleRTOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping related to the geometry	83
mainModuleRTOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping	85
ModuleLogging.f90	Module for steering the extra logging	85
omkeerVariantModule.f90	This file contains the omkeerVariant	86
overtoppingInterface.f90	This file contains the parameters and types (structs) as part of the interface to and from dll↔ Overtopping	86
OvertoppingMessages.f90	This file contains the messages in the overtopping dll, in Dutch or English	87
typeDefinitionsRTOvertopping.f90	This file contains a module with the type definitions for Dikes Overtopping	87
waveRunup.f90	This file contains a module with the iteration procedure for 2% wave runup	89
zFunctionsWTIOvertopping.f90	This file contains the limit state functions for wave overtopping within WTI	89

Chapter 4

Module Documentation

4.1 dllovertopping Module Reference

Main entry for the dll DikesOvertopping.

Functions/Subroutines

- subroutine, public [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 [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 [omkeervariantf](#) (load, geometryF, givenDischarge, dikeHeight, modelFactors, overtopping, success, errorText, logging)
Subroutine with omkeerVariant.
- 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

Main entry for the dll DikesOvertopping.

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*)

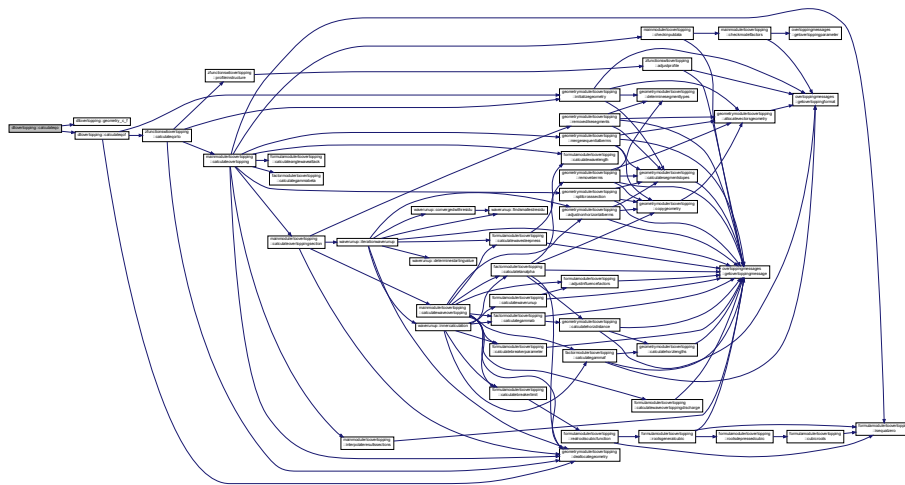
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF: convert C-like input structures to Fortran input structures.

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 44 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*)

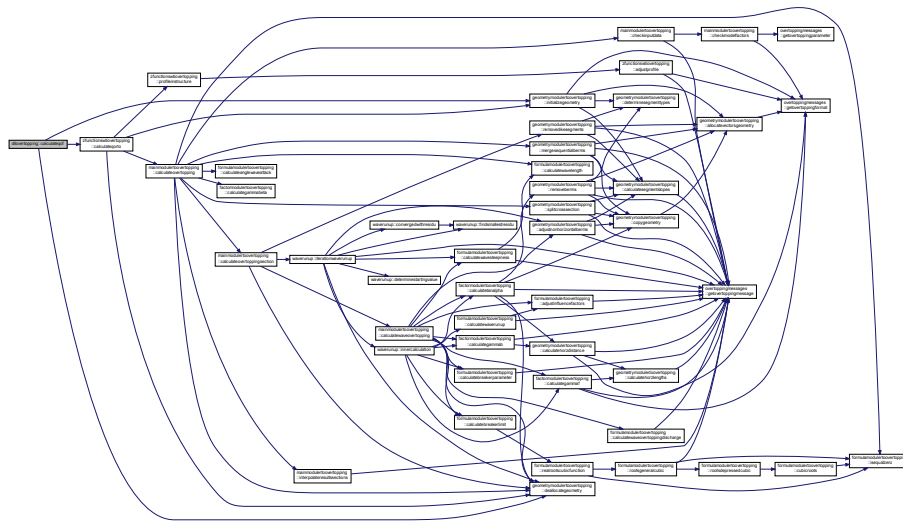
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.

Parameters

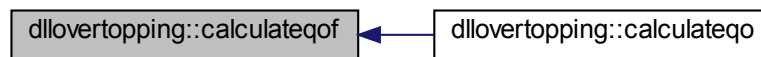
in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelFactors
out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success
out	<i>errortext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

Definition at line 75 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*)

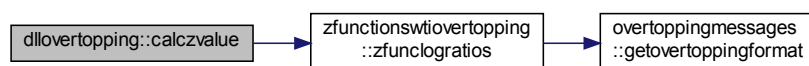
Subroutine that calculates the Z-function Dikes Overtopping based on the discharge calculated with calculateQoF.

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 108 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

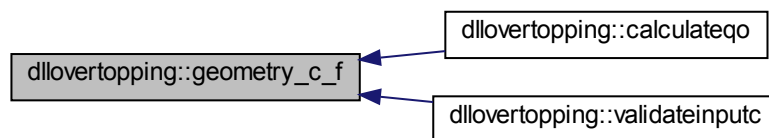
in	<i>geometryinput</i>	struct with geometry and roughness as c-pointers
----	----------------------	--

Returns

fortran struct with geometry and roughness

Definition at line 331 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 297 of file dllovertopping.f90.

4.1.2.6 `subroutine, public dllovertopping::omkeervariantf (type(tpload), intent(in) load, type(overtoppinggeometrytype), intent(in) geometryF, real(kind=wp), intent(in) givenDischarge, real(kind=wp), intent(out) dikeHeight, type(tpovertoppinginput), intent(inout) modelFactors, type(tpovertopping), intent(inout) overtopping, logical, intent(out) success, character(len=*), intent(out) errorText, type(tlogging), intent(in) logging)`

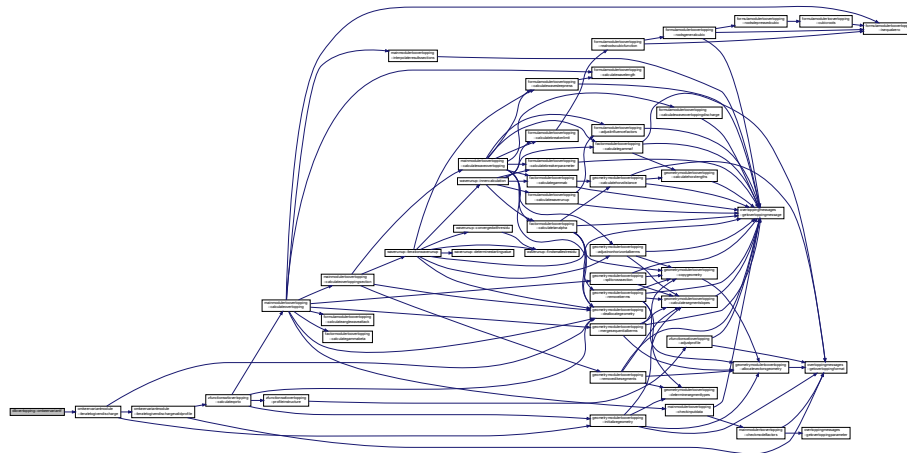
Subroutine with omkeerVariant.

Parameters

in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>givedischarge</i>	discharge to iterate to
out	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelFactors
in, 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 259 of file dllovertopping.f90.

Here is the call graph for this function:



4.1.2.7 subroutine, public dlovertopping::setlanguage (character(len=*), intent(in) lang)

Subroutine that sets the language for error and validation messages.

Definition at line 284 of file dllOvertopping.f90.

4.1.2.8 subroutine, public dlovertopping::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)

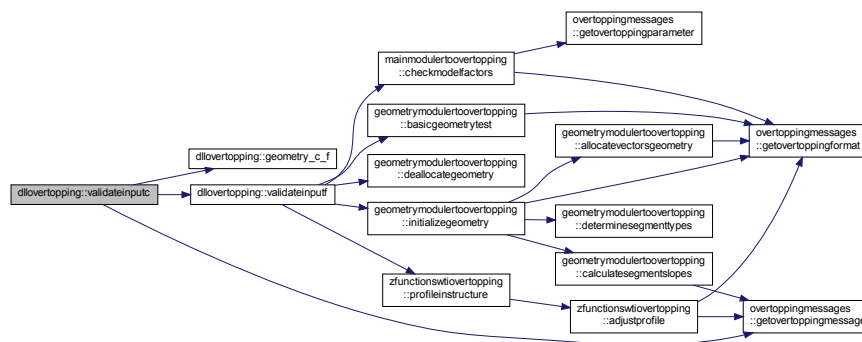
Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.

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 128 of file dllOvertopping.f90.

Here is the call graph for this function:



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

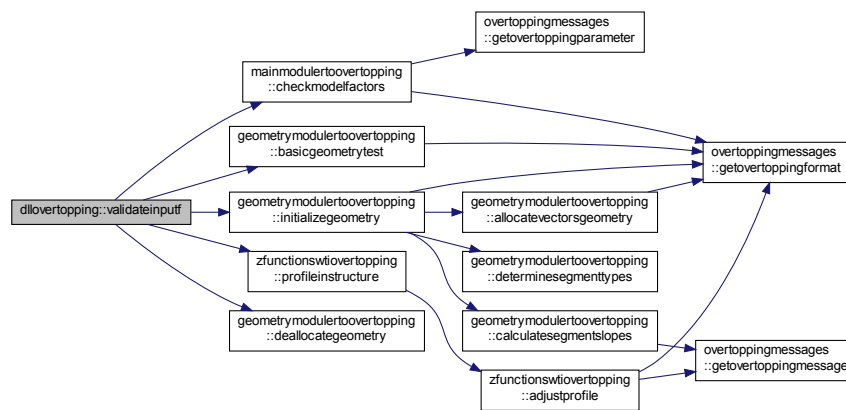
Subroutine that validates the geometry.

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 179 of file dllOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.2.10 subroutine, public dllovertopping::versionnumber (character(len=*), intent(out) *version*)

Subroutine that delivers the version number.

Parameters

out	<i>version</i>	version number
-----	----------------	----------------

Definition at line 309 of file dllOvertopping.f90.

4.2 factormodulertoovertopping Module Reference

functions for the slope angle and influence factors

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, ksi0Limit, gammaB, z2, geometry, gammaF, succes, errorMessage)
calculateGammaF influence factor roughness
- subroutine, public [calculategammab](#) (h, Hm0, z2, geometry, gammaB, succes, errorMessage)
calculateGammaB influence factor berms

4.2.1 Detailed Description

functions for the slope angle and influence factors

4.2.2 Function/Subroutine Documentation

- 4.2.2.1 subroutine, public `factormoduletoovertopping::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)`

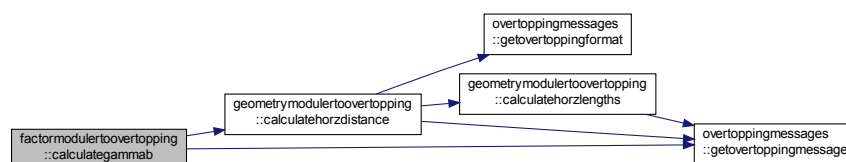
calculateGammaB influence factor berms

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 289 of file `factorModuleRTOOvertopping.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.2 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*)

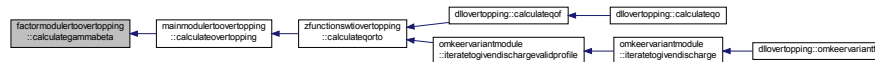
calculateGammaBeta influence factor angle of wave attack

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 118 of file factorModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.2.2.3 subroutine, public factormodulertooveropping::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*)

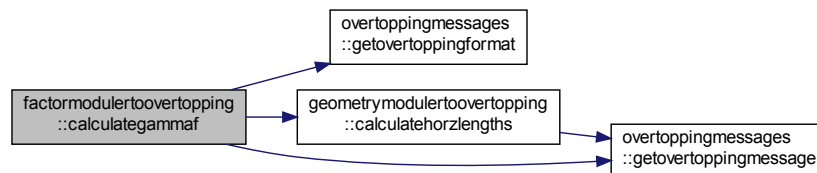
calculateGammaF influence factor roughness

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 158 of file factorModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.4 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*)

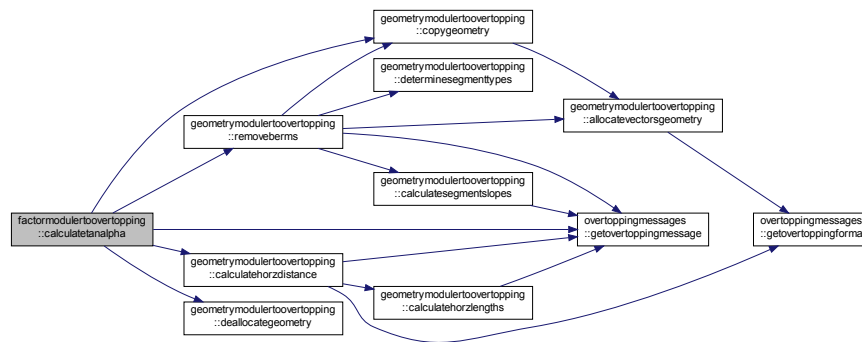
calculateTanAlpha representative slope angle

Parameters

in	h	local water level (m+NAP)
in	$hm0$	significant wave height (m)
in	$z2$	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 39 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

the core computations for Dikes Overtopping

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](#) (gammaB, ksiOLimit, succes, errorMessage)
 - calculateBreakerLimit*: calculate the breaker limit
- subroutine, public [adjustInfluenceFactors](#) (gammaB, gammaF, gammaBeta, gammaBetaType, ksi0, ksiOLimit, 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 Detailed Description

the core computations for Dikes Overtopping

4.3.2 Function/Subroutine Documentation

- 4.3.2.1 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) *ksiOLimit*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

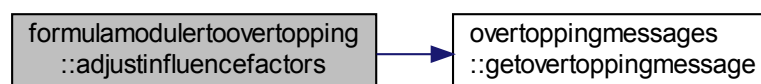
adjustInfluenceFactors: adjust the influence factors

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>ksiOLimit</i>	limit value breaker parameter
out	<i>succes</i>	flag for succes
out	<i>errorMessage</i>	error message

Definition at line 379 of file formulaModuleRTOOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.2 subroutine, public formulamodulertooverlapping::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 288 of file formulaModuleRTOoverlapping.f90.

Here is the caller graph for this function:

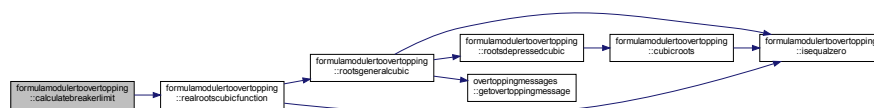


4.3.2.3 subroutine, public formulamodulertooverlapping::calculatebreakerlimit (real(wp), intent(in) *gammaB*, real(wp), intent(out) *ksi0Limit*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

calculateBreakerLimit: calculate the breaker limit

Definition at line 311 of file formulaModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



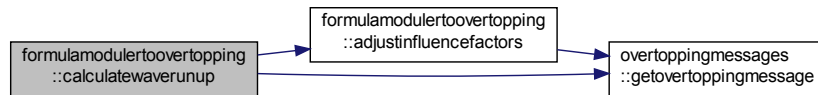
4.3.2.4 subroutine, public formulamodulertooverlapping::calculatebreakerparameter (real(wp), intent(in) *tanAlpha*, real(wp), intent(in) *s0*, real(wp), intent(out) *ksi0*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

calculateBreakerParameter: calculate the breaker parameter

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

Definition at line 38 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



```
4.3.2.8 subroutine, public formulamodulertooverlapping::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 )
```

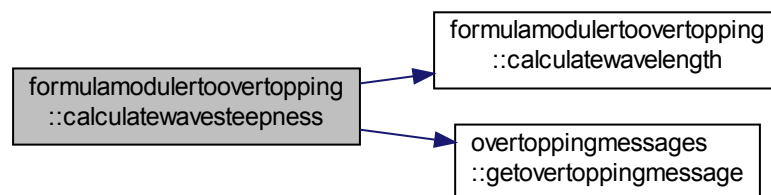
calculateWaveSteepness: calculate the wave steepness

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 205 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.9 subroutine formulamodulertovertopping::cubicroots (double complex, intent(in) z, double complex, dimension(3), intent(out) roots) [private]

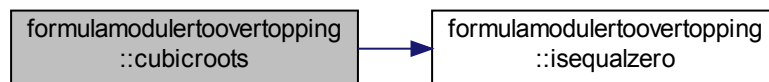
cubicRoots: calculate the roots of a cubic function

Parameters

in	z	complex number
out	roots	cubic roots

Definition at line 627 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.10 logical function, public formulamodulertovertopping::isequalreal (real(wp), intent(in) x1, real(wp), intent(in) x2)

isEqualReal: are two reals (almost) equal

Parameters

in	x1	first real
in	x2	second real

Definition at line 668 of file formulaModuleRTOovertopping.f90.

4.3.2.11 logical function, public formulamodulertovertopping::isequalzero (real(wp), intent(in) x)

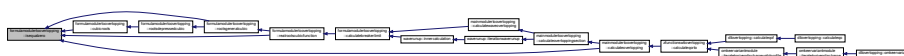
isEqualZero: is a real (almost) zero

Parameters

in	x	real number
----	---	-------------

Definition at line 692 of file formulaModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.3.2.12 subroutine formulamodulertoovertopping::realrootscubicfunction (real(wp), intent(in) *a*, real(wp), intent(in) *b*,
real(wp), intent(in) *c*, real(wp), intent(in) *d*, integer, intent(out) *N*, real(wp), dimension(3), intent(out) *x*, logical,
intent(out) *succes*, character(len=*), intent(out) *errorMessage*) [private]

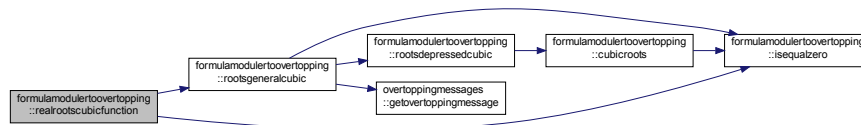
realRootsCubicFunction: calculate the roots of a cubic function

Parameters

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

Definition at line 477 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.13 subroutine formulaModuleRTOovertopping::rootsdepressedcubic (real(wp), intent(in) p , real(wp), intent(in) q , double complex, dimension(3), intent(out) z) [private]

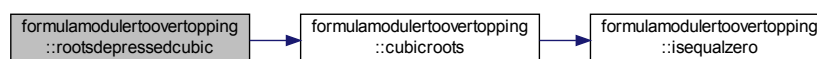
rootsDepressedCubic: calculate the roots of a depressed cubic function

Parameters

in	p	coefficient p depressed cubic
in	q	coefficient q depressed cubic
out	z	roots depressed cubic

Definition at line 586 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.14 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) *succes*, character(len=*), intent(out) *errorMessage*) [private]

rootsGeneralCubic: calculate the roots of a generic cubic function

adjustNonHorizontalBerms: adjust non-horizontal berms

- subroutine, public [removeberms](#) (geometry, geometryNoBerms, succes, errorMessage)
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 Detailed Description

core computations related to the geometry

4.4.2 Function/Subroutine Documentation

4.4.2.1 subroutine, public `geometrymodulertovertopping::adjustnonhorizontalberms` (type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(out) *geometryFlatBerms*, logical, intent(out) *success*, character(len=*) , intent(out) *errorMessage*)

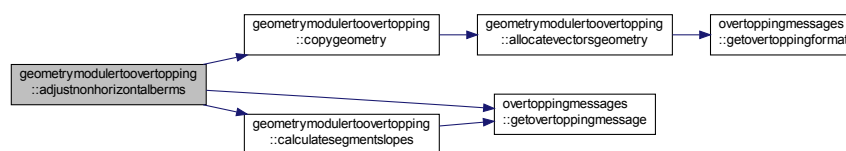
adjustNonHorizontalBerms: adjust non-horizontal berms

Parameters

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

Definition at line 529 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.2 subroutine, public geometrymodulertooverlapping::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>succes</i>	succes flag
in, out	<i>errormessage</i>	error message (only set in case of error)

Definition at line 225 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

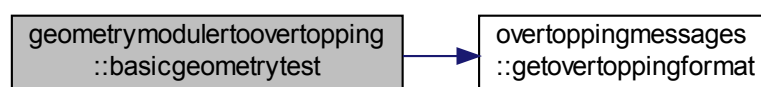
basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)

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 1032 of file geometryModuleRTOovertopping.f90.

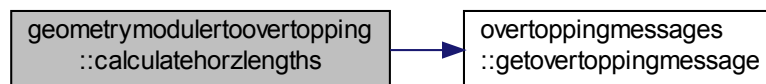
Here is the call graph for this function:



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 887 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.6 subroutine, public geometrymodulertoovertopping::calculatesegmentsslopes (type (tpgeometry), intent(inout) *geometry*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

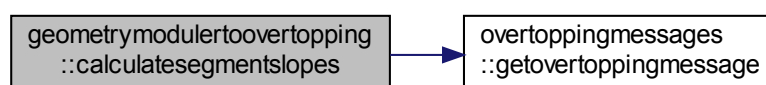
calculateSegmentSlopes: calculate the segment slopes

Parameters

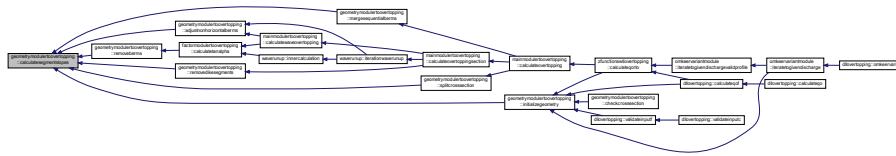
in, out	<i>geometry</i>	structure with geometry data
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 287 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.7 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*)

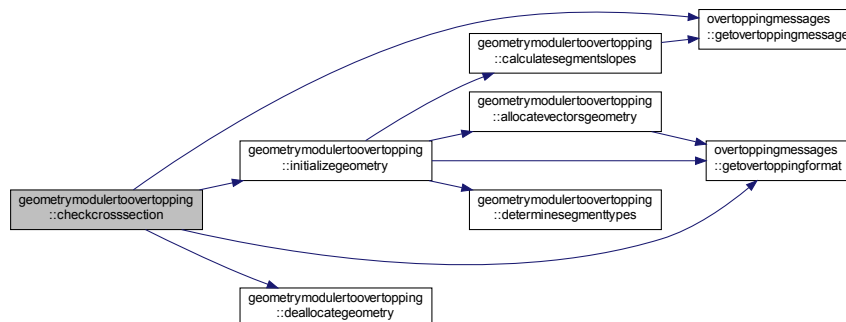
checkCrossSection: check cross section

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 38 of file geometryModuleRTOOvertopping.f90.

Here is the call graph for this function:



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

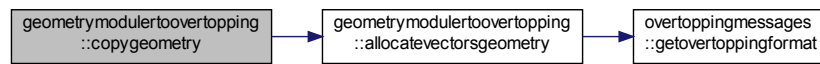
copyGeometry: copy a geometry structure

Parameters

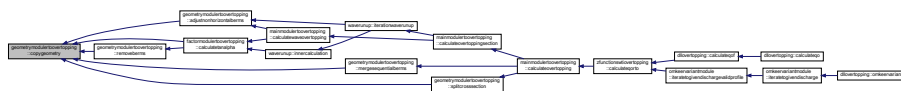
in	<i>geometry</i>	structure with geometry data
in, out	<i>geometrycopy</i>	structure with geometry data copy
out	<i>succes</i>	succes flag
in, out	<i>errormessage</i>	error message, only set in case of error

Definition at line 369 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.9 subroutine, public geometrymodulertovertopping::deallocateggeometry (type (tpgeometry), intent(inout) *geometry*)

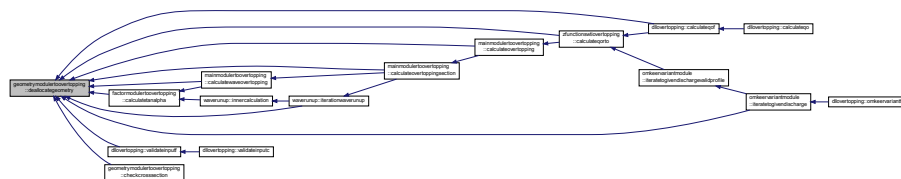
```
deallocateGeometry: deallocate the geometry vectors
```

Parameters

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

Definition at line 264 of file geometryModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.4.2.10 subroutine, public geometrymodule::determinesegmenttypes (type(tpgeometry), intent(inout)
geometry)

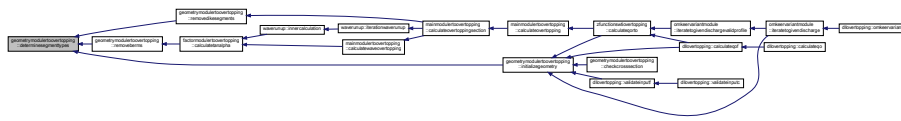
determineSegmentTypes: determine the segment types

Parameters

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

Definition at line 326 of file geometryModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.4.2.11 subroutine, public geometryModuleToOvertopping::initializeGeometry (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*, type (tpgeometry), intent(out) *geometry*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

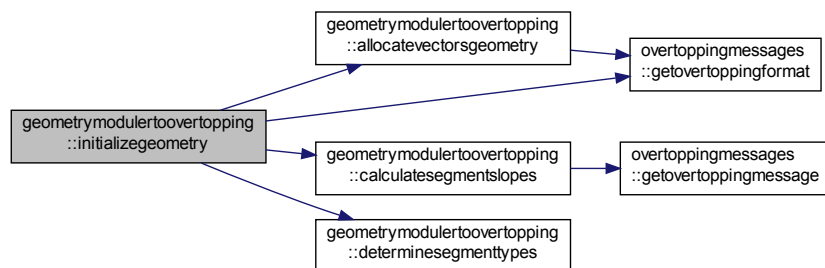
initializeGeometry: initialize the geometry

Parameters

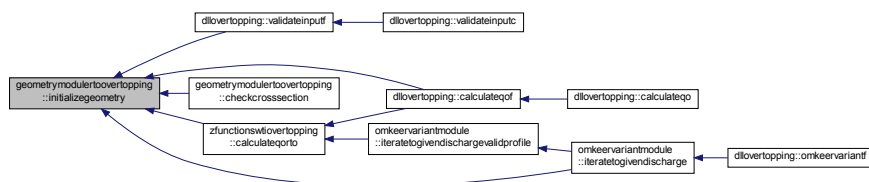
in	<i>psi</i>	dike normal (degree)
in	<i>ncoordinates</i>	number of coordinates
in	<i>xcoordinates</i>	x-coordinates (m)
in	<i>ycoordinates</i>	y-coordinates (m+NAP)
in	<i>roughnessfactors</i>	roughness factors
out	<i>geometry</i>	structure with geometry data
out	<i>succes</i>	flag for succes
out	<i>errorMessage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

Definition at line 149 of file geometryModuleRTOOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

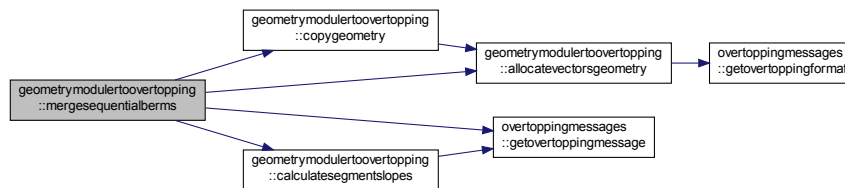
mergeSequentialBerms: merge sequential berms

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 420 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

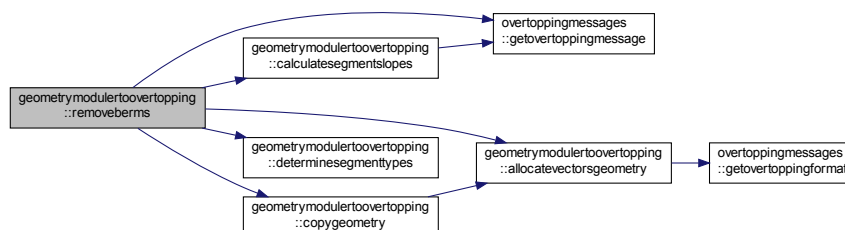
removeBerms: remove berms

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 618 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.14 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*)

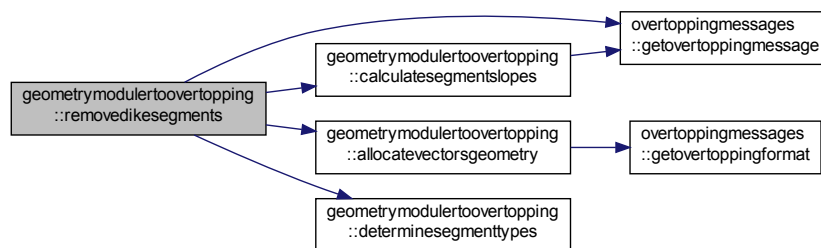
removeDikeSegments: remove dike segments

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 718 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.15 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*)

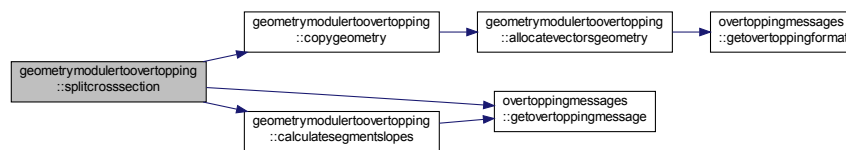
splitCrossSection: split a cross section

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 782 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5 mainmoduletooverlapping Module Reference

core computations for Dikes Overtopping

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

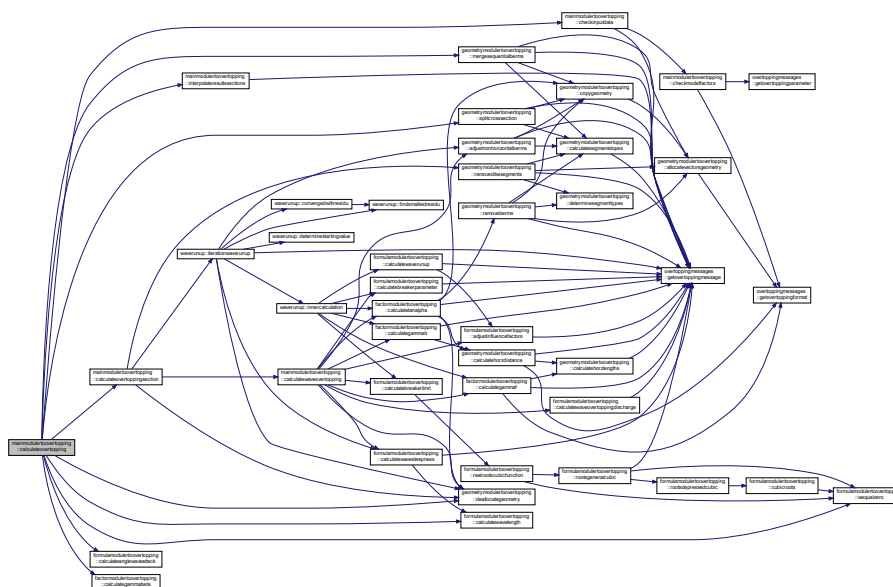
core computations for Dikes Overtopping

4.5.2.1 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*)

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

Here is the call graph for this function:



```

graph LR
    A["mainmodule::topoverlapping::calculateoverlapping"] --> B["zfunctions::wtoverlapping::calculateqto"]
    B --> C["dloverlapping::calculateqof"]
    C --> D["dloverlapping::calculateqo"]
    D --> E["omkevariantmodule::iterateqtoivendischarge"]
    E --> F["dloverlapping::omkevariant"]
  
```

4.5.2.2 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 481 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



4.5.2.3 subroutine, public mainmodulertoovertopping::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

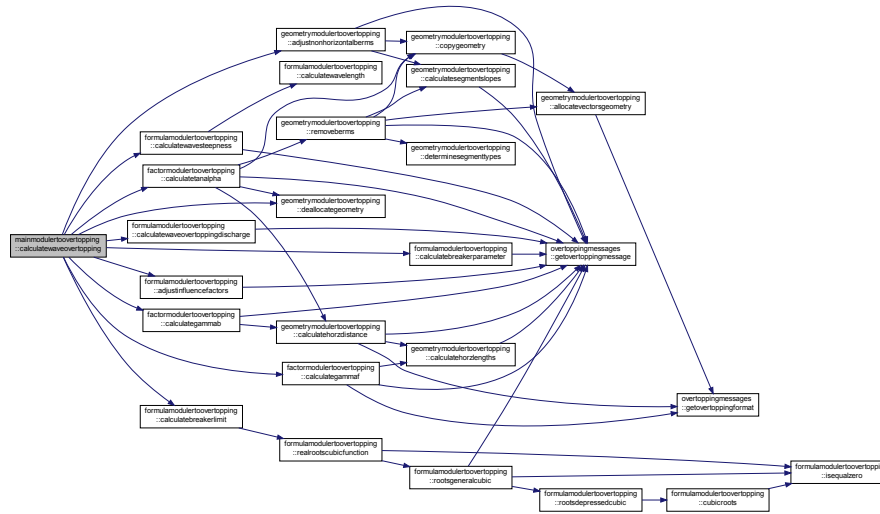
Definition at line 168 of file mainModuleRTOovertopping.f90.

calculateWaveOvertopping: calculate wave overtopping

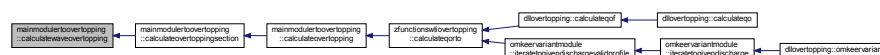
in	<i>geometry</i>	structure with geometry data
in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>tm_10</i>	spectral wave period (s)
in	<i>z2</i>	2% wave run-up (m)
in, out	<i>gammabeta_o</i>	influence angle wave attack overtopping
in	<i>modelfactors</i>	structure with model factors
out	<i>qo</i>	wave overtopping discharge (m ³ /m per s)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Generated on Wed May 4 2016 12:34:15 for Dikes Overtopping Kernel - Technical documentation by Doxygen

Here is the call graph for this function:



Here is the caller graph for this function:



```
4.5.2.5 subroutine, public mainmodulertovertopping::checkinputdata ( type (tpgeometry), intent(in) geometry, type (tpload),
intent(in) load, type (tpvertoppinginput), intent(in) modelFactors, logical, intent(out) succes, character(len=*),
intent(out) errorMessage )
```

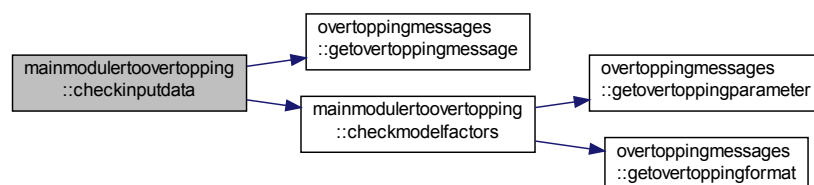
checkInputdata: check the input data

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

Definition at line 598 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.6 subroutine, public mainmoduletovertopping::checkmodelfactors (type (tpovertoppinginput), intent(in) *modelFactors*, integer, intent(in) *dimErrMessage*, character(len=*), dimension(dimerrmessage), intent(out) *errorMessages*, integer, intent(out) *ierr*)

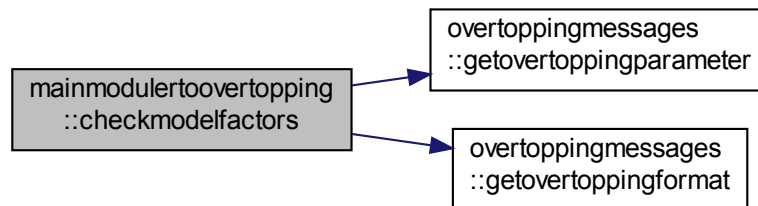
checkModelFactors: check the input data

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 657 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.7 subroutine, public mainmoduletovertopping::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) *succes*, character(len=*), intent(out) *errorMessage*)

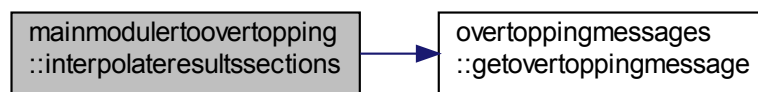
interpolateResultsSections: interpolate results for split cross sections

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

Definition at line 517 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

steering the extra logging

Data Types

- type `tlogging`

TLogging: structure for steering the logging.

Variables

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

4.6.1 Detailed Description

steering the extra logging

4.6.2 Variable Documentation

4.6.2.1 `type(tlogging) modulelogging::currentlogging`

copy of argument logging

Definition at line 23 of file ModuleLogging.f90.

4.6.2.2 `integer, parameter modulelogging::maxfilenamelen = 256`

maximum length of filename

Definition at line 15 of file ModuleLogging.f90.

4.7 omkeervariantmodule Module Reference

Module for the 'omkeerVariant'.

Functions/Subroutines

- subroutine, public [iteratetogivendischarge](#) (load, geometryF, givenDischarge, dikeHeight, modelFactors, overtopping, success, errorText, logging)
Subroutine with omkeerVariant.
- subroutine [iteratetogivendischargevalidprofile](#) (load, geometry, givenDischarge, dikeHeight, modelFactors, overtopping, success, errorText)
Subroutine with iterateToGivenDischarge, with already checked profile.

4.7.1 Detailed Description

Module for the 'omkeerVariant'.

4.7.2 Function/Subroutine Documentation

- 4.7.2.1 subroutine, public omkeervariantmodule::iteratetogivendischarge (type(tpload), intent(in) *load*, type(overtoppinggeometrytypef), intent(in) *geometryF*, real(kind=wp), intent(in) *givenDischarge*, real(kind=wp), intent(out) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, type(tpovertopping), intent(inout) *overtopping*, logical, intent(out) *success*, character(len=*), intent(out) *errorText*, type(tlogging), intent(in) *logging*)

Subroutine with omkeerVariant.

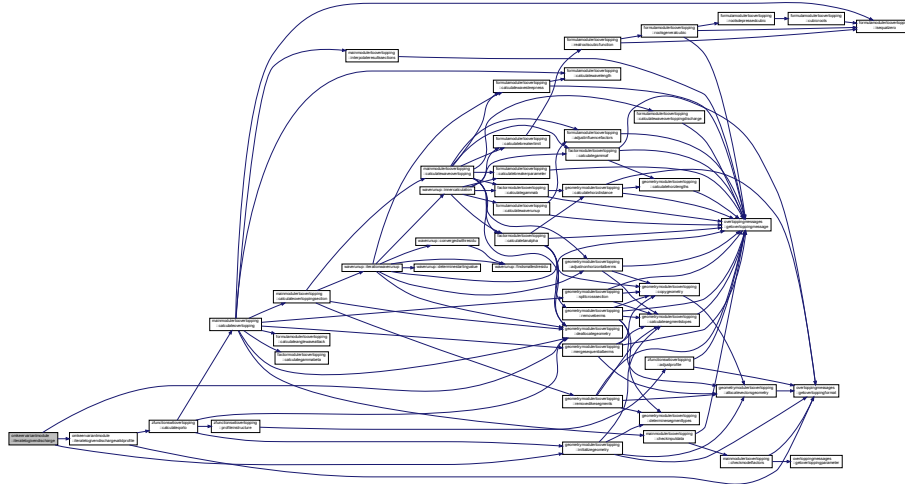
Parameters

in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>givendischarge</i>	discharge to iterate to
out	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with modelFactors
in, out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success

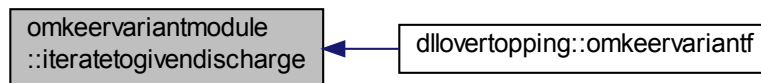
out	<i>errortext</i>	error message (only set if not successful)
in	<i>logging</i>	logging struct

Definition at line 25 of file omkeerVariantModule.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.2 subroutine omkeervariantmodule::iteratetogivendischargevalidprofile (type(tpload), intent(in) *load*, type (tpgeometry), intent(in) *geometry*, real(kind=wp), intent(in) *givenDischarge*, real(kind=wp), intent(out) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, type (tpovertopping), intent(inout) *overtopping*, logical, intent(out) *success*, character(len=*), intent(out) *errorText*)

Subroutine with iterateToGivenDischarge, with already checked profile.

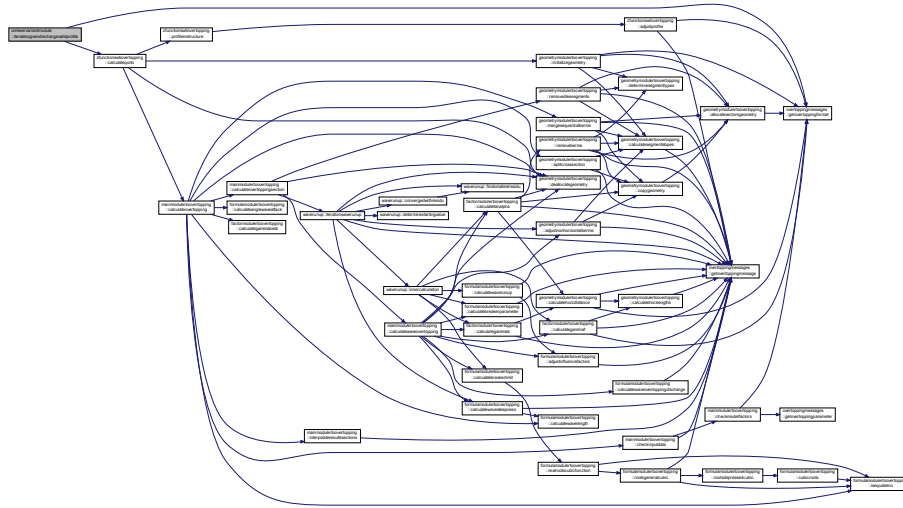
Parameters

in	<i>geometry</i>	internal structure with geometry data
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>givedischarge</i>	discharge to iterate to
out	<i>dikeheight</i>	dike height
in,out	<i>modelfactors</i>	struct with modelFactors
in,out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success

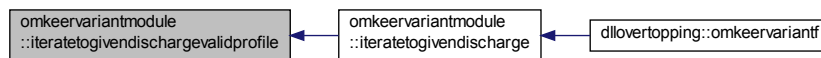
out	errortext	error message (only set if not successful)
-----	-----------	--

Definition at line 71 of file omkeerVariantModule.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.8 overtoppinginterface Module Reference

Module for the interface of dllOvertopping.

Data Types

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

Variables

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

4.8.1 Detailed Description

Module for the interface of dllOvertopping.

4.8.2 Variable Documentation

4.8.2.1 integer, parameter, public overtoppinginterface::varmodelfactorcriticalovertopping = 8

Model factor critical overtopping.

Definition at line 17 of file overtoppingInterface.f90.

4.9 overtoppingmessages Module Reference

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

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
- integer, parameter, private [maxpar](#) = 32
- character(len=2), private [language](#) = 'NL'
default : Dutch

4.9.1 Detailed Description

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

4.9.2 Function/Subroutine Documentation

4.9.2.1 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.

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

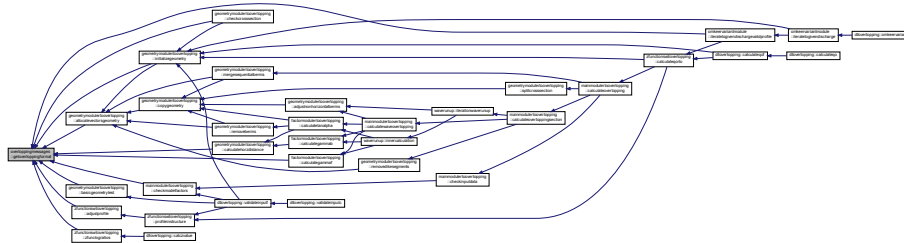
Subroutine that returns a Fortran format string with the corresponding ID in the current language.

Parameters

<i>in</i>	<i>id</i>	identification number of string
-----------	-----------	---------------------------------

Definition at line 265 of file OvertoppingMessages.f90.

Here is the caller graph for this function:



4.9.2.3 character(len=maxmsg) function overtoppingmessages::getovertoppingmessage (integer, intent(in) *ID*)

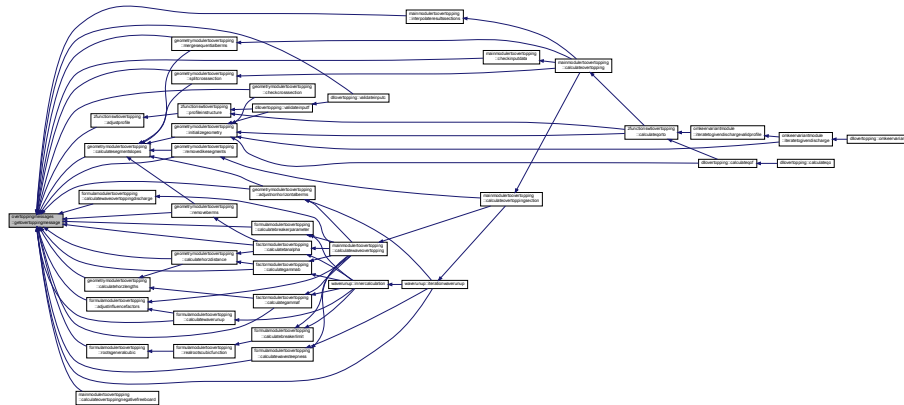
Subroutine that returns a message with the corresponding ID in the current language.

Parameters

<i>in</i>	<i>id</i>	identification number of string
-----------	-----------	---------------------------------

Definition at line 111 of file OvertoppingMessages.f90.

Here is the caller graph for this function:



4.9.2.4 character(len=maxpar) function overtoppingmessages::getovertoppingparameter (integer, intent(in) *ID*)

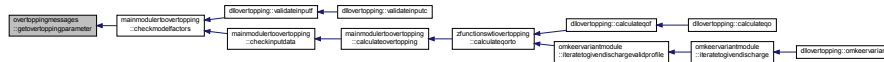
Subroutine that returns the name of an input parameter with the corresponding ID in the current language.

Parameters

<i>in</i>	<i>id</i>	identification number of string
-----------	-----------	---------------------------------

Definition at line 323 of file OvertoppingMessages.f90.

Here is the caller graph for this function:



4.9.2.5 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

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

Definition at line 83 of file OvertoppingMessages.f90.

4.9.3 Variable Documentation

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

default : Dutch

Definition at line 17 of file OvertoppingMessages.f90.

4.9.3.2 integer, parameter, private overtoppingmessages::maxmsg = 128

Definition at line 15 of file OvertoppingMessages.f90.

4.9.3.3 integer, parameter, private overtoppingmessages::maxpar =32

Definition at line 15 of file OvertoppingMessages.f90.

4.10 typedefinitionsrtooverlapping Module Reference

type definitions for Dikes Overtopping

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(kind=wp), parameter `frunup1` = 1.65_wp
- real(kind=wp), parameter `frunup2` = 4.00_wp
- real(kind=wp), parameter `frunup3` = 1.50_wp
- 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 `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

4.10.1 Detailed Description

type definitions for Dikes Overtopping

4.10.2 Variable Documentation

4.10.2.1 `real(wp), parameter typedefinitionsrtooverlapping::berm_max = 1.0d0/15`

maximal value gradient berm segment

Definition at line 70 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.2 `real(wp), parameter typedefinitionsrtooverlapping::berm_min = 0.0d0`

minimal value gradient berm segment

Definition at line 69 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.3 `real(wp), parameter typedefinitionsrtooverlapping::fb_max = huge(fb_max)`

maximal value model factor for breaking waves

Definition at line 79 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.4 `real(wp), parameter typedefinitionsrtooverlapping::fb_min = 0.0d0`

minimal value model factor for breaking waves

Definition at line 78 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.5 `real(wp), parameter typedefinitionsrtooverlapping::fn_max = huge(fn_max)`

maximal value model factor for non-breaking waves

Definition at line 81 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.6 `real(wp), parameter typedefinitionsrtooverlapping::fn_min = 0.0d0`

minimal value model factor for non-breaking waves

Definition at line 80 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.7 `real(wp), parameter typedefinitionsrtooverlapping::foreshore_max = 1.0d0`

maximal value reduction factor foreshore

Definition at line 85 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.8 `real(wp), parameter typedefinitionsrtooverlapping::foreshore_min = 0.3d0`

minimal value reduction factor foreshore

Definition at line 84 of file typeDefinitionsRTOoverlapping.f90.

4.10.2.9 `real(kind=wp), parameter typedefinitionsrtoovertopping::frunup1 = 1.65_wp`

Definition at line 42 of file typeDefinitionsRTOovertopping.f90.

4.10.2.10 `real(kind=wp), parameter typedefinitionsrtoovertopping::frunup2 = 4.00_wp`

Definition at line 43 of file typeDefinitionsRTOovertopping.f90.

4.10.2.11 `real(kind=wp), parameter typedefinitionsrtoovertopping::frunup3 = 1.50_wp`

Definition at line 44 of file typeDefinitionsRTOovertopping.f90.

4.10.2.12 `real(wp), parameter typedefinitionsrtoovertopping::fs_max = huge(fs_max)`

maximal value model factor for shallow waves

Definition at line 83 of file typeDefinitionsRTOovertopping.f90.

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

minimal value model factor for shallow waves

Definition at line 82 of file typeDefinitionsRTOovertopping.f90.

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

margin for minimal distance (m)

Definition at line 68 of file typeDefinitionsRTOovertopping.f90.

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

margin for minimal and maximal gradients

Definition at line 73 of file typeDefinitionsRTOovertopping.f90.

4.10.2.16 `real(wp), parameter typedefinitionsrtoovertopping::mz2_max = huge(mz2_max)`

maximal value model factor of 2% runup height

Definition at line 77 of file typeDefinitionsRTOovertopping.f90.

4.10.2.17 `real(wp), parameter typedefinitionsrtoovertopping::mz2_min = 0.0d0`

minimal value model factor of 2% runup height

Definition at line 76 of file typeDefinitionsRTOovertopping.f90.

4.10.2.18 `real(wp), parameter typedefinitionsrtoovertopping::rfactor_max = 1.0d0`

maximal value roughness factor dike segments

Definition at line 75 of file typeDefinitionsRTOovertopping.f90.

4.10.2.19 `real(wp), parameter typeDefinitionsRTOovertopping::rfactor_min = 0.5d0`

minimal value roughness factor dike segments

Definition at line 74 of file typeDefinitionsRTOovertopping.f90.

4.10.2.20 `real(wp), parameter typeDefinitionsRTOovertopping::slope_max = 1.0d0`

maximal value gradient slope segment

Definition at line 72 of file typeDefinitionsRTOovertopping.f90.

4.10.2.21 `real(wp), parameter typeDefinitionsRTOovertopping::slope_min = 1.0d0/8`

minimal value gradient slope segment

Definition at line 71 of file typeDefinitionsRTOovertopping.f90.

4.10.2.22 `real(wp), parameter typeDefinitionsRTOovertopping::xdiff_min = 2.0d-2`

minimal value distance between x-coordinates (m)

Definition at line 67 of file typeDefinitionsRTOovertopping.f90.

4.10.2.23 `integer, parameter typeDefinitionsRTOovertopping::z2_iter_max1 = 49`

maximal number of iterations for calculation z2 part 1

Definition at line 86 of file typeDefinitionsRTOovertopping.f90.

4.10.2.24 `integer, parameter typeDefinitionsRTOovertopping::z2_iter_max2 = 70`

maximal number of iterations for calculation z2 part 1 & 2

Definition at line 87 of file typeDefinitionsRTOovertopping.f90.

4.10.2.25 `real(wp), parameter typeDefinitionsRTOovertopping::z2_margin = 0.001d0`

margin for convergence criterium calculation z2

Definition at line 88 of file typeDefinitionsRTOovertopping.f90.

4.11 waverunup Module Reference

Iteration procedure for 2% wave runup.

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)

4.11.1 Detailed Description

Iteration procedure for 2% wave runup.

4.11.2 Function/Subroutine Documentation

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

Definition at line 319 of file `waveRunup.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.2 `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 268 of file `waveRunup.f90`.

Here is the caller graph for this function:



4.11.2.3 integer function `waverunup::findsmallestresidu` (`real(kind=wp)`, `dimension(:)`, `intent(in)` `z2_start`, `real(kind=wp)`, `dimension(:)`, `intent(in)` `z2_end`, `integer`, `intent(in)`, `optional` `n`) [private]

Definition at line 290 of file `waveRunup.f90`.

Here is the caller graph for this function:



- `real(kind=wp)` function, public [zfunclogratios](#) (`qo`, `qc`, `mgo`, `mqc`, `success`, `errorMessage`)

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

4.12.1 Detailed Description

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

4.12.2 Function/Subroutine Documentation

4.12.2.1 subroutine `zfunctionswtiovertopping::adjustprofile` (`integer`, `intent(in)` `nrCoordinates`, `type(tpprofilecoordinate)`, `dimension(nrcoordinates)`, `intent(in)` `coordinates`, `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`) [private]

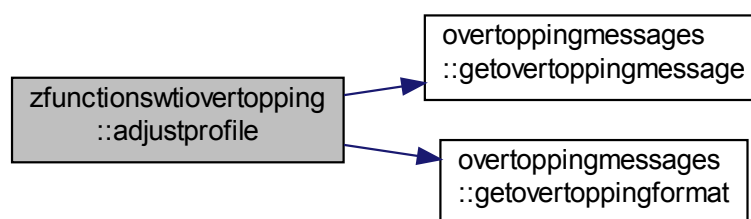
Subroutine adjust the profile due to a desired dike height.

Parameters

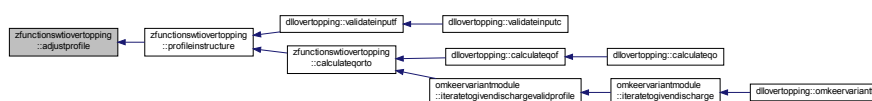
in	<code>nrcoordinates</code>	number of coordinates of the profile
in	<code>coordinates</code>	structure for the profile
in	<code>dikeheight</code>	dike height
out	<code>nrcoordsadjusted</code>	number of coordinates in the adjusted profile
	<code>xcoordsadjusted</code>	vector with x-coordinates of the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

Definition at line 108 of file `zFunctionsWTIOvertopping.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.2.2 subroutine, public zfunctionswtiovertopping::calculateqorto (real(kind=wp), intent(in) *dikeHeight*,
type(tpovertoppinginput), intent(inout) *modelFactors*, type (tpovertopping), intent(out) *overtopping*, type (tpload),
intent(in) *load*, type (tpgeometry), intent(in) *geometry*, logical, intent(out) *succes*, character(len=*), intent(out)
errorMessage)

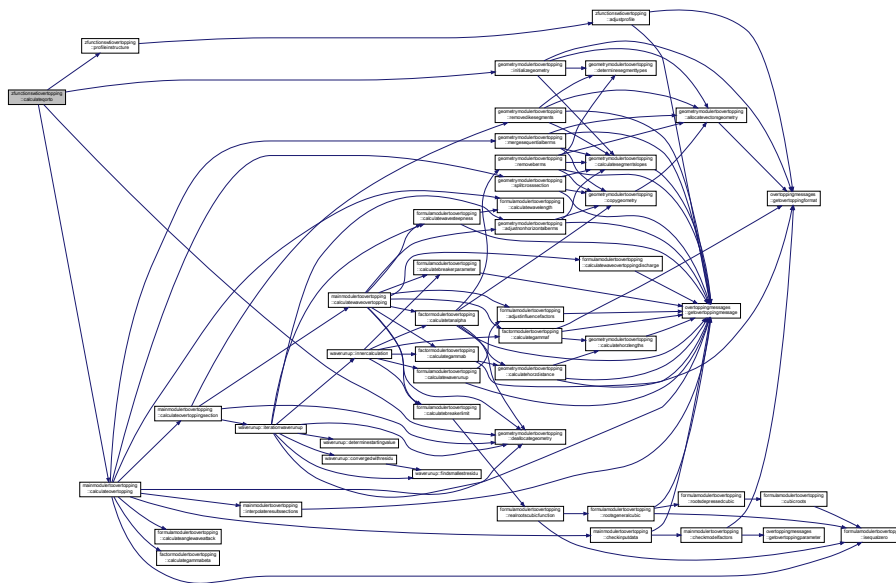
Subroutine to calculate the overtopping discharge with the RTO-overtopping dll.

Parameters

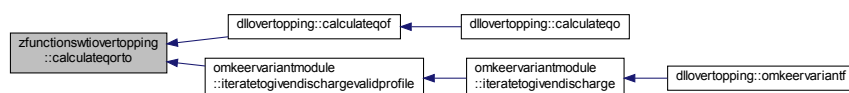
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with model factors
out	<i>overtopping</i>	structure with overtopping results
in	<i>geometry</i>	structure with geometry data
in	<i>load</i>	structure with load parameters
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error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

Definition at line 34 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.2.3 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

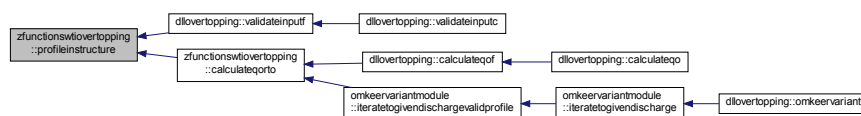
in	<i>nrcoordinates</i>	number of coordinates of the profile
in	<i>xcoordinates</i>	vector with x-coordinates of the profile
in	<i>ycoordinates</i>	vector with y-coordinates of the profile
in	<i>dikeheight</i>	dike height
out	<i>nrcoordsadjusted</i>	number of coordinates in the adjusted profile
	<i>xcoordsadjusted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

Definition at line 83 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.2.4 `real (kind=wp) function, public zfunctionswtiovertopping::zfunclogratios (real (kind=wp), intent(in) qo, real (kind=wp), intent(in) qc, real (kind=wp), intent(in) mqo, 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

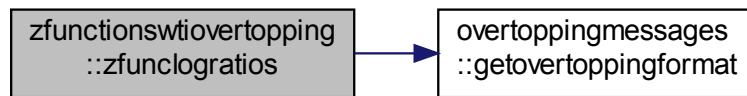
in	<i>qo</i>	computed overtopping discharge
in	<i>qc</i>	Critical overtopping discharge
in	<i>mqo</i>	Model factor computed overtopping discharge
in	<i>mqc</i>	Model factor Critical overtopping discharge
out	<i>success</i>	Flag for succes
out	<i>errormessage</i>	error message, only set if not successful

Returns

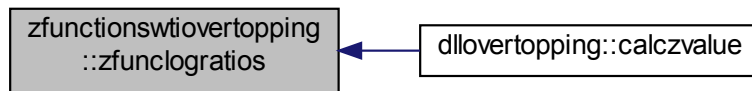
Value z-function

Definition at line 212 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

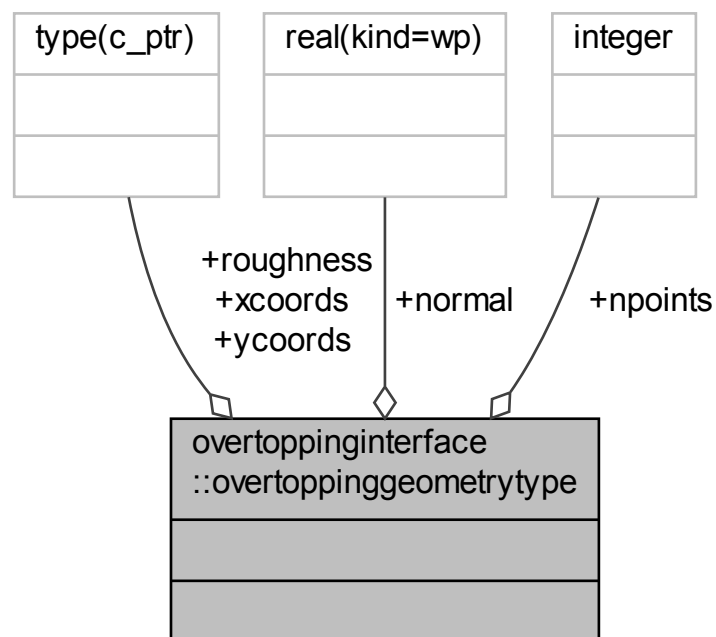


Chapter 5

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`

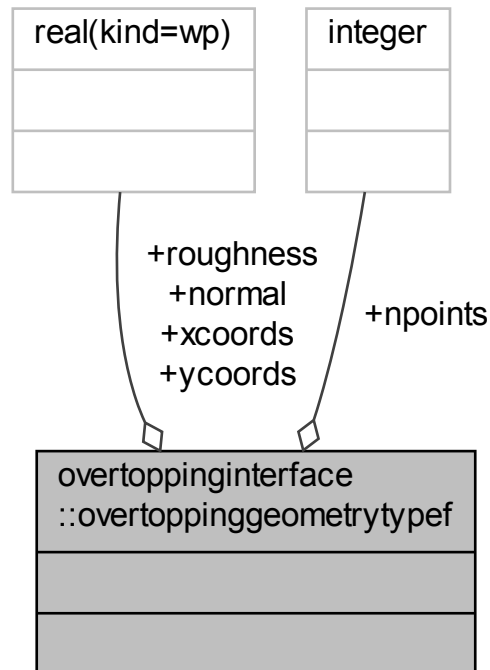
Definition at line 28 of file overtoppingInterface.f90.

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`.

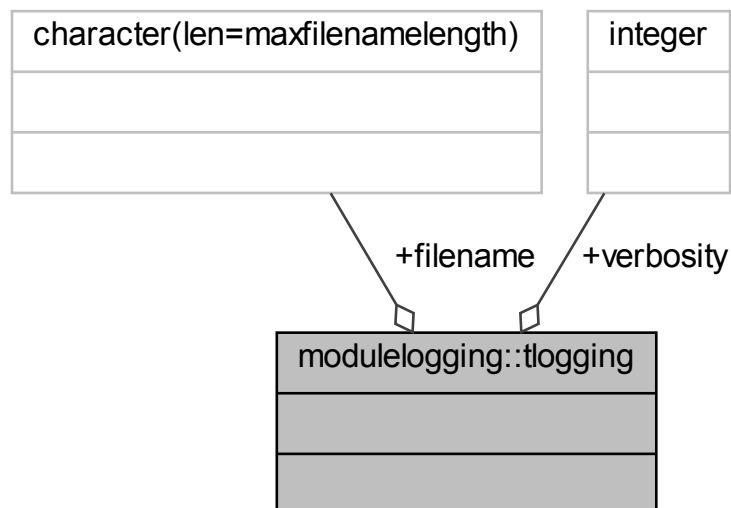
5.2.2.5 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::ycoords`

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 18 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 20 of file ModuleLogging.f90.

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

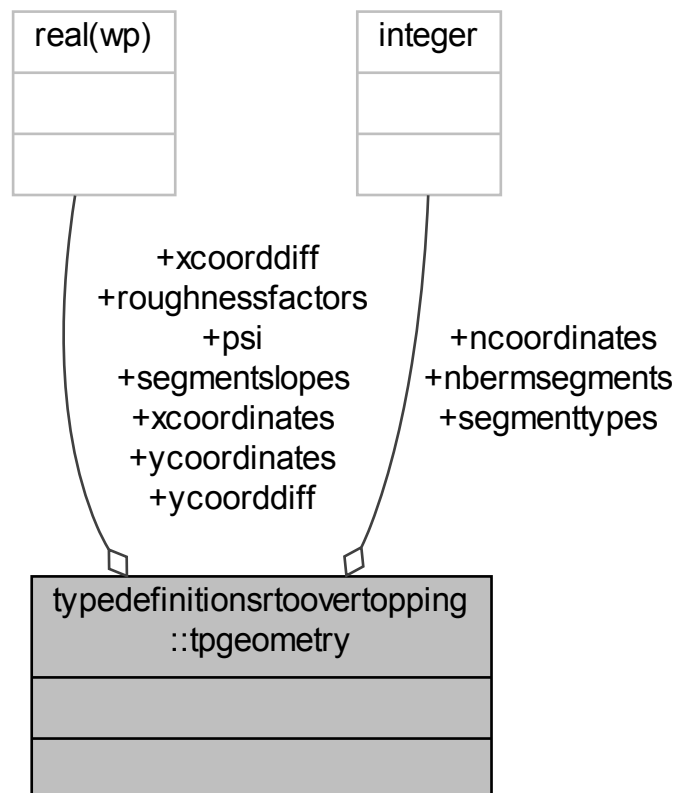
level of verbosity: one of verboseNone, verboseBasic, verboseDetailed, verboseDebugging

Definition at line 19 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 [nbermsegments](#)
number of berm segments

5.4.1 Detailed Description

tpGeometry: structure with geometry data

Definition at line 21 of file typeDefinitionsRTOovertopping.f90.

5.4.2 Member Data Documentation

5.4.2.1 integer typedefinitionsrtooverlapping::tpgeometry::nbermsegments

number of berm segments

Definition at line 31 of file typeDefinitionsRTOovertopping.f90.

5.4.2.2 integer typedefinitionsrtooverlapping::tpgeometry::ncoordinates

number of coordinates cross section

Definition at line 23 of file typeDefinitionsRTOovertopping.f90.

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

dike normal (degrees)

Definition at line 22 of file typeDefinitionsRTOovertopping.f90.

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

vector with roughness factors cross section

Definition at line 26 of file typeDefinitionsRTOovertopping.f90.

5.4.2.5 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::segmentslopes

vector with slopes dike segments

Definition at line 29 of file typeDefinitionsRTOoverlapping.f90.

5.4.2.6 integer, dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::segmenttypes

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

Definition at line 30 of file typeDefinitionsRTOoverlapping.f90.

5.4.2.7 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::xcoorddiff

vector with differences in x-coordinates (m)

Definition at line 27 of file typeDefinitionsRTOoverlapping.f90.

5.4.2.8 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::xcoordinates

vector with x-coordinates cross section (m)

Definition at line 24 of file typeDefinitionsRTOoverlapping.f90.

5.4.2.9 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::ycoorddiff

vector with differences in y-coordinates (m)

Definition at line 28 of file typeDefinitionsRTOoverlapping.f90.

5.4.2.10 real(wp), dimension(:), pointer typedefinitionsrtooverlapping::tpgeometry::ycoordinates

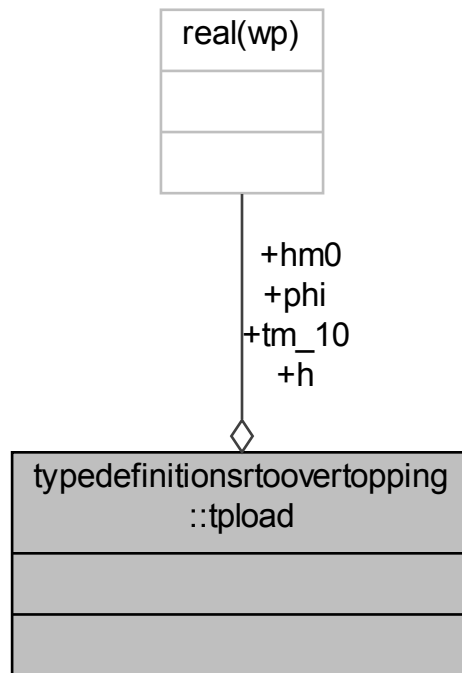
vector with y-coordinates cross section (m+NAP)

Definition at line 25 of file typeDefinitionsRTOoverlapping.f90.

5.5 typedefinitionsrtooverlapping::tpload Type Reference

tpLoad: structure with load parameters

Collaboration diagram for `typedefinitionsrtoover topping::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 35 of file `typeDefinitionsRTOover topping.f90`.

5.5.2 Member Data Documentation

5.5.2.1 `real(wp)` `typedefinitionsrtoover topping::tpload::h`

local water level (m+NAP)

Definition at line 36 of file `typeDefinitionsRTOover topping.f90`.

5.5.2.2 `real(wp) typedefinitionsrtoovertopping::pload::hm0`

significant wave height (m)

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

5.5.2.3 `real(wp) typedefinitionsrtoovertopping::pload::phi`

wave direction (degrees)

Definition at line 39 of file `typeDefinitionsRTOovertopping.f90`.

5.5.2.4 `real(wp) typedefinitionsrtoovertopping::pload::tm_10`

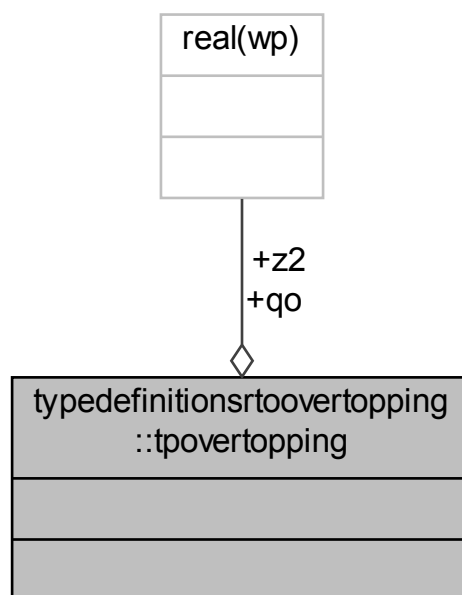
spectral wave period (s)

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

5.6 `typedefinitionsrtoovertopping::tpovertopping` Type Reference

`tpOvertopping`: structure with overtopping results

Collaboration diagram for `typedefinitionsrtoovertopping::tpovertopping`:



Public Attributes

- `real(wp)` `z2`
2% wave run-up (m)
- `real(wp)` `qo`

wave overtopping discharge (m³/m per s)

5.6.1 Detailed Description

tpOvertopping: structure with overtopping results

Definition at line 59 of file typeDefinitionsRTOovertopping.f90.

5.6.2 Member Data Documentation

5.6.2.1 `real(wp) typedefinitionsrtooveropping::tpovertopping::qo`

wave overtopping discharge (m³/m per s)

Definition at line 61 of file typeDefinitionsRTOovertopping.f90.

5.6.2.2 `real(wp) typedefinitionsrtooveropping::tpovertopping::z2`

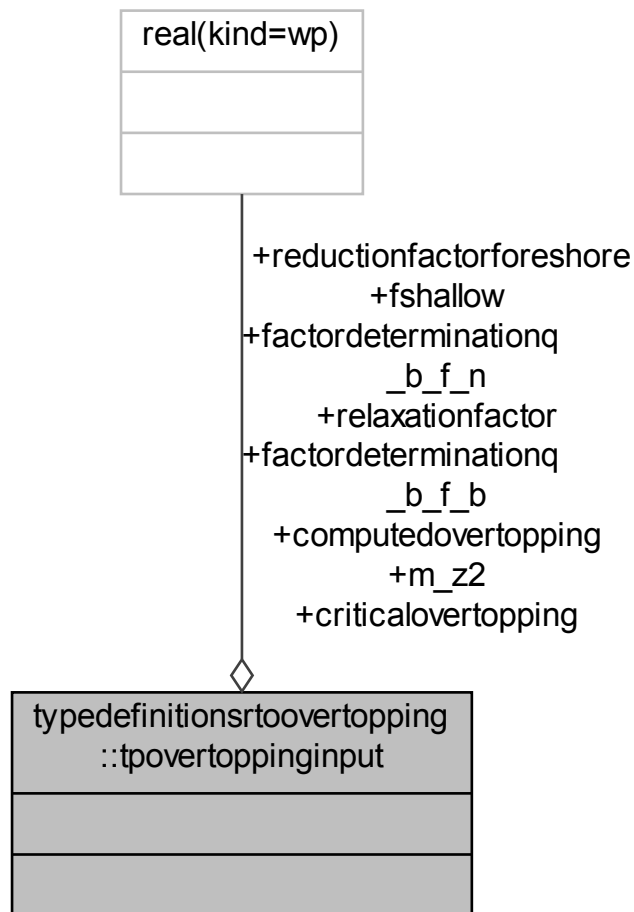
2% wave run-up (m)

Definition at line 60 of file typeDefinitionsRTOovertopping.f90.

5.7 `typedefinitionsrtooveropping::tpovertoppinginput` 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)` `fshallow`
model factor for shallow waves
- `real(kind=wp)` `computedovertopping`
model factor computed overtopping
- `real(kind=wp)` `criticalovertopping`
model factor critical overtopping
- `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 47 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 52 of file typeDefinitionsRTOovertopping.f90.

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

model factor critical overtopping

Definition at line 53 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 49 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 48 of file typeDefinitionsRTOovertopping.f90.

5.7.2.5 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::fshallow`

model factor for shallow waves

Definition at line 51 of file typeDefinitionsRTOovertopping.f90.

5.7.2.6 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::m_z2`

model factor describing the uncertainty of 2% runup height

Definition at line 50 of file typeDefinitionsRTOovertopping.f90.

5.7.2.7 `real(kind=wp) typedefinitionsrtooverlapping::tpovertoppinginput::reductionfactorforeshore = 0.5_wp`

reduction factor foreshore

Definition at line 55 of file typeDefinitionsRTOovertopping.f90.

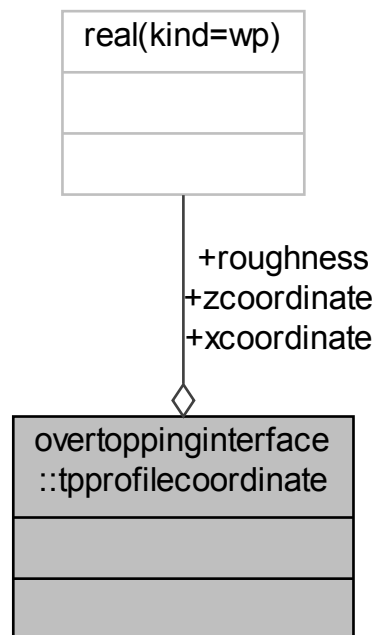
5.7.2.8 `real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::relaxationfactor`

relaxation factor iteration procedure wave runup

Definition at line 54 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](#)
Main entry for the dll DikesOvertopping.

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::omkeervariantf](#) (load, geometryF, givenDischarge, dikeHeight, model↔ Factors, overtopping, success, errorText, logging)
Subroutine with omkeerVariant.
- 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
- omkeerVariantF
- 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 [factormodulertooveropping](#)
functions for the slope angle and influence factors

Functions/Subroutines

- subroutine, public [factormodulertooveropping::calculatetanalpha](#) (h, Hm0, z2, geometry, tanAlpha, succes, errorMessage)
calculateTanAlpha representative slope angle
- subroutine, public [factormodulertooveropping::calculategammabeta](#) (Hm0, Tm_10, beta, gammaBeta_↔
z, gammaBeta_o)
calculateGammaBeta influence factor angle of wave attack
- subroutine, public [factormodulertooveropping::calculategammaf](#) (h, ksi0, ksi0Limit, gammaB, z2, geometry, gammaF, succes, errorMessage)
calculateGammaF influence factor roughness
- subroutine, public [factormodulertooveropping::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](#)
the core computations for Dikes Overtopping

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](#) (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](#)
core computations related to the geometry

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)
deallocategemetry: 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](#)
core computations for Dikes Overtopping

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, errorMessage, ierr)
checkModelFactors: check the input data

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](#)
steering the extra logging

Variables

- integer, parameter `modulelogging::maxfilenamelen` = 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 omkeerVariantModule.f90 File Reference

This file contains the omkeerVariant.

Modules

- module `omkeervariantmodule`
Module for the 'omkeerVariant'.

Functions/Subroutines

- subroutine, public `omkeervariantmodule::iteratetogivendischarge` (load, geometryF, givenDischarge, dike↔Height, modelFactors, overtopping, success, errorText, logging)
Subroutine with omkeerVariant.
- subroutine `omkeervariantmodule::iteratetogivendischargevalidprofile` (load, geometry, givenDischarge, dike↔Height, modelFactors, overtopping, success, errorText)
Subroutine with iterateToGivenDischarge, with already checked profile.

6.7.1 Detailed Description

This file contains the omkeerVariant.

6.8 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`
Module for the interface of dllOvertopping.

Variables

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

6.8.1 Detailed Description

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

6.9 OvertoppingMessages.f90 File Reference

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

Modules

- module `overtoppingmessages`
Module for the messages in the overtopping dll, in Dutch or English.

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
- integer, parameter, private `overtoppingmessages::maxpar` = 32
- character(len=2), private `overtoppingmessages::language` = 'NL'
default : Dutch

6.9.1 Detailed Description

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

6.10 typeDefinitionsRTOovertopping.f90 File Reference

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

Data Types

- type [typedefinitionsrtooverlapping::tpgeometry](#)
tpGeometry: structure with geometry data
- type [typedefinitionsrtooverlapping::tpload](#)
tpLoad: structure with load parameters
- type [typedefinitionsrtooverlapping::tpoverlappinginput](#)
OverlappingModelFactors: C-structure with model factors.
- type [typedefinitionsrtooverlapping::tpoverlapping](#)
tpOverlapping: structure with overtopping results

Modules

- module [typedefinitionsrtooverlapping](#)
type definitions for Dikes Overtopping

Variables

- real(kind=wp), parameter [typedefinitionsrtooverlapping::frunup1](#) = 1.65_wp
- real(kind=wp), parameter [typedefinitionsrtooverlapping::frunup2](#) = 4.00_wp
- real(kind=wp), parameter [typedefinitionsrtooverlapping::frunup3](#) = 1.50_wp
- real(wp), parameter [typedefinitionsrtooverlapping::xdiff_min](#) = 2.0d-2
minimal value distance between x-coordinates (m)
- real(wp), parameter [typedefinitionsrtooverlapping::margindiff](#) = 1.0d-14
margin for minimal distance (m)
- real(wp), parameter [typedefinitionsrtooverlapping::berm_min](#) = 0.0d0
minimal value gradient berm segment
- real(wp), parameter [typedefinitionsrtooverlapping::berm_max](#) = 1.0d0/15
maximal value gradient berm segment
- real(wp), parameter [typedefinitionsrtooverlapping::slope_min](#) = 1.0d0/8
minimal value gradient slope segment
- real(wp), parameter [typedefinitionsrtooverlapping::slope_max](#) = 1.0d0
maximal value gradient slope segment
- real(wp), parameter [typedefinitionsrtooverlapping::maringrad](#) = 0.0025d0
margin for minimal and maximal gradients
- real(wp), parameter [typedefinitionsrtooverlapping::rfactor_min](#) = 0.5d0
minimal value roughness factor dike segments
- real(wp), parameter [typedefinitionsrtooverlapping::rfactor_max](#) = 1.0d0
maximal value roughness factor dike segments
- real(wp), parameter [typedefinitionsrtooverlapping::mz2_min](#) = 0.0d0
minimal value model factor of 2% runup height
- real(wp), parameter [typedefinitionsrtooverlapping::mz2_max](#) = huge(mz2_max)
maximal value model factor of 2% runup height
- real(wp), parameter [typedefinitionsrtooverlapping::fb_min](#) = 0.0d0
minimal value model factor for breaking waves
- real(wp), parameter [typedefinitionsrtooverlapping::fb_max](#) = huge(fb_max)
maximal value model factor for breaking waves
- real(wp), parameter [typedefinitionsrtooverlapping::fn_min](#) = 0.0d0
minimal value model factor for non-breaking waves
- real(wp), parameter [typedefinitionsrtooverlapping::fn_max](#) = huge(fn_max)
maximal value model factor for non-breaking waves

- real(wp), parameter [typedefinitionsrtooverlapping::fs_min](#) = 0.0d0
minimal value model factor for shallow waves
- real(wp), parameter [typedefinitionsrtooverlapping::fs_max](#) = huge(fs_max)
maximal value model factor for shallow waves
- real(wp), parameter [typedefinitionsrtooverlapping::foreshore_min](#) = 0.3d0
minimal value reduction factor foreshore
- real(wp), parameter [typedefinitionsrtooverlapping::foreshore_max](#) = 1.0d0
maximal value reduction factor foreshore
- integer, parameter [typedefinitionsrtooverlapping::z2_iter_max1](#) = 49
maximal number of iterations for calculation z2 part 1
- integer, parameter [typedefinitionsrtooverlapping::z2_iter_max2](#) = 70
maximal number of iterations for calculation z2 part 1 & 2
- real(wp), parameter [typedefinitionsrtooverlapping::z2_margin](#) = 0.001d0
margin for convergence criterium calculation z2

6.10.1 Detailed Description

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

6.11 waveRunup.f90 File Reference

This file contains a module with the iteration procedure for 2% wave runup.

Modules

- module [waverunup](#)
Iteration procedure for 2% wave runup.

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)

6.11.1 Detailed Description

This file contains a module with the iteration procedure for 2% wave runup.

6.12 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)

6.12.1 Detailed Description

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

Index

- adjustinfluencefactors
 - formulamodulertoovertopping, [19](#)
- adjustnonhorizontalberms
 - geometrymodulertoovertopping, [29](#)
- adjustprofile
 - zfunctionswtioovertopping, [61](#)
- allocatevectorsgeometry
 - geometrymodulertoovertopping, [29](#)
- basicgeometrytest
 - geometrymodulertoovertopping, [31](#)
- berm_max
 - typedefinitionsrtoovertopping, [55](#)
- berm_min
 - typedefinitionsrtoovertopping, [55](#)
- calculateanglewaveattack
 - formulamodulertoovertopping, [20](#)
- calculatebreakerlimit
 - formulamodulertoovertopping, [20](#)
- calculatebreakerparameter
 - formulamodulertoovertopping, [20](#)
- calculategammab
 - factormodulertoovertopping, [14](#)
- calculategammabeta
 - factormodulertoovertopping, [14](#)
- calculategammalf
 - factormodulertoovertopping, [16](#)
- calculatehorzdistance
 - geometrymodulertoovertopping, [32](#)
- calculatehorzlengths
 - geometrymodulertoovertopping, [32](#)
- calculateovertopping
 - mainmodulertoovertopping, [41](#)
- calculateovertoppingnegativefreeboard
 - mainmodulertoovertopping, [41](#)
- calculateovertoppingsection
 - mainmodulertoovertopping, [43](#)
- calculateqo
 - dllovertopping, [8](#)
- calculateqof
 - dllovertopping, [9](#)
- calculateqorto
 - zfunctionswtioovertopping, [61](#)
- calculatesegmentslopes
 - geometrymodulertoovertopping, [33](#)
- calculatetanalpha
 - factormodulertoovertopping, [16](#)
- calculatewavelength
 - formulamodulertoovertopping, [21](#)
- calculatewaveovertopping
 - mainmodulertoovertopping, [44](#)
- calculatewaveovertoppingdischarge
 - formulamodulertoovertopping, [21](#)
- calculatewaverunup
 - formulamodulertoovertopping, [22](#)
- calculatewavesteepness
 - formulamodulertoovertopping, [23](#)
- calczvalue
 - dllovertopping, [10](#)
- checkcrosssection
 - geometrymodulertoovertopping, [34](#)
- checkinputdata
 - mainmodulertoovertopping, [45](#)
- checkmodelfactors
 - mainmodulertoovertopping, [46](#)
- computedovertopping
 - typedefinitionsrtoovertopping::tpovertoppinginput, [78](#)
- convergedwithresidu
 - waverunup, [58](#)
- copygeometry
 - geometrymodulertoovertopping, [34](#)
- criticalovertopping
 - typedefinitionsrtoovertopping::tpovertoppinginput, [78](#)
- cubicroots
 - formulamodulertoovertopping, [23](#)
- currentlogging
 - modulelogging, [48](#)
- deallocategeometry
 - geometrymodulertoovertopping, [35](#)
- determinesegmenttypes
 - geometrymodulertoovertopping, [35](#)
- determinestartingvalue
 - waverunup, [58](#)
- dllovertopping.f90, [81](#)
- dllovertopping, [7](#)
 - calculateqo, [8](#)
 - calculateqof, [9](#)
 - calczvalue, [10](#)
 - geometry_c_f, [11](#)
 - getlanguage, [11](#)
 - omkeervariantf, [11](#)
 - setlanguage, [12](#)
 - validateinputc, [12](#)
 - validateinputf, [12](#)
 - versionnumber, [13](#)

- factorModuleRTOovertopping.f90, 82
- factordeterminationq_b_f_b
 - typedefinitionsrtooveropping::tpovertoppinginput, 78
- factordeterminationq_b_f_n
 - typedefinitionsrtooveropping::tpovertoppinginput, 78
- factormodulertooveropping, 13
 - calculategammab, 14
 - calculategammabeta, 14
 - calculategammaf, 16
 - calculatetanalpha, 16
- fb_max
 - typedefinitionsrtooveropping, 55
- fb_min
 - typedefinitionsrtooveropping, 55
- filename
 - modulelogging::tlogging, 71
- findsmallestresidu
 - waverunup, 58
- fn_max
 - typedefinitionsrtooveropping, 55
- fn_min
 - typedefinitionsrtooveropping, 55
- foreshore_max
 - typedefinitionsrtooveropping, 55
- foreshore_min
 - typedefinitionsrtooveropping, 55
- formulaModuleRTOovertopping.f90, 82
- formulamodulertooveropping, 18
 - adjustinfluencefactors, 19
 - calculateanglewaveattack, 20
 - calculatebreakerlimit, 20
 - calculatebreakerparameter, 20
 - calculatewavelength, 21
 - calculatewaveoveroppingdischarge, 21
 - calculatewaverunup, 22
 - calculatewavesteepness, 23
 - cubicroots, 23
 - isequalreal, 24
 - isequalzero, 24
 - realrootscubicfunction, 24
 - rootsdepressedcubic, 26
 - rootsgeneralcubic, 26
- frunup1
 - typedefinitionsrtooveropping, 55
- frunup2
 - typedefinitionsrtooveropping, 56
- frunup3
 - typedefinitionsrtooveropping, 56
- fs_max
 - typedefinitionsrtooveropping, 56
- fs_min
 - typedefinitionsrtooveropping, 56
- fshallow
 - typedefinitionsrtooveropping::tpovertoppinginput, 78
- geometry_c_f
 - dllovertopping, 11
- geometryModuleRTOovertopping.f90, 83
- geometrymodulertooveropping, 28
 - adjustnonhorizontalberms, 29
 - allocatevectorsgeometry, 29
 - basicgeometrytest, 31
 - calculatehorzdistance, 32
 - calculatehorzlengths, 32
 - calculatesegmentslopes, 33
 - checkcrosssection, 34
 - copygeometry, 34
 - deallocategeometry, 35
 - determinesegmenttypes, 35
 - initializegeometry, 36
 - mergesequentialberms, 36
 - removeberms, 38
 - removedikesegments, 39
 - splitcrosssection, 39
- getlanguage
 - dllovertopping, 11
 - overtoppingmessages, 51
- getovertoppingformat
 - overtoppingmessages, 51
- getovertoppingmessage
 - overtoppingmessages, 52
- getovertoppingparameter
 - overtoppingmessages, 52
- h
 - typedefinitionsrtooveropping::tpload, 74
- hm0
 - typedefinitionsrtooveropping::tpload, 74
- initializegeometry
 - geometrymodulertooveropping, 36
- innercalculation
 - waverunup, 58
- interpolateresultssections
 - mainmodulertooveropping, 46
- isequalreal
 - formulamodulertooveropping, 24
- isequalzero
 - formulamodulertooveropping, 24
- iteratetogivendischarge
 - omkeervariantmodule, 48
- iteratetogivendischargevalidprofile
 - omkeervariantmodule, 49
- iterationwaverunup
 - waverunup, 59
- language
 - overtoppingmessages, 53
- m_z2
 - typedefinitionsrtooveropping::tpovertoppinginput, 78
- mainModuleRTOovertopping.f90, 85
- mainmodulertooveropping, 40
 - calculateovertopping, 41

- calculateovertoppingnegativefreeboard, 41
 - calculateovertoppingsection, 43
 - calculatewaveovertopping, 44
 - checkinputdata, 45
 - checkmodelfactors, 46
 - interpolateresultssections, 46
- margindiff
 - typedefinitionsrtooverlapping, 56
- maringrad
 - typedefinitionsrtooverlapping, 56
- maxfilenamelength
 - modulelogging, 48
- maxmsg
 - overtoppingmessages, 53
- maxpar
 - overtoppingmessages, 53
- mergesequentialberms
 - geometrymodulertooverlapping, 36
- ModuleLogging.f90, 85
- modulelogging, 47
 - currentlogging, 48
 - maxfilenamelength, 48
- modulelogging::tlogging, 70
 - filename, 71
 - verbosity, 71
- mz2_max
 - typedefinitionsrtooverlapping, 56
- mz2_min
 - typedefinitionsrtooverlapping, 56
- nbermssegments
 - typedefinitionsrtooverlapping::tpgeometry, 72
- ncoordinates
 - typedefinitionsrtooverlapping::tpgeometry, 72
- normal
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
- npoints
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
- omkeerVariantModule.f90, 86
- omkeervariantf
 - dlllovertopping, 11
- omkeervariantmodule, 48
 - iteratetogivendischarge, 48
 - iteratetogivendischargevalidprofile, 49
- overtoppingInterface.f90, 86
- OvertoppingMessages.f90, 87
- overtoppinginterface, 50
 - varmodelfactorcriticalovertopping, 51
- overtoppinginterface::overtoppinggeometrytype, 67
 - normal, 68
 - npoints, 68
 - roughness, 68
 - xcoords, 68
 - ycoords, 68
- overtoppinginterface::overtoppinggeometrytypef, 69
 - normal, 69
 - npoints, 69
 - roughness, 69
 - xcoords, 70
 - ycoords, 70
- overtoppinginterface::tpprofilecoordinate, 79
 - roughness, 79
 - xcoordinate, 80
 - zcoordinate, 80
- overtoppingmessages, 51
 - getlanguage, 51
 - getovertoppingformat, 51
 - getovertoppingmessage, 52
 - getovertoppingparameter, 52
 - language, 53
 - maxmsg, 53
 - maxpar, 53
 - setlanguage, 53
- phi
 - typedefinitionsrtooverlapping::tpload, 75
- profileinstructure
 - zfunctionswtiooverlapping, 63
- psi
 - typedefinitionsrtooverlapping::tpgeometry, 72
- qo
 - typedefinitionsrtooverlapping::tpovertopping, 76
- realrootscubicfunction
 - formulamodulertooverlapping, 24
- reductionfactorforeshore
 - typedefinitionsrtooverlapping::tpovertoppinginput, 78
- relaxationfactor
 - typedefinitionsrtooverlapping::tpovertoppinginput, 78
- removeberms
 - geometrymodulertooverlapping, 38
- removedikessegments
 - geometrymodulertooverlapping, 39
- rfactor_max
 - typedefinitionsrtooverlapping, 56
- rfactor_min
 - typedefinitionsrtooverlapping, 56
- rootsdepressedcubic
 - formulamodulertooverlapping, 26
- rootsgeneralcubic
 - formulamodulertooverlapping, 26
- roughness
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
 - overtoppinginterface::tpprofilecoordinate, 79
- roughnessfactors

- typedefinitionsrtooverlapping::tpgeometry, 72
- segmentslopes
 - typedefinitionsrtooverlapping::tpgeometry, 72
- segmenttypes
 - typedefinitionsrtooverlapping::tpgeometry, 73
- setlanguage
 - dllovertopping, 12
 - overlappingmessages, 53
- slope_max
 - typedefinitionsrtooverlapping, 57
- slope_min
 - typedefinitionsrtooverlapping, 57
- splitcrosssection
 - geometrymodulertooverlapping, 39
- tm_10
 - typedefinitionsrtooverlapping::tpload, 75
- typeDefinitionsRTOoverlapping.f90, 87
- typedefinitionsrtooverlapping, 53
 - berm_max, 55
 - berm_min, 55
 - fb_max, 55
 - fb_min, 55
 - fn_max, 55
 - fn_min, 55
 - foreshore_max, 55
 - foreshore_min, 55
 - frunup1, 55
 - frunup2, 56
 - frunup3, 56
 - fs_max, 56
 - fs_min, 56
 - margindiff, 56
 - maringrad, 56
 - mz2_max, 56
 - mz2_min, 56
 - rfactor_max, 56
 - rfactor_min, 56
 - slope_max, 57
 - slope_min, 57
 - xdiff_min, 57
 - z2_iter_max1, 57
 - z2_iter_max2, 57
 - z2_margin, 57
- typedefinitionsrtooverlapping::tpgeometry, 71
 - nbermsegments, 72
 - ncoordinates, 72
 - psi, 72
 - roughnessfactors, 72
 - segmentslopes, 72
 - segmenttypes, 73
 - xcoorddiff, 73
 - xcoordinates, 73
 - ycoorddiff, 73
 - ycoordinates, 73
- typedefinitionsrtooverlapping::tpload, 73
 - h, 74
 - hm0, 74
 - phi, 75
 - tm_10, 75
- typedefinitionsrtooverlapping::tpoverlapping, 75
 - qo, 76
 - z2, 76
- typedefinitionsrtooverlapping::tpoverlappinginput, 76
 - computedoverlapping, 78
 - criticaloverlapping, 78
 - factordeterminationq_b_f_b, 78
 - factordeterminationq_b_f_n, 78
 - fshallow, 78
 - m_z2, 78
 - reductionfactorforeshore, 78
 - relaxationfactor, 78
- validateinputc
 - dllovertopping, 12
- validateinputf
 - dllovertopping, 12
- varmodelfactorcriticaloverlapping
 - overlappinginterface, 51
- verbosity
 - modulelogging::tlogging, 71
- versionnumber
 - dllovertopping, 13
- waveRunup.f90, 89
- waverunup, 57
 - convergedwithresidu, 58
 - determinestartingvalue, 58
 - findsmallestresidu, 58
 - innercalculation, 58
 - iterationwaverunup, 59
- xcoorddiff
 - typedefinitionsrtooverlapping::tpgeometry, 73
- xcoordinate
 - overlappinginterface::tpprofilecoordinate, 80
- xcoordinates
 - typedefinitionsrtooverlapping::tpgeometry, 73
- xcoords
 - overlappinginterface::overlappinggeometrytype, 68
 - overlappinginterface::overlappinggeometrytypef, 70
- xdiff_min
 - typedefinitionsrtooverlapping, 57
- ycoorddiff
 - typedefinitionsrtooverlapping::tpgeometry, 73
- ycoordinates
 - typedefinitionsrtooverlapping::tpgeometry, 73
- ycoords
 - overlappinginterface::overlappinggeometrytype, 68
 - overlappinginterface::overlappinggeometrytypef, 70
- z2

- typedefinitionsrtoover topping::tpover topping, [76](#)
- z2_iter_max1
 - typedefinitionsrtoover topping, [57](#)
- z2_iter_max2
 - typedefinitionsrtoover topping, [57](#)
- z2_margin
 - typedefinitionsrtoover topping, [57](#)
- zFunctionsWTIOver topping.f90, [89](#)
- zcoordinate
 - over toppinginterface::tpprofilecoordinate, [80](#)
- zfunclogratios
 - zfunctionswtiover topping, [64](#)
- zfunctionswtiover topping, [60](#)
 - adjustprofile, [61](#)
 - calculateqorto, [61](#)
 - profileinstructure, [63](#)
 - zfunclogratios, [64](#)