

Dikes Overtopping Kernel - Technical documentation

Generated by Doxygen 1.8.9.1

Mon Jun 27 2016 14:50:18

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	14
4.2	factormoduleovertopping Module Reference	14
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	formulamoduleovertopping 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	21
4.3.2.4	calculatebreakerparameter	21
4.3.2.5	calculatewavelength	22
4.3.2.6	calculatewaveovertoppingdischarge	22
4.3.2.7	calculatewaverunup	23
4.3.2.8	calculatewavesteepness	24
4.3.2.9	cubicroots	25
4.3.2.10	isequalreal	26
4.3.2.11	isequalzero	26
4.3.2.12	realrootscubicfunction	26
4.3.2.13	rootsdepressedcubic	27
4.3.2.14	rootsgeneralcubic	28
4.4	geometrymoduleovertopping Module Reference	29
4.4.1	Detailed Description	30
4.4.2	Function/Subroutine Documentation	30
4.4.2.1	adjustnonhorizontalberms	30
4.4.2.2	allocatevectorsgeometry	31
4.4.2.3	basicgeometrytest	32
4.4.2.4	calculatehorzdistance	33
4.4.2.5	calculatehorzlengths	33
4.4.2.6	calculatesegmentslopes	34
4.4.2.7	checkcrosssection	35
4.4.2.8	copygeometry	35
4.4.2.9	deallocategeometry	36
4.4.2.10	determinesegmenttypes	36
4.4.2.11	initializegeometry	37
4.4.2.12	mergesequentialberms	38
4.4.2.13	removeberms	39
4.4.2.14	removedikesegments	40
4.4.2.15	splitcrosssection	40
4.5	mainmoduleovertopping Module Reference	41
4.5.1	Detailed Description	42
4.5.2	Function/Subroutine Documentation	42
4.5.2.1	calculateovertopping	42
4.5.2.2	calculateovertoppingnegativefreeboard	43
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	typedefinitionsovertopping 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	59
4.11.2.4	innercalculation	59
4.11.2.5	iterationwaverunup	60
4.12	zfunctionsovertopping Module Reference	61
4.12.1	Detailed Description	62
4.12.2	Function/Subroutine Documentation	62
4.12.2.1	adjustprofile	62
4.12.2.2	calculateqorto	63
4.12.2.3	profileinstructure	64
4.12.2.4	zfunclogratios	65
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	typedefinitionsovertopping::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	typedefinitionsovertopping::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	typedefinitionsovertopping::tpovertopping 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	typedefinitionsovertopping::tpovertoppinginput Type Reference	76
5.7.1	Detailed Description	78
5.7.2	Member Data Documentation	78
5.7.2.1	computedovertopping	78
5.7.2.2	criticalovertopping	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	overtoppinginterface::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	factorModuleOvertopping.f90 File Reference	82
6.2.1	Detailed Description	82
6.3	formulaModuleOvertopping.f90 File Reference	82
6.3.1	Detailed Description	83
6.4	geometryModuleOvertopping.f90 File Reference	83
6.4.1	Detailed Description	84
6.5	mainModuleOvertopping.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	overtoppingInterface.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	typeDefinitionsOvertopping.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 zFunctionsOvertopping.f90 File Reference	90
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
factormoduleovertopping	Functions for the slope angle and influence factors	14
formulamoduleovertopping	Core computations for Dikes Overtopping	18
geometrymoduleovertopping	Core computations related to the geometry	29
mainmoduleovertopping	Core computations for Dikes Overtopping	41
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
typedefinitionsovertopping	Type definitions for Dikes Overtopping	53
waverunup	Iteration procedure for 2% wave runup	57
zfunctionsovertopping	Module for the Limit State Functions (Z-functions) for wave overtopping	61

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
typedefinitionsovertopping::tpgeometry	
TpGeometry: structure with geometry data	71
typedefinitionsovertopping::tpload	
TpLoad: structure with load parameters	73
typedefinitionsovertopping::tpovertopping	
TpOvertopping: structure with overtopping results	75
typedefinitionsovertopping::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
factorModuleOvertopping.f90	This file contains a module with functions for the slope angle and influence factors	82
formulaModuleOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping	82
geometryModuleOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping related to the geometry	83
mainModuleOvertopping.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
typeDefinitionsOvertopping.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
zFunctionsOvertopping.f90	This file contains the limit state functions for wave overtopping within VTV	90

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*)

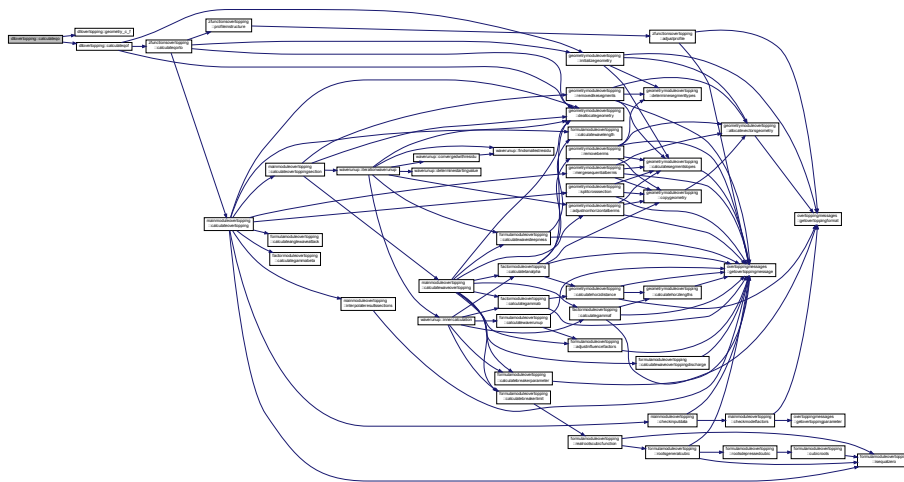
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF: convert C-like input structures to Fortran input structures.

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*)

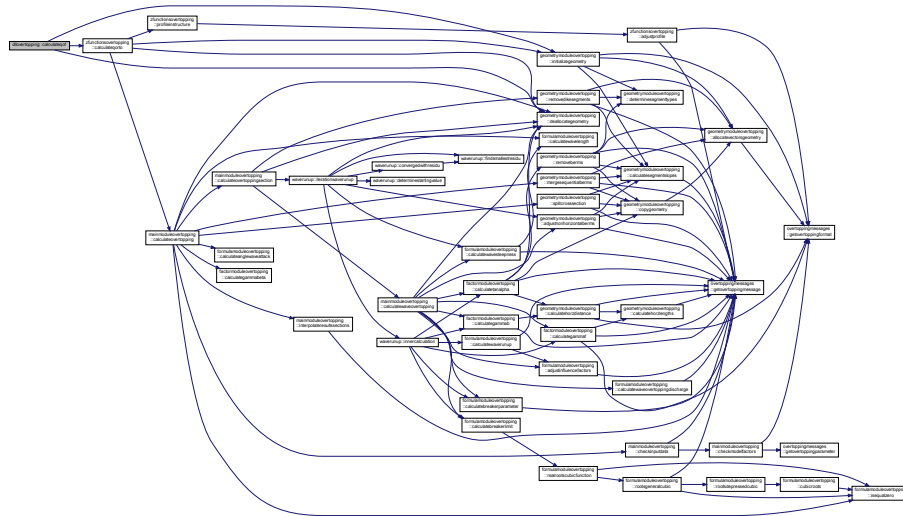
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.

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*)

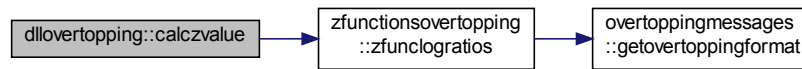
Subroutine that calculates the Z-function Dikes Overtopping based on the discharge calculated with calculateQoF.

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

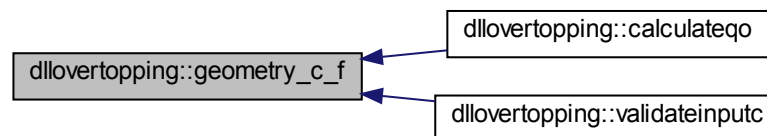
<code>in</code>	<code>geometryInput</code>	struct with geometry and roughness as c-pointers
-----------------	----------------------------	--

Returns

fortran struct with geometry and roughness

Definition at line 329 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 295 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)`

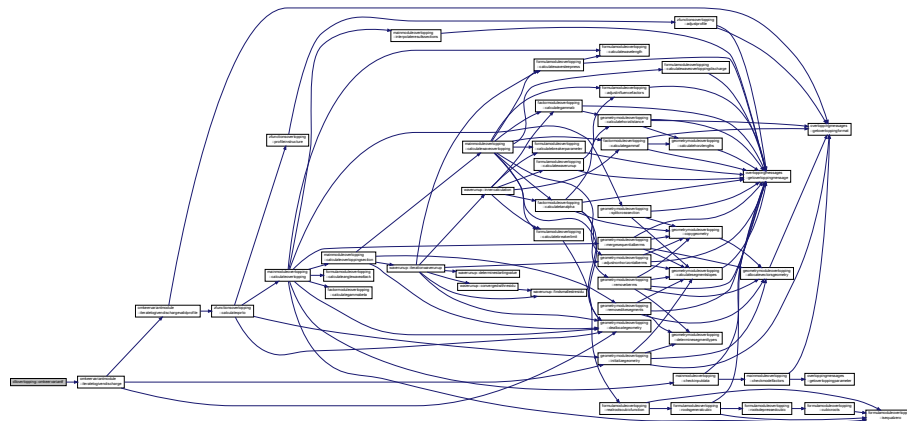
Subroutine with omkeerVariant.

Parameters

in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>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 259 of file dllovertopping.f90.

Here is the call graph for this function:



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

Subroutine that sets the language for error and validation messages.

Definition at line 282 of file dllovertopping.f90.

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

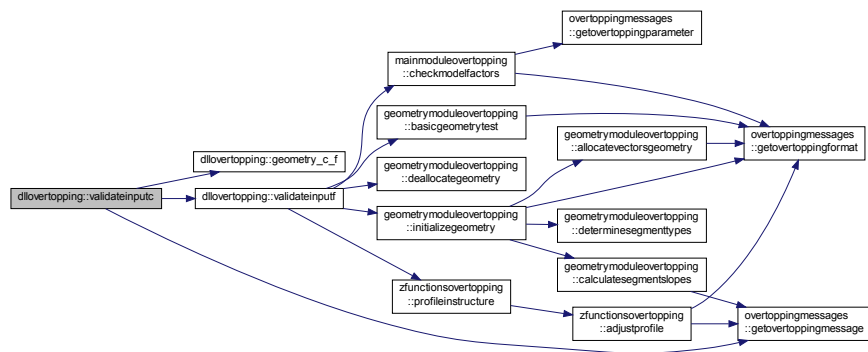
Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.

Parameters

in	<i>geometryinput</i>	struct with geometry and roughness as c-pointers
in	<i>dikeheight</i>	dike height
in,out	<i>modelfactors</i>	struct with modelfactors
out	<i>success</i>	flag for success
out	<i>errortext</i>	error message (only set if not successful)

Definition at line 128 of file dllovertopping.f90.

Here is the call graph for this function:



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

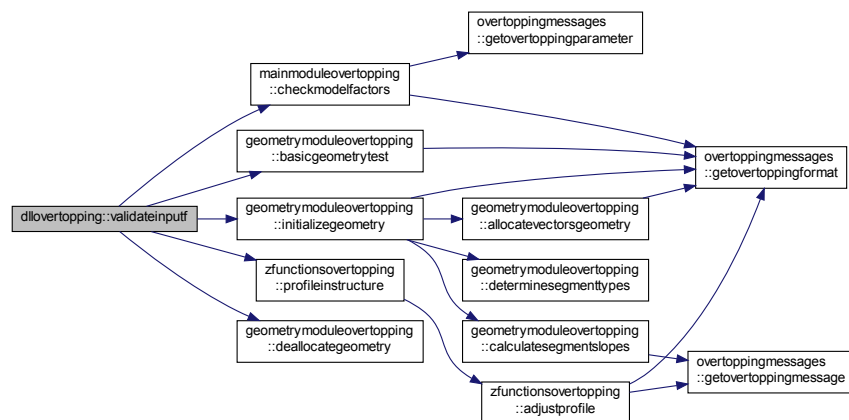
Subroutine that validates the geometry.

Parameters

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

Definition at line 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 307 of file dllovertopping.f90.

4.2 factormoduleovertopping 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 factormoduleovertopping::calculategammab (real(kind=wp), intent(in) *h*, real(kind=wp), intent(in) *Hm0*, real(kind=wp), intent(in) *z2*, type(tpgeometry), intent(in) *geometry*, real(kind=wp), intent(out) *gammaB*, logical, intent(out) *succes*, character(len=*), intent(out) *errorMessage*)

calculateGammaB influence factor berms

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

calculateGammaF influence factor roughness

- 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 `formulamoduleovertopping::adjustinfluencefactors` (`real(kind=wp)`, intent(inout) *gammaB*, `real(kind=wp)`, intent(inout) *gammaF*, `real(kind=wp)`, intent(inout) *gammaBeta*, `integer`, intent(in) *gammaBetaType*, `real(kind=wp)`, intent(in) *ksi0*, `real(kind=wp)`, intent(in) *ksi0Limit*, `logical`, intent(out) *succes*, `character(len=*)`, intent(out) *errorMessage*)

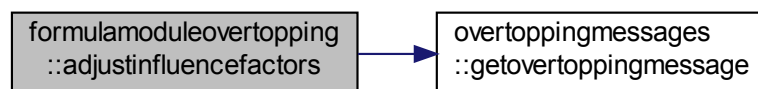
`adjustInfluenceFactors`: adjust the influence factors

Parameters

in, out	<i>gammaB</i>	influence factor berms
in, out	<i>gammaF</i>	influence factor roughness
in, out	<i>gammaBeta</i>	influence factor angle of wave attack
in	<i>gammaBetaType</i>	type influence factor angle of wave attack: 1 = wave run-up, 2 = overtopping
in	<i>ksi0</i>	breaker parameter
in	<i>ksi0Limit</i>	limit value breaker parameter
out	<i>succes</i>	flag for succes
out	<i>errorMessage</i>	error message

Definition at line 382 of file `formulaModuleOvertopping.f90`.

Here is the call graph for this function:



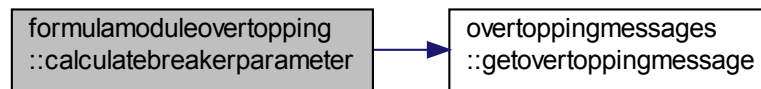
Here is the caller graph for this function:



4.3.2.2 subroutine, public formulamoduleovertopping::calculateanglewaveattack (real(kind=wp), intent(in) *phi*, real(kind=wp),
intent(in) *psi*, real(kind=wp), intent(out) *beta*)

calculateAngleWaveAttack: calculate the angle of wave attack

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.5 subroutine, public formulamoduleovertopping::calculatewavelength (real(kind=wp), intent(in) *Tm_10*, real(kind=wp), intent(out) *L0*)

calculateWaveLength: calculate the wave length

Parameters

in	<i>tm_10</i>	spectral wave period (s)
out	<i>l0</i>	wave length (m)

Definition at line 184 of file formulaModuleOvertopping.f90.

Here is the caller graph for this function:



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

calculateWaveOvertoppingDischarge: calculate the wave overtopping discharge

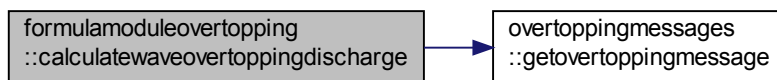
Parameters

in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in	<i>tanalpha</i>	representative slope angle
in	<i>gammab</i>	influence factor berms

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

Definition at line 87 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

calculateWaveRunup: calculate wave runup

Parameters

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

Definition at line 38 of file formulaModuleOvertopping.f90.

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

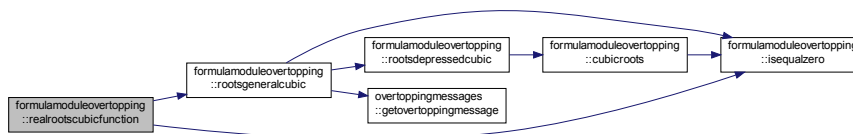
cubicRoots: calculate the roots of a cubic function

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	$succes$	flag for succes
out	$errormessage$	error message

Definition at line 480 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

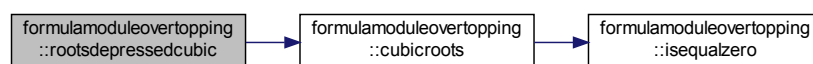
rootsDepressedCubic: calculate the roots of a depressed cubic function

Parameters

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

Definition at line 589 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.14 subroutine formulamoduleovertopping::rootsgeneralcubic (real(kind=wp), intent(in) *a*, real(kind=wp), intent(in) *b*,
real(kind=wp), intent(in) *c*, real(kind=wp), intent(in) *d*, double complex, dimension(3), intent(out) *z*, logical, intent(out)
succes, character(len=*), intent(out) *errorMessage*) [private]

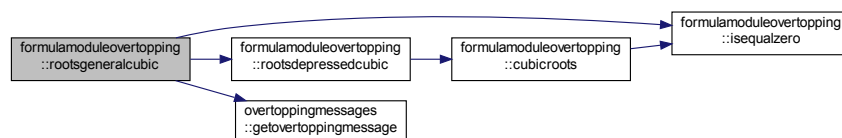
rootsGeneralCubic: calculate the roots of a generic cubic function

Parameters

in	a	coefficients a cubic function
in	b	coefficients b cubic function
in	c	coefficients c cubic function
in	d	coefficients d cubic function
out	z	roots cubic function
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 536 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4 geometrymoduleoverlapping Module Reference

core computations related to the geometry

Functions/Subroutines

- subroutine, public [checkcrosssection](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, succes, errorMessage)
checkCrossSection: check cross section
- subroutine, public [initializegeometry](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, geometry, succes, errorMessage)
initializeGeometry: initialize the geometry
- subroutine, public [allocatevectorsgeometry](#) (nCoordinates, geometry, succes, errorMessage)
allocateVectorsGeometry: allocate the geometry vectors
- subroutine, public [deallocategeometry](#) (geometry)
deallocateGeometry: deallocate the geometry vectors
- subroutine, public [calculatesegmentsslopes](#) (geometry, succes, errorMessage)
calculateSegmentSlopes: calculate the segment slopes
- subroutine, public [determinesegmenttypes](#) (geometry)
determineSegmentTypes: determine the segment types
- subroutine, public [copygeometry](#) (geometry, geometryCopy, succes, errorMessage)
copyGeometry: copy a geometry structure
- subroutine, public [mergesequentialberms](#) (geometry, geometryMergedBerms, succes, errorMessage)
mergeSequentialBerms: merge sequential berms
- subroutine, public [adjustnonhorizontalberms](#) (geometry, geometryFlatBerms, succes, errorMessage)

- 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 `geometrymoduleovertopping::adjustnonhorizontalberms` (type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(out) *geometryFlatBerms*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

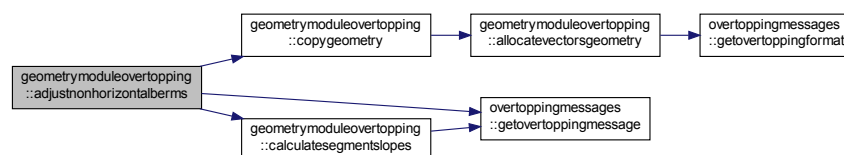
adjustNonHorizontalBerms: adjust non-horizontal berms

Parameters

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

Definition at line 529 of file `geometryModuleOvertopping.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.2 subroutine, public geometrymoduleovertopping::allocatevectorsgeometry (integer, intent(in) *nCoordinates*, type (tpgeometry), intent(inout) *geometry*, logical, intent(out) *succes*, character(len=*), intent(inout) *errorMessage*)

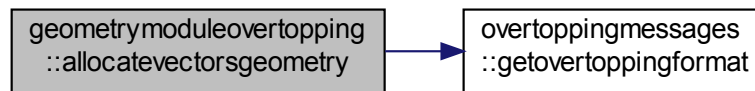
allocateVectorsGeometry: allocate the geometry vectors

Parameters

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

Definition at line 225 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

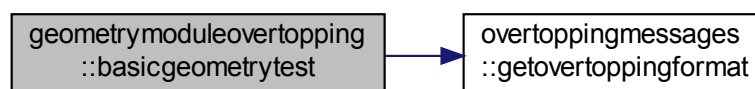
basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)

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 geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

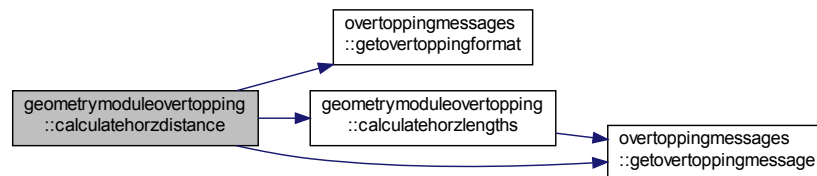
calculateHorzDistance: calculate horizontal distance

Parameters

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

Definition at line 983 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



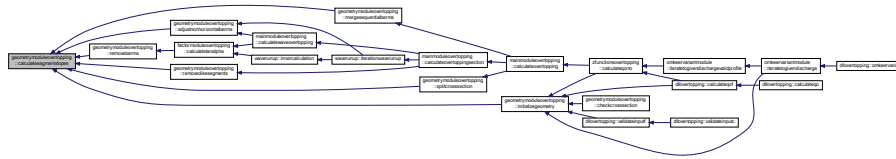
Here is the caller graph for this function:



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

calculateHorzLengths: calculate horizontal lengths

Here is the caller graph for this function:



4.4.2.7 subroutine, public geometrymoduleovertopping::checkcrosssection (real(kind=wp), intent(in) *psi*, integer, intent(in) *nCoordinates*, real(kind=wp), dimension (ncoordinates), intent(in) *xCoordinates*, real(kind=wp), dimension (ncoordinates), intent(in) *yCoordinates*, real(kind=wp), dimension(ncoordinates-1), intent(in) *roughnessFactors*, logical, intent(out) *succes*, character(len=*), intent(out) *errorMessage*)

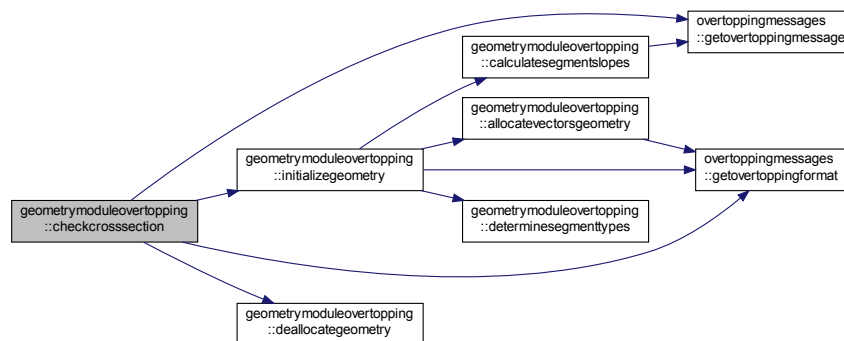
checkCrossSection: check cross section

Parameters

in	<i>psi</i>	dike normal (degrees)
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 geometryModuleOvertopping.f90.

Here is the call graph for this function:



4.4.2.8 subroutine, public geometrymoduleovertopping::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

Here is the caller graph for this function:



4.4.2.11 subroutine, public geometrymoduleovertopping::initializegeometry (real(kind=wp), intent(in) *psi*, integer, intent(in) *nCoordinates*, real(kind=wp), dimension(ncoordinates), intent(in) *xCoordinates*, real(kind=wp), dimension(ncoordinates), intent(in) *yCoordinates*, real(kind=wp), dimension(ncoordinates-1), intent(in) *roughnessFactors*, type(tpgeometry), intent(out) *geometry*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

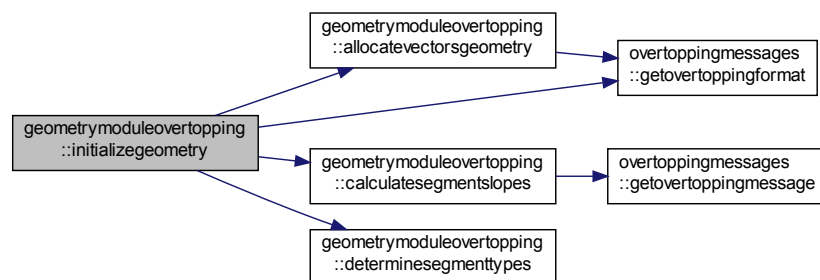
initializeGeometry: initialize the geometry

Parameters

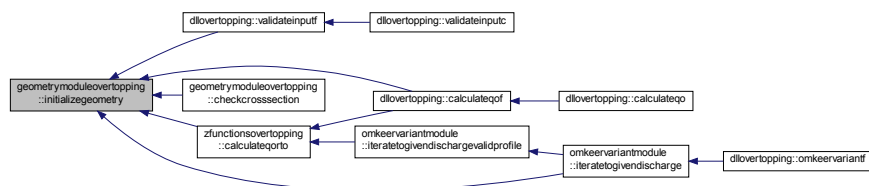
in	<i>psi</i>	dike normal (degree)
in	<i>ncoordinates</i>	number of coordinates
in	<i>xcoordinates</i>	x-coordinates (m)
in	<i>ycoordinates</i>	y-coordinates (m+NAP)
in	<i>roughnessfactors</i>	roughness factors
out	<i>geometry</i>	structure with geometry data
out	<i>succes</i>	flag for succes
out	<i>errormessage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

Definition at line 149 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

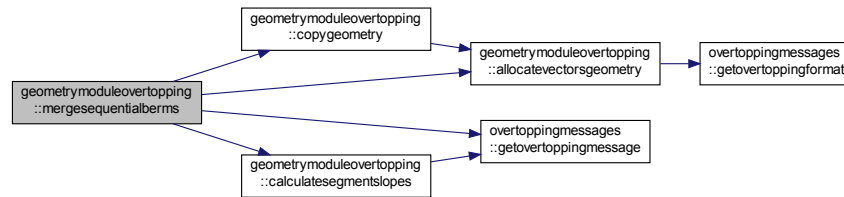
mergeSequentialBerms: merge sequential berms

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 geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

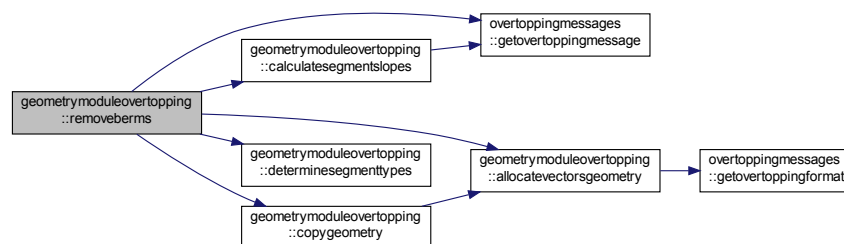
removeBerms: remove berms

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 geometryModuleOvertopping.f90.

Here is the call graph for this function:



[illegible]

removeDikeSegments: remove dike segments

Parameters

Definition at line 718 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:

```

graph LR
    A["geometrymoduleovertopping  
::removedikesegments"] --> B["geometrymoduleovertopping  
::calculatesegmentsslopes"]
    A --> C["geometrymoduleovertopping  
::allocatevectorsgeometry"]
    A --> D["geometrymoduleovertopping  
::determinesegmenttypes"]
    A --> E["overtoppingmessages  
::getovertoppingmessage"]
    B --> E
    C --> F["overtoppingmessages  
::getovertoppingformat"]
  
```

The call graph for `geometrymoduleovertopping::removedikesegments` shows the following structure:

- geometrymoduleovertopping::removedikesegments** (shaded box) is the entry point. It has four outgoing calls:
 - to `geometrymoduleovertopping::calculatesegmentsslopes`
 - to `geometrymoduleovertopping::allocatevectorsgeometry`
 - to `geometrymoduleovertopping::determinesegmenttypes`
 - to `overtoppingmessages::getovertoppingmessage`
- `geometrymoduleovertopping::calculatesegmentsslopes` has one outgoing call to `overtoppingmessages::getovertoppingmessage`.
- `geometrymoduleovertopping::allocatevectorsgeometry` has one outgoing call to `overtoppingmessages::getovertoppingformat`.

```

graph LR
    A[geometry module overtopping  
- removed ke segments] --> B[main module overtopping  
- calculate overtopping section]
    B --> C[main module overtopping  
- calculate overtopping]
    C --> D[z function overtopping  
- calculate q to b]
    D --> E[d b overtopping - calculate q f]
    E --> F[d b overtopping - calculate q p]
    F --> G[omkeer variant module  
- calculate q to b]
    G --> H[omkeer variant module  
- iteratie q vendscharge galsid profile]
    H --> I[omkeer variant module  
- iteratie q vendscharge]
    I --> J[d b overtopping - omkeer variant]
    J --> D
  
```

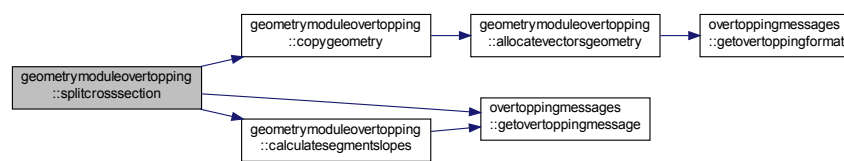
splitCrossSection: split a cross section

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 geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5 mainmoduleovertopping 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, dimErrMsg, errorMessages, ierr)
checkModelFactors: check the input data

4.5.1 Detailed Description

core computations for Dikes Overtopping

4.5.2 Function/Subroutine Documentation

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

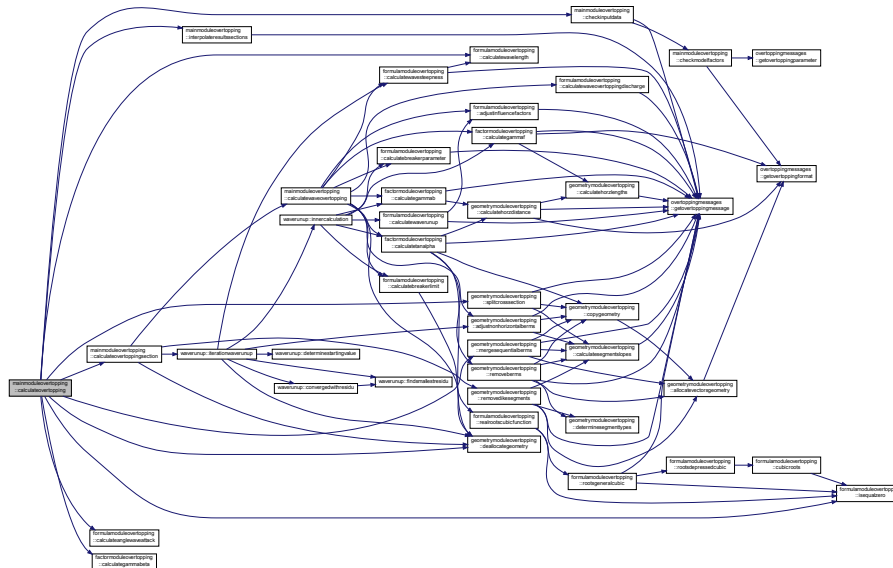
calculateOvertopping: calculate the overtopping

Parameters

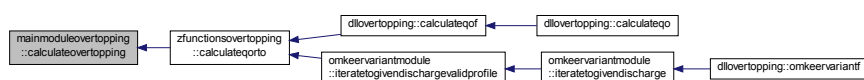
in	<i>geometry</i>	structure with geometry data
in	<i>load</i>	structure with load parameters
in	<i>modelfactors</i>	structure with model factors
out	<i>overtopping</i>	structure with overtopping results
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 39 of file mainModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.2 subroutine mainmoduleovertopping::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 mainModuleOvertopping.f90.

Here is the call graph for this function:



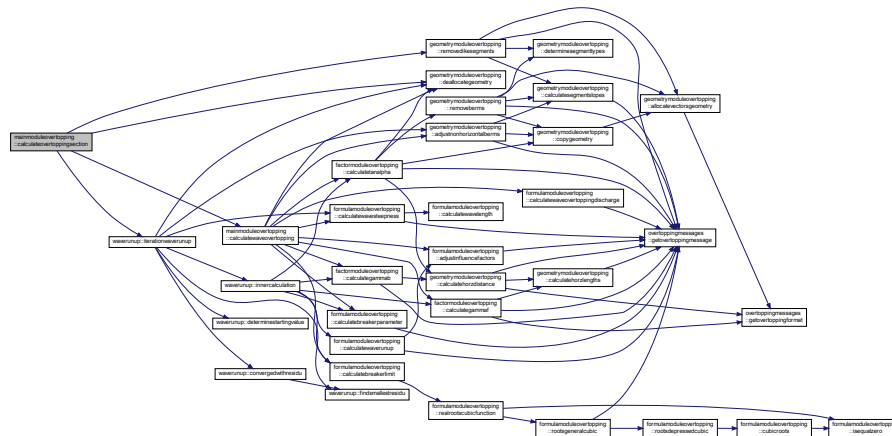
4.5.2.3 subroutine, public mainmoduleovertopping::calculateovertoppingsection (type (tpgeometry), intent(in) *geometry*, real(kind=wp), intent(in) *h*, real(kind=wp), intent(in) *Hm0*, real(kind=wp), intent(in) *Tm_10*, real(kind=wp), intent(in) *L0*, real(kind=wp), intent(inout) *gammaBeta_z*, real(kind=wp), 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

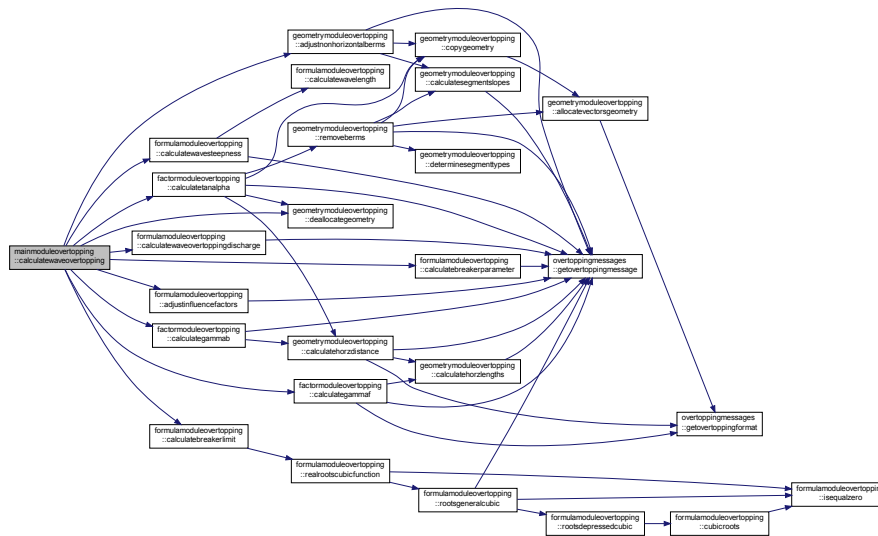
Definition at line 168 of file mainModuleOvertopping.f90.



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>z2</i>	2% wave run-up (m)
in, out	<i>gammabeta_o</i>	influence angle wave attack overtopping
in	<i>modelfactors</i>	structure with model factors
out	<i>qo</i>	wave overtopping discharge (m3/m per s)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Here is the call graph for this function:



Here is the caller graph for this function:



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

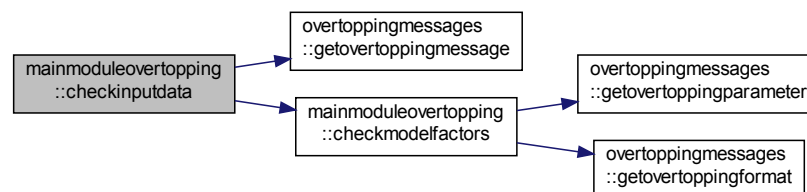
checkInputdata: check the input data

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 mainModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

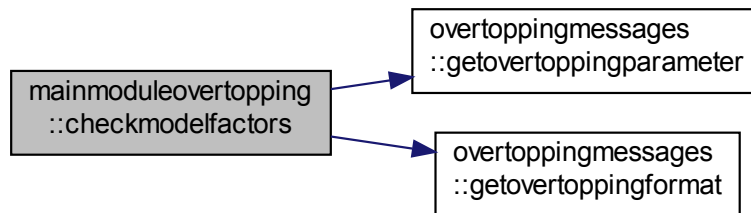
checkModelFactors: check the input data

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 660 of file mainModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.7 subroutine, public mainmoduleovertopping::interpolateresultssections (type (tpgeometry), intent(in) *geometry*, real(kind=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*)

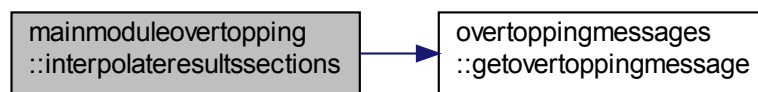
interpolateResultsSections: interpolate results for split cross sections

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 mainModuleOvertopping.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.

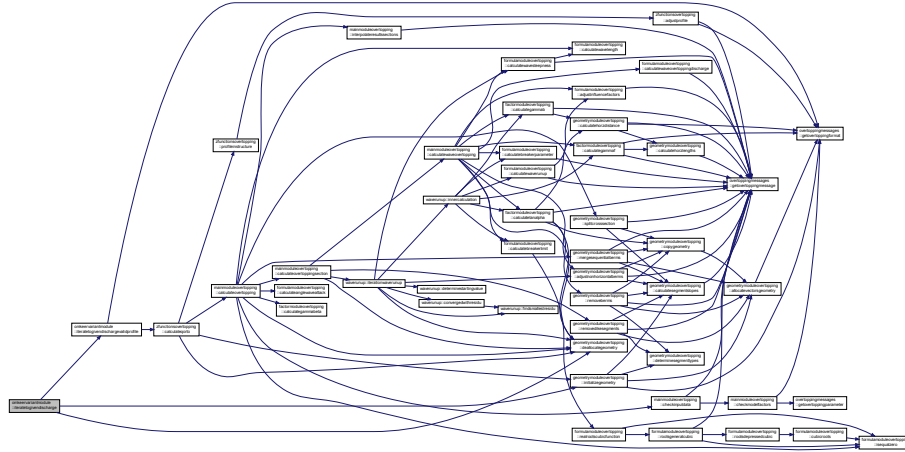
Parameters

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

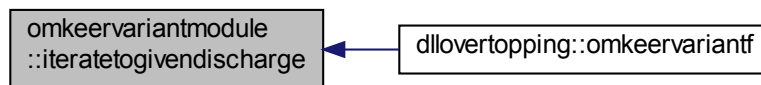
out	<i>errortext</i>	error message (only set if not successful)
in	<i>logging</i>	logging struct

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.

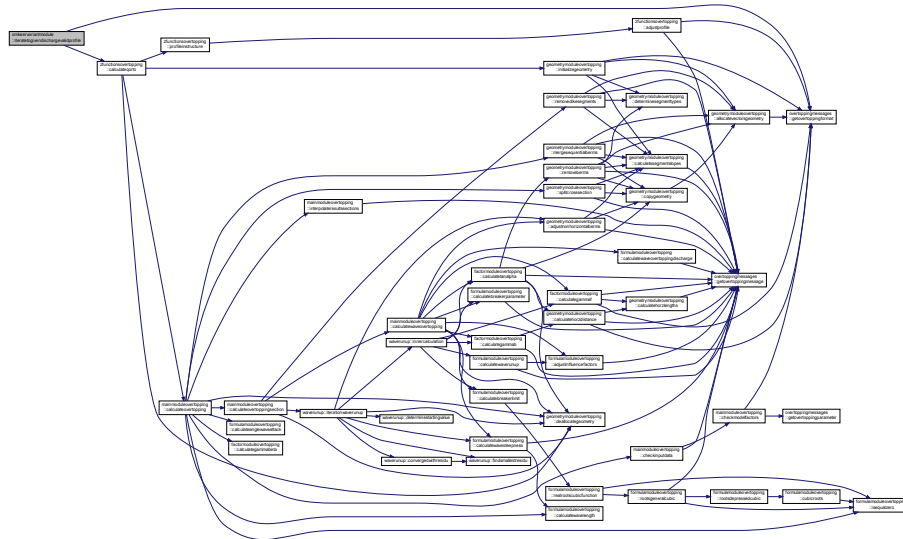
Parameters

in	<i>geometry</i>	internal structure with geometry data
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>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

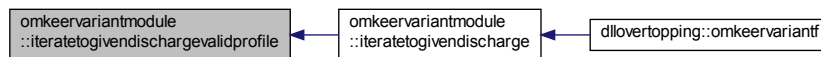
out	errortext	error message (only set if not successful)
-----	-----------	--

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 100 of file OvertoppingMessages.f90.

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

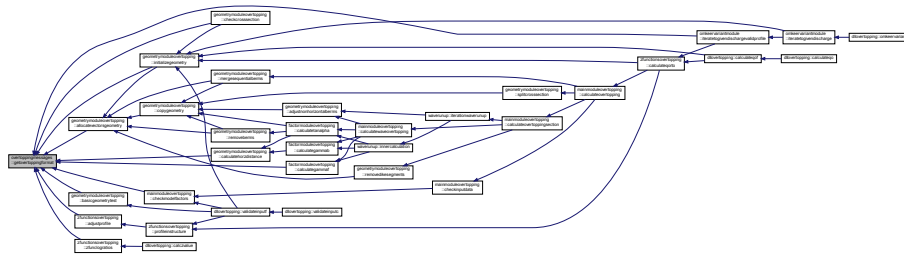
Subroutine that returns a Fortran format string with the corresponding ID in the current language.

Parameters

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

Definition at line 260 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)

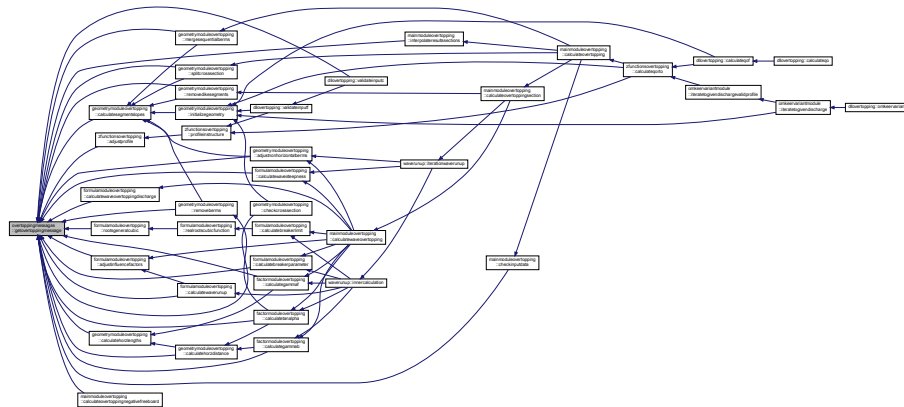
Subroutine that returns a message with the corresponding ID in the current language.

Parameters

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

Definition at line 110 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)

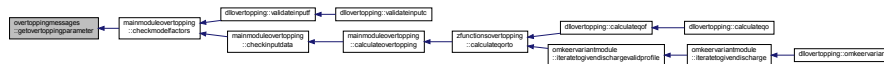
Subroutine that returns the name of an input parameter with the corresponding ID in the current language.

Parameters

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

Definition at line 318 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 recognized (lower and upper case)

Parameters

in	lang	new language ID to be used
----	------	----------------------------

Definition at line 82 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 typedefinitionsovertopping 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(kind=wp), parameter `xdiff_min` = 2.0d-2
minimal value distance between x-coordinates (m)
- real(kind=wp), parameter `margin_diff` = 1.0d-14
margin for minimal distance (m)
- real(kind=wp), parameter `berm_min` = 0.0d0
minimal value gradient berm segment
- real(kind=wp), parameter `berm_max` = 1.0d0/15
maximal value gradient berm segment
- real(kind=wp), parameter `slope_min` = 1.0d0/8
minimal value gradient slope segment
- real(kind=wp), parameter `slope_max` = 1.0d0
maximal value gradient slope segment
- real(kind=wp), parameter `margin_grad` = 0.0025d0
margin for minimal and maximal gradients
- real(kind=wp), parameter `rfactor_min` = 0.5d0
minimal value roughness factor dike segments
- real(kind=wp), parameter `rfactor_max` = 1.0d0
maximal value roughness factor dike segments
- real(kind=wp), parameter `mz2_min` = 0.0d0
minimal value model factor of 2% runup height
- real(kind=wp), parameter `mz2_max` = huge(mz2_max)
maximal value model factor of 2% runup height
- real(kind=wp), parameter `fb_min` = 0.0d0
minimal value model factor for breaking waves
- real(kind=wp), parameter `fb_max` = huge(fb_max)
maximal value model factor for breaking waves
- real(kind=wp), parameter `fn_min` = 0.0d0
minimal value model factor for non-breaking waves
- real(kind=wp), parameter `fn_max` = huge(fn_max)
maximal value model factor for non-breaking waves
- real(kind=wp), parameter `fs_min` = 0.0d0
minimal value model factor for shallow waves
- real(kind=wp), parameter `fs_max` = huge(fs_max)
maximal value model factor for shallow waves
- real(kind=wp), parameter `foreshore_min` = 0.3d0
minimal value reduction factor foreshore
- real(kind=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(kind=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(kind=wp), parameter typedefinitionsovertopping::berm_max = 1.0d0/15`

maximal value gradient berm segment

Definition at line 70 of file typeDefinitionsOvertopping.f90.

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

minimal value gradient berm segment

Definition at line 69 of file typeDefinitionsOvertopping.f90.

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

maximal value model factor for breaking waves

Definition at line 79 of file typeDefinitionsOvertopping.f90.

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

minimal value model factor for breaking waves

Definition at line 78 of file typeDefinitionsOvertopping.f90.

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

maximal value model factor for non-breaking waves

Definition at line 81 of file typeDefinitionsOvertopping.f90.

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

minimal value model factor for non-breaking waves

Definition at line 80 of file typeDefinitionsOvertopping.f90.

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

maximal value reduction factor foreshore

Definition at line 85 of file typeDefinitionsOvertopping.f90.

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

minimal value reduction factor foreshore

Definition at line 84 of file typeDefinitionsOvertopping.f90.

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

Definition at line 42 of file typeDefinitionsOvertopping.f90.

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

Definition at line 43 of file typeDefinitionsOvertopping.f90.

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

Definition at line 44 of file typeDefinitionsOvertopping.f90.

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

maximal value model factor for shallow waves

Definition at line 83 of file typeDefinitionsOvertopping.f90.

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

minimal value model factor for shallow waves

Definition at line 82 of file typeDefinitionsOvertopping.f90.

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

margin for minimal distance (m)

Definition at line 68 of file typeDefinitionsOvertopping.f90.

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

margin for minimal and maximal gradients

Definition at line 73 of file typeDefinitionsOvertopping.f90.

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

maximal value model factor of 2% runup height

Definition at line 77 of file typeDefinitionsOvertopping.f90.

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

minimal value model factor of 2% runup height

Definition at line 76 of file typeDefinitionsOvertopping.f90.

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

maximal value roughness factor dike segments

Definition at line 75 of file typeDefinitionsOvertopping.f90.

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

minimal value roughness factor dike segments

Definition at line 74 of file typeDefinitionsOvertopping.f90.

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

maximal value gradient slope segment

Definition at line 72 of file typeDefinitionsOvertopping.f90.

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

minimal value gradient slope segment

Definition at line 71 of file typeDefinitionsOvertopping.f90.

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

minimal value distance between x-coordinates (m)

Definition at line 67 of file typeDefinitionsOvertopping.f90.

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

maximal number of iterations for calculation z2 part 1

Definition at line 86 of file typeDefinitionsOvertopping.f90.

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

maximal number of iterations for calculation z2 part 1 & 2

Definition at line 87 of file typeDefinitionsOvertopping.f90.

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

margin for convergence criterium calculation z2

Definition at line 88 of file typeDefinitionsOvertopping.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)

innerCalculation: inner calculation for the wave runup

- real(kind=wp) function [determinestartingvalue](#) (i, relaxationFactor, z2_start, z2_end, Hm0)
determineStartingValue: helper function to find a start value for z2
- integer function [findsmallestresidu](#) (z2_start, z2_end, n)
findSmallestResidu: helper function to find the smallest residu
- subroutine [convergedwithresidu](#) (z2_start, z2_end)
convergedWithResidu: helper function to handle convergence with a higher residu than expected if logging is enabled, it writes a small message to the logfile

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]

`convergedWithResidu`: helper function to handle convergence with a higher residu than expected if logging is enabled, it writes a small message to the logfile

Parameters

in	<code>z2_start</code>	array with z2 values at the start of the iteration i
in, out	<code>z2_end</code>	array with z2 values at the end of iteration i

Definition at line 352 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]

`determineStartingValue`: helper function to find a start value for z2

Parameters

in	<i>i</i>	current iteration number
in	<i>relaxationfactor</i>	relaxation factor as given by user
in	<i>z2_start</i>	array with z2 values at the start of the iteration i
in	<i>z2_end</i>	array with z2 values at the end of iteration i
in	<i>hm0</i>	significant wave height

Returns

return value: start value for z2 in current iteration

Definition at line 271 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]

findSmallestResidu: helper function to find the smallest residu

Parameters

in	<i>z2_start</i>	array with z2 values at the start of the iteration i
in	<i>z2_end</i>	array with z2 values at the end of iteration i
in	<i>n</i>	number of iterations already done

Definition at line 309 of file waveRunup.f90.

Here is the caller graph for this function:



4.11.2.4 real(kind=wp) function waverunup::innercalculation (type (tpgeometry), intent(in) *geometry*, real(kind=wp), intent(in) *h*, real(kind=wp), intent(in) *Hm0*, real(kind=wp), intent(inout) *gammaBeta_z*, type (tpoverlappinginput), intent(in) *modelFactors*, real(kind=wp), intent(in) *z2*, real(kind=wp), intent(in) *s0*, type (tpgeometry), intent(in) *geometryFlatBerms*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*) [private]

innerCalculation: inner calculation for the wave runup

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, out	<i>gammabeta_z</i>	influence factor angle wave attack 2% run-up
in	<i>modelfactors</i>	structure with model factors

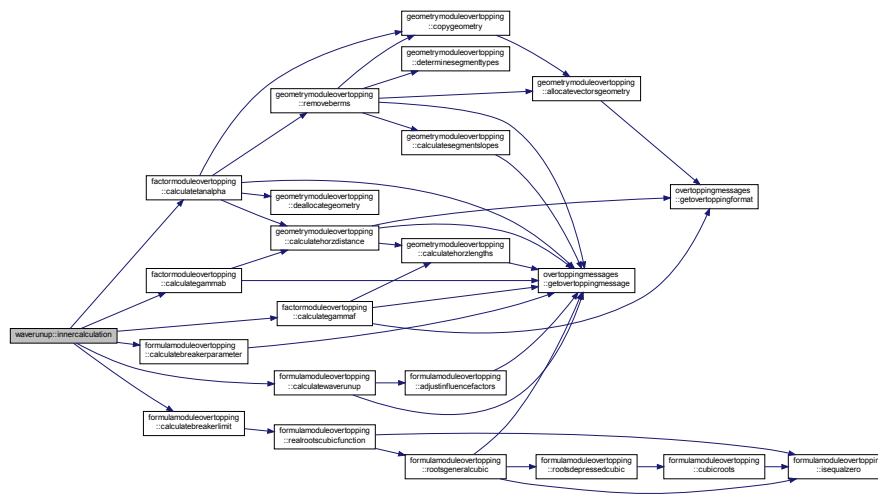
in	<i>z2</i>	2% wave run-up (m)
in	<i>s0</i>	wave steepness
in	<i>geometryflat-berms</i>	structure with geometry data with horizontal berms
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Returns

2% wave run-up at end of inner calculation

Definition at line 160 of file waveRunup.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.5 subroutine, public waverunup::iterationwaverunup (type (tpgeometry), intent(in) *geometry*, real(kind=wp), intent(in) *h*, real(kind=wp), intent(in) *Hm0*, real(kind=wp), intent(in) *Tm_10*, real(kind=wp), intent(inout) *gammaBeta_z*, type (tpoverlappinginput), intent(in) *modelFactors*, real(kind=wp), intent(out) *z2*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

iterationWaveRunup: iteration for the wave runup

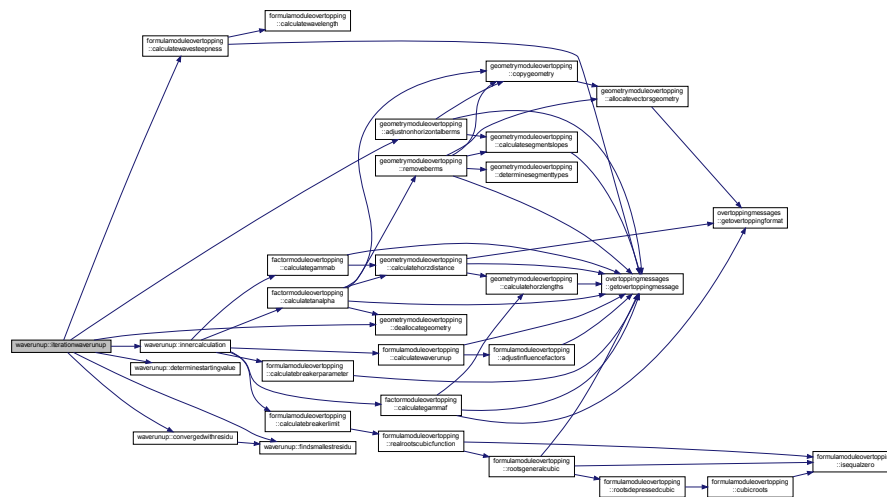
Parameters

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, out	<i>gammabeta_z</i>	influence factor angle wave attack 2% run-up
in	<i>modelfactors</i>	structure with model factors
out	<i>z2</i>	2% wave run-up (m)
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 38 of file waveRunup.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12 zfunctionsovertopping Module Reference

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

Functions/Subroutines

- subroutine, public [calculateqorto](#) (dikeHeight, modelFactors, overtopping, load, geometry, succes, error←Message)

Subroutine to calculate the overtopping discharge with the Overtopping dll.

- subroutine, public [profileinstructure](#) (nrCoordinates, xcoordinates, ycoordinates, dikeHeight, nrCoords←Adjusted, xCoordsAdjusted, zCoordsAdjusted, succes, errorMessage)

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

- subroutine [adjustprofile](#) (nrCoordinates, coordinates, dikeHeight, nrCoordsAdjusted, xCoordsAdjusted, z←CoordsAdjusted, succes, errorMessage)

Subroutine adjust the profile due to a desired dike height.

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

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

4.12.1 Detailed Description

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

4.12.2 Function/Subroutine Documentation

4.12.2.1 subroutine `zfunctionsovertopping::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]

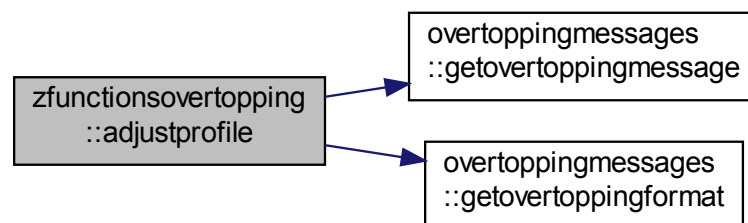
Subroutine adjust the profile due to a desired dike height.

Parameters

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

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

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.2.2 subroutine, public zfunctionsovertopping::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)

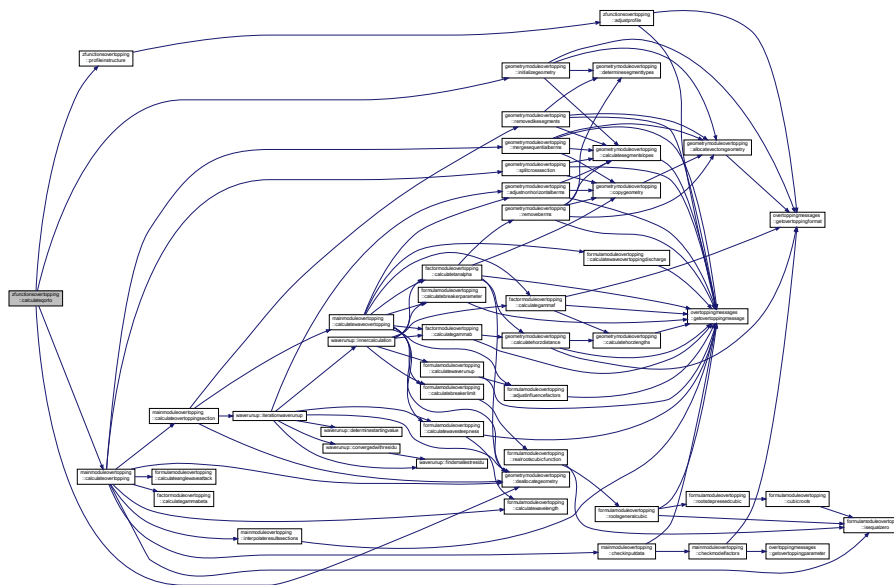
Subroutine to calculate the overtopping discharge with the Overtopping dll.

Parameters

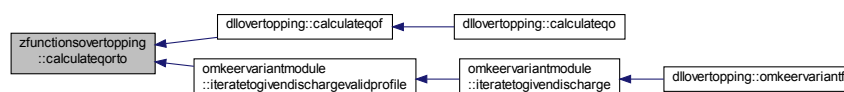
in	<i>dikeheight</i>	dike height
in, out	<i>modelfactors</i>	struct with model factors
out	<i>overtopping</i>	structure with overtopping results
in	<i>geometry</i>	structure with geometry data
in	<i>load</i>	structure with load parameters
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error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

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

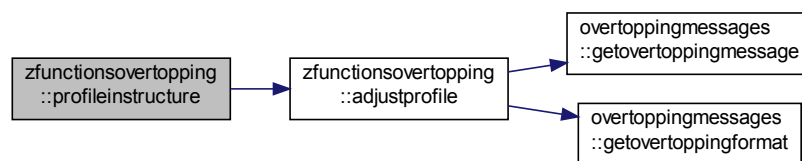
Subroutine to fill the profile in a structure and call the adjustment function of the profile due to a desired dike height.

Parameters

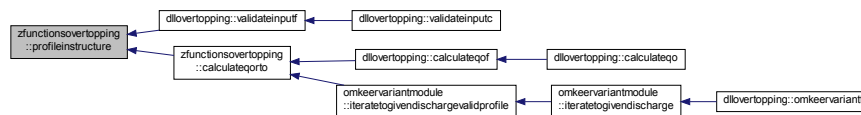
in	<i>nrcoordinates</i>	number of coordinates of the profile
in	<i>xcoordinates</i>	vector with x-coordinates of the profile
in	<i>ycoordinates</i>	vector with y-coordinates of the profile
in	<i>dikeheight</i>	dike height
out	<i>nrcoordsadjusted</i>	number of coordinates in the adjusted profile
	<i>xcoordsadjusted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

Definition at line 83 of file zFunctionsOvertopping.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 zfunctionsovertopping::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

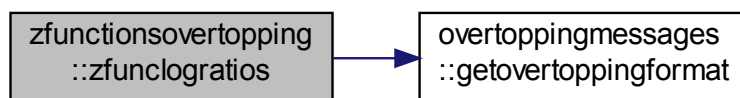
in	<i>qo</i>	computed overtopping discharge
in	<i>qc</i>	Critical overtopping discharge
in	<i>mqo</i>	Model factor computed overtopping discharge
in	<i>mqc</i>	Model factor Critical overtopping discharge
out	<i>success</i>	Flag for succes
out	<i>errormessage</i>	error message, only set if not successful

Returns

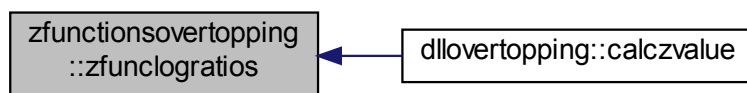
Value z-function

Definition at line 212 of file zFunctionsOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

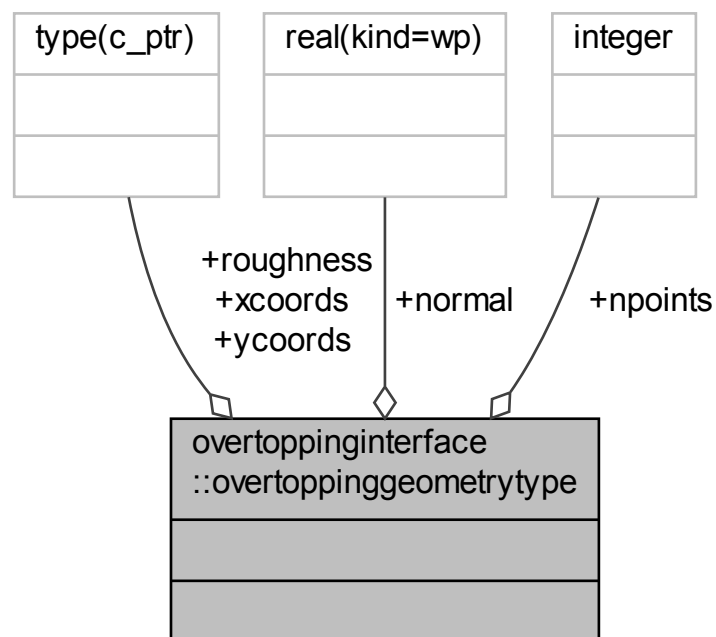


Chapter 5

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`

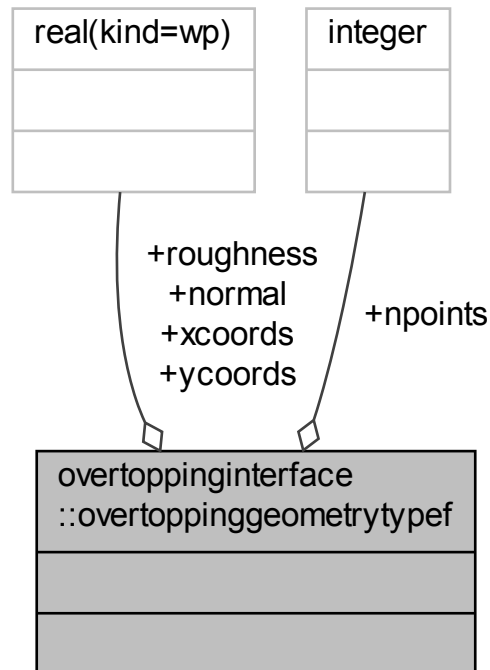
Definition at line 28 of file overtoppingInterface.f90.

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`.

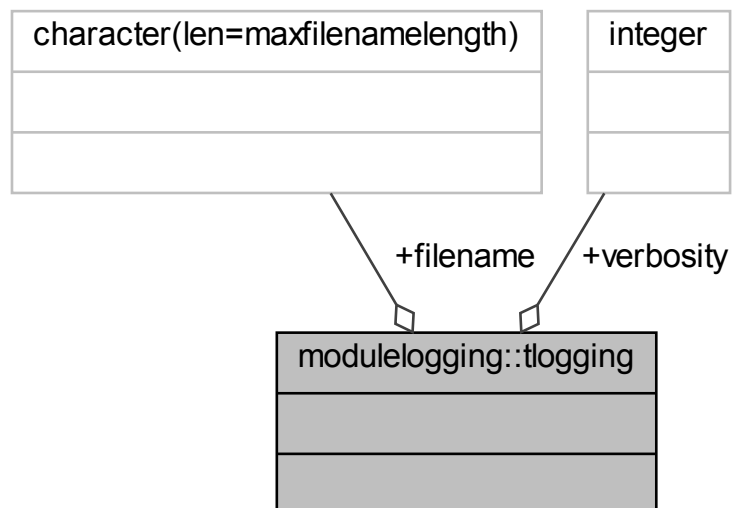
5.2.2.5 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::ycoords`

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`

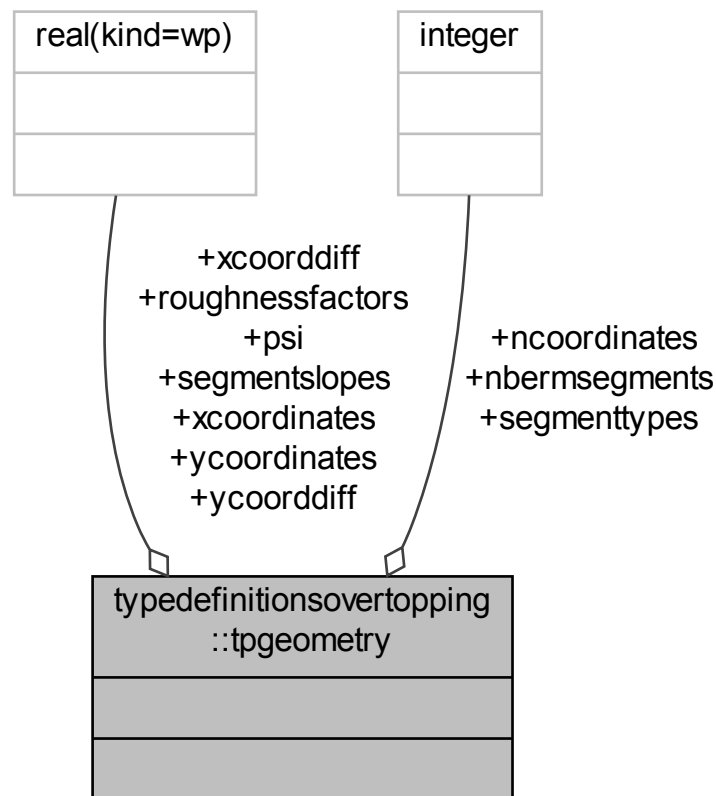
level of verbosity: one of verboseNone, verboseBasic, verboseDetailed, verboseDebugging

Definition at line 19 of file ModuleLogging.f90.

5.4 typedefinitionsovertopping::tpgeometry Type Reference

tpGeometry: structure with geometry data

Collaboration diagram for typedefinitionsovertopping::tpgeometry:



Public Attributes

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

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

5.4.1 Detailed Description

tpGeometry: structure with geometry data

Definition at line 21 of file typeDefinitionsOvertopping.f90.

5.4.2 Member Data Documentation

5.4.2.1 integer typedefinitionsovertopping::tpgeometry::nbermsements

number of berm segments

Definition at line 31 of file typeDefinitionsOvertopping.f90.

5.4.2.2 integer typedefinitionsovertopping::tpgeometry::ncoordinates

number of coordinates cross section

Definition at line 23 of file typeDefinitionsOvertopping.f90.

5.4.2.3 real(kind=wp) typedefinitionsovertopping::tpgeometry::psi

dike normal (degrees)

Definition at line 22 of file typeDefinitionsOvertopping.f90.

5.4.2.4 real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::roughnessfactors => null()

vector with roughness factors cross section

Definition at line 26 of file typeDefinitionsOvertopping.f90.

5.4.2.5 `real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::segmentslopes => null()`

vector with slopes dike segments

Definition at line 29 of file typeDefinitionsOvertopping.f90.

5.4.2.6 `integer, dimension(:), pointer typedefinitionsovertopping::tpgeometry::segmenttypes`

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

Definition at line 30 of file typeDefinitionsOvertopping.f90.

5.4.2.7 `real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::xcoorddiff => null()`

vector with differences in x-coordinates (m)

Definition at line 27 of file typeDefinitionsOvertopping.f90.

5.4.2.8 `real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::xcoordinates => null()`

vector with x-coordinates cross section (m)

Definition at line 24 of file typeDefinitionsOvertopping.f90.

5.4.2.9 `real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::ycoorddiff => null()`

vector with differences in y-coordinates (m)

Definition at line 28 of file typeDefinitionsOvertopping.f90.

5.4.2.10 `real(kind=wp), dimension(:), pointer typedefinitionsovertopping::tpgeometry::ycoordinates => null()`

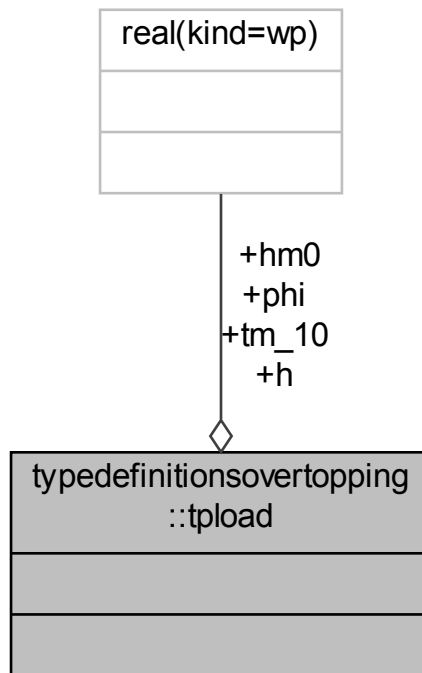
vector with y-coordinates cross section (m+NAP)

Definition at line 25 of file typeDefinitionsOvertopping.f90.

5.5 typedefinitionsovertopping::tpload Type Reference

tpLoad: structure with load parameters

Collaboration diagram for `typedefinitionsovertopping::tload`:



Public Attributes

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

5.5.1 Detailed Description

`tpLoad`: structure with load parameters

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

5.5.2 Member Data Documentation

5.5.2.1 `real(kind=wp)` `typedefinitionsovertopping::tload::h`

local water level (m+NAP)

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

5.5.2.2 `real(kind=wp) typedefinitionsovertopping::tpload::hm0`

significant wave height (m)

Definition at line 37 of file typeDefinitionsOvertopping.f90.

5.5.2.3 `real(kind=wp) typedefinitionsovertopping::tpload::phi`

wave direction (degrees)

Definition at line 39 of file typeDefinitionsOvertopping.f90.

5.5.2.4 `real(kind=wp) typedefinitionsovertopping::tpload::tm_10`

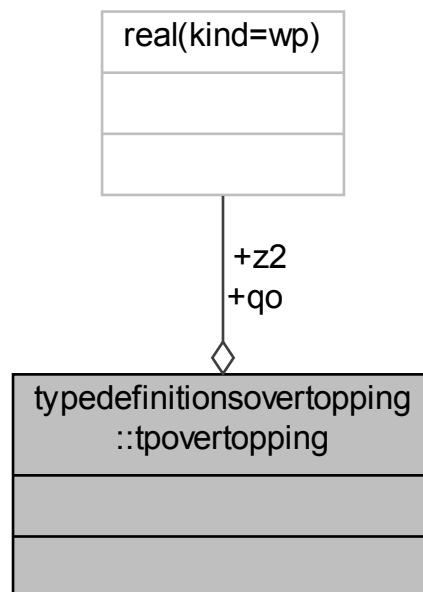
spectral wave period (s)

Definition at line 38 of file typeDefinitionsOvertopping.f90.

5.6 typedefinitionsovertopping::tpovertopping Type Reference

tpOvertopping: structure with overtopping results

Collaboration diagram for typedefinitionsovertopping::tpovertopping:



Public Attributes

- `real(kind=wp)` [z2](#)
2% wave run-up (m)
- `real(kind=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 typeDefinitionsOvertopping.f90.

5.6.2 Member Data Documentation

5.6.2.1 `real(kind=wp) typedefinitionsovertopping::tpovertopping::qo`

wave overtopping discharge (m³/m per s)

Definition at line 61 of file typeDefinitionsOvertopping.f90.

5.6.2.2 `real(kind=wp) typedefinitionsovertopping::tpovertopping::z2`

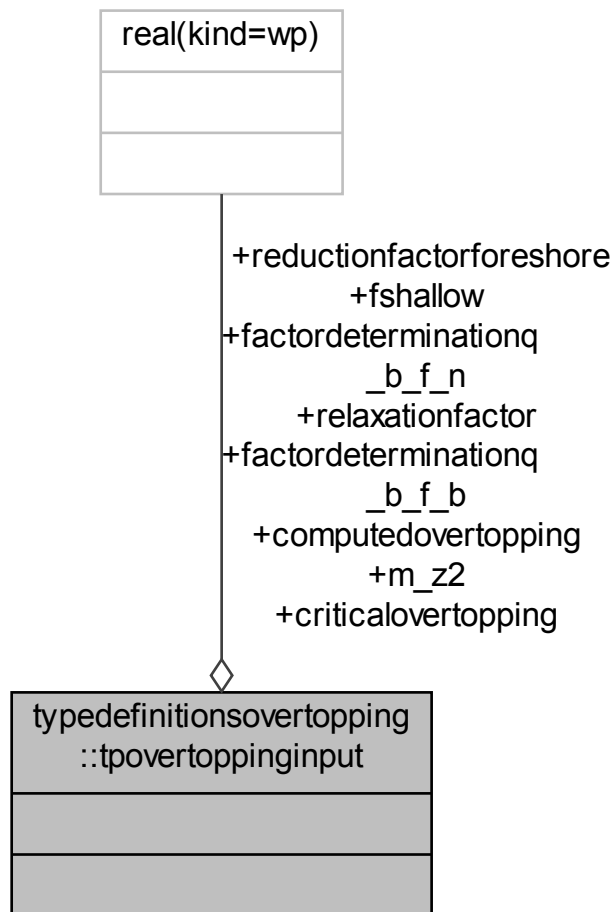
2% wave run-up (m)

Definition at line 60 of file typeDefinitionsOvertopping.f90.

5.7 `typedefinitionsovertopping::tpovertoppinginput` Type Reference

OvertoppingModelFactors: C-structure with model factors.

Collaboration diagram for typedefinitionsovertopping::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 typeDefinitionsOvertopping.f90.

5.7.2 Member Data Documentation

5.7.2.1 `real(kind=wp) typedefinitionsovertopping::tpovertoppinginput::computedovertopping`

model factor computed overtopping

Definition at line 52 of file typeDefinitionsOvertopping.f90.

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

model factor critical overtopping

Definition at line 53 of file typeDefinitionsOvertopping.f90.

5.7.2.3 `real(kind=wp) typedefinitionsovertopping::tpovertoppinginput::factordeterminationq_b_f_b`

model factor for breaking waves

Definition at line 49 of file typeDefinitionsOvertopping.f90.

5.7.2.4 `real(kind=wp) typedefinitionsovertopping::tpovertoppinginput::factordeterminationq_b_f_n`

model factor for non-breaking waves

Definition at line 48 of file typeDefinitionsOvertopping.f90.

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

model factor for shallow waves

Definition at line 51 of file typeDefinitionsOvertopping.f90.

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

model factor describing the uncertainty of 2% runup height

Definition at line 50 of file typeDefinitionsOvertopping.f90.

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

reduction factor foreshore

Definition at line 55 of file typeDefinitionsOvertopping.f90.

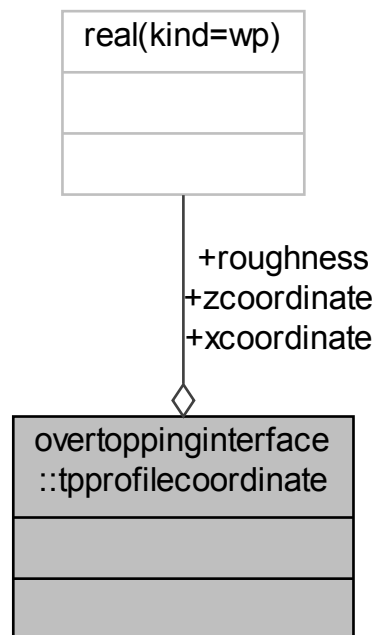
5.7.2.8 `real(kind=wp) typedefinitionsovertopping::tpovertoppinginput::relaxationfactor`

relaxation factor iteration procedure wave runup

Definition at line 54 of file typeDefinitionsOvertopping.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 factorModuleOvertopping.f90 File Reference

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

Modules

- module [factormoduleovertopping](#)
functions for the slope angle and influence factors

Functions/Subroutines

- subroutine, public [factormoduleovertopping::calculatetanalpha](#) (h, Hm0, z2, geometry, tanAlpha, succes, errorMessage)
calculateTanAlpha representative slope angle
- subroutine, public [factormoduleovertopping::calculategammabeta](#) (Hm0, Tm_10, beta, gammaBeta_z, gammaBeta_o)
calculateGammaBeta influence factor angle of wave attack
- subroutine, public [factormoduleovertopping::calculategammaf](#) (h, ksi0, ksi0Limit, gammaB, z2, geometry, gammaF, succes, errorMessage)
calculateGammaF influence factor roughness
- subroutine, public [factormoduleovertopping::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 formulaModuleOvertopping.f90 File Reference

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

Modules

- module [formulamoduleovertopping](#)
the core computations for Dikes Overtopping

Functions/Subroutines

- subroutine, public [formulamoduleovertopping::calculatewaverunup](#) (Hm0, ksi0, ksi0Limit, gammaB, gammaF, gammaBeta, modelFactors, z2, succes, errorMessage)
calculateWaveRunup: calculate wave runup
- subroutine, public [formulamoduleovertopping::calculatewaveovertoppingdischarge](#) (h, Hm0, tanAlpha, gammaB, gammaF, gammaBeta, ksi0, hCrest, modelFactors, Qo, succes, errorMessage)
calculateWaveOvertoppingDischarge: calculate the wave overtopping discharge
- subroutine, public [formulamoduleovertopping::calculatewavelength](#) (Tm_10, L0)
calculateWaveLength: calculate the wave length
- subroutine, public [formulamoduleovertopping::calculatewavesteepness](#) (Hm0, Tm_10, s0, succes, errorMessage)
calculateWaveSteepness: calculate the wave steepness
- subroutine, public [formulamoduleovertopping::calculatebreakerparameter](#) (tanAlpha, s0, ksi0, succes, errorMessage)
calculateBreakerParameter: calculate the breaker parameter
- subroutine, public [formulamoduleovertopping::calculateanglewaveattack](#) (phi, psi, beta)
calculateAngleWaveAttack: calculate the angle of wave attack
- subroutine, public [formulamoduleovertopping::calculatebreakerlimit](#) (gammaB, ksi0Limit, succes, errorMessage)
calculateBreakerLimit: calculate the breaker limit
- subroutine, public [formulamoduleovertopping::adjustinfluencefactors](#) (gammaB, gammaF, gammaBeta, gammaBetaType, ksi0, ksi0Limit, succes, errorMessage)
adjustInfluenceFactors: adjust the influence factors
- subroutine [formulamoduleovertopping::realrootscubicfunction](#) (a, b, c, d, N, x, succes, errorMessage)
realRootsCubicFunction: calculate the roots of a cubic function
- subroutine [formulamoduleovertopping::rootsgeneralcubic](#) (a, b, c, d, z, succes, errorMessage)
rootsGeneralCubic: calculate the roots of a generic cubic function
- subroutine [formulamoduleovertopping::rootsdepressedcubic](#) (p, q, z)
rootsDepressedCubic: calculate the roots of a depressed cubic function
- subroutine [formulamoduleovertopping::cubicroots](#) (z, roots)
cubicRoots: calculate the roots of a cubic function
- logical function, public [formulamoduleovertopping::isequalreal](#) (x1, x2)
isEqualReal: are two reals (almost) equal
- logical function, public [formulamoduleovertopping::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 geometryModuleOvertopping.f90 File Reference

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

Modules

- module [geometrymoduleovertopping](#)
core computations related to the geometry

Functions/Subroutines

- subroutine, public [geometrymoduleovertopping::checkcrosssection](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, succes, errorMessage)
checkCrossSection: check cross section
- subroutine, public [geometrymoduleovertopping::initializegeometry](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, geometry, succes, errorMessage)
initializeGeometry: initialize the geometry
- subroutine, public [geometrymoduleovertopping::allocatevectorsgeometry](#) (nCoordinates, geometry, succes, errorMessage)
allocateVectorsGeometry: allocate the geometry vectors
- subroutine, public [geometrymoduleovertopping::deallocategemetry](#) (geometry)
deallocateGeometry: deallocate the geometry vectors
- subroutine, public [geometrymoduleovertopping::calculatesegmentslopes](#) (geometry, succes, errorMessage)
calculateSegmentSlopes: calculate the segment slopes
- subroutine, public [geometrymoduleovertopping::determinesegmenttypes](#) (geometry)
determineSegmentTypes: determine the segment types
- subroutine, public [geometrymoduleovertopping::copygeometry](#) (geometry, geometryCopy, succes, errorMessage)
copyGeometry: copy a geometry structure
- subroutine, public [geometrymoduleovertopping::mergesesequentialberms](#) (geometry, geometryMergedBerms, succes, errorMessage)
mergeSequentialBerms: merge sequential berms
- subroutine, public [geometrymoduleovertopping::adjustnonhorizontalberms](#) (geometry, geometryFlatBerms, succes, errorMessage)
adjustNonHorizontalBerms: adjust non-horizontal berms
- subroutine, public [geometrymoduleovertopping::removeberms](#) (geometry, geometryNoBerms, succes, errorMessage)
removeBerms: remove berms
- subroutine, public [geometrymoduleovertopping::removedikesegments](#) (geometry, index, geometryAdjusted, succes, errorMessage)
removeDikeSegments: remove dike segments
- subroutine, public [geometrymoduleovertopping::splitcrosssection](#) (geometry, L0, NwideBerms, geometrysectionF, geometrysectionB, succes, errorMessage)
splitCrossSection: split a cross section
- subroutine, public [geometrymoduleovertopping::calculatehorzlengths](#) (geometry, yLower, yUpper, horzLengths, succes, errorMessage)
calculateHorzLengths: calculate horizontal lengths
- subroutine, public [geometrymoduleovertopping::calculatehorzdistance](#) (geometry, yLower, yUpper, dx, succes, errorMessage)
calculateHorzDistance: calculate horizontal distance
- subroutine, public [geometrymoduleovertopping::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 mainModuleOvertopping.f90 File Reference

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

Modules

- module [mainmoduleovertopping](#)
core computations for Dikes Overtopping

Functions/Subroutines

- subroutine, public [mainmoduleovertopping::calculateovertopping](#) (geometry, load, modelFactors, overtopping, succes, errorMessage)
calculateOvertopping: calculate the overtopping
- subroutine, public [mainmoduleovertopping::calculateovertoppingsection](#) (geometry, h, Hm0, Tm_10, L0, gammaBeta_z, gammaBeta_o, modelFactors, overtopping, succes, errorMessage)
calculateOvertoppingSection: calculate the overtopping for a section
- subroutine, public [mainmoduleovertopping::calculatewaveovertopping](#) (geometry, h, Hm0, Tm_10, z2, gammaBeta_o, modelFactors, Qo, succes, errorMessage)
calculateWaveOvertopping: calculate wave overtopping
- subroutine [mainmoduleovertopping::calculateovertoppingnegativefreeboard](#) (load, geometry, overtopping, succes, errorMessage)
calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard
- subroutine, public [mainmoduleovertopping::interpolateresultssections](#) (geometry, L0, NwideBerms, overtoppingB, overtoppingF, overtopping, succes, errorMessage)
interpolateResultsSections: interpolate results for split cross sections
- subroutine, public [mainmoduleovertopping::checkinputdata](#) (geometry, load, modelFactors, succes, errorMessage)
checkInputdata: check the input data
- subroutine, public [mainmoduleovertopping::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::maxfilenamelength` = 256
maximum length of filename
- type(tlogging) `modulelogging::currentlogging`
copy of argument logging

6.6.1 Detailed Description

Module for steering the extra logging.

6.7 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 typeDefinitionsOvertopping.f90 File Reference

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

Data Types

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

Modules

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

Variables

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

- real(kind=wp), parameter [typedefinitionsovertopping::fs_min](#) = 0.0d0
minimal value model factor for shallow waves
- real(kind=wp), parameter [typedefinitionsovertopping::fs_max](#) = huge(fs_max)
maximal value model factor for shallow waves
- real(kind=wp), parameter [typedefinitionsovertopping::foreshore_min](#) = 0.3d0
minimal value reduction factor foreshore
- real(kind=wp), parameter [typedefinitionsovertopping::foreshore_max](#) = 1.0d0
maximal value reduction factor foreshore
- integer, parameter [typedefinitionsovertopping::z2_iter_max1](#) = 49
maximal number of iterations for calculation z2 part 1
- integer, parameter [typedefinitionsovertopping::z2_iter_max2](#) = 70
maximal number of iterations for calculation z2 part 1 & 2
- real(kind=wp), parameter [typedefinitionsovertopping::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)
innerCalculation: inner calculation for the wave runup
- real(kind=wp) function [waverunup::determinestartingvalue](#) (i, relaxationFactor, z2_start, z2_end, Hm0)
determineStartingValue: helper function to find a start value for z2
- integer function [waverunup::findsmallestresidu](#) (z2_start, z2_end, n)
findSmallestResidu: helper function to find the smallest residu
- subroutine [waverunup::convergedwithresidu](#) (z2_start, z2_end)
convergedWithResidu: helper function to handle convergence with a higher residu than expected if logging is enabled, it writes a small message to the logfile

6.11.1 Detailed Description

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

6.12 zFunctionsOvertopping.f90 File Reference

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

Modules

- module [zfunctionsovertopping](#)
Module for the Limit State Functions (Z-functions) for wave overtopping.

Functions/Subroutines

- subroutine, public [zfunctionsovertopping::calculateqorto](#) (dikeHeight, modelFactors, overtopping, load, geometry, succes, errorMessage)
Subroutine to calculate the overtopping discharge with the Overtopping dll.
- subroutine, public [zfunctionsovertopping::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 [zfunctionsovertopping::adjustprofile](#) (nrCoordinates, coordinates, dikeHeight, nrCoordsAdjusted, xCoordsAdjusted, zCoordsAdjusted, succes, errorMessage)
Subroutine adjust the profile due to a desired dike height.
- real(kind=wp) function, public [zfunctionsovertopping::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 VTV.

Index

- adjustinfluencefactors
 - formulamoduleovertopping, [19](#)
- adjustnonhorizontalberms
 - geometrymoduleovertopping, [30](#)
- adjustprofile
 - zfunctionsovertopping, [62](#)
- allocatevectorsgeometry
 - geometrymoduleovertopping, [30](#)
- basicgeometrytest
 - geometrymoduleovertopping, [32](#)
- berm_max
 - typedefinitionsovertopping, [55](#)
- berm_min
 - typedefinitionsovertopping, [55](#)
- calculateanglewaveattack
 - formulamoduleovertopping, [19](#)
- calculatebreakerlimit
 - formulamoduleovertopping, [21](#)
- calculatebreakerparameter
 - formulamoduleovertopping, [21](#)
- calculategammab
 - factormoduleovertopping, [14](#)
- calculategammabeta
 - factormoduleovertopping, [15](#)
- calculategammaf
 - factormoduleovertopping, [15](#)
- calculatehorzdistance
 - geometrymoduleovertopping, [33](#)
- calculatehorzlengths
 - geometrymoduleovertopping, [33](#)
- calculateovertopping
 - mainmoduleovertopping, [42](#)
- calculateovertoppingnegativefreeboard
 - mainmoduleovertopping, [42](#)
- calculateovertoppingsection
 - mainmoduleovertopping, [43](#)
- calculateqo
 - dllovertopping, [8](#)
- calculateqof
 - dllovertopping, [9](#)
- calculateqorto
 - zfunctionsovertopping, [62](#)
- calculatesegmentslopes
 - geometrymoduleovertopping, [34](#)
- calculatetanalpha
 - factormoduleovertopping, [17](#)
- calculatewavelength
 - formulamoduleovertopping, [22](#)
- calculatewaveovertopping
 - mainmoduleovertopping, [44](#)
- calculatewaveovertoppingdischarge
 - formulamoduleovertopping, [22](#)
- calculatewaverunup
 - formulamoduleovertopping, [23](#)
- calculatewavesteepness
 - formulamoduleovertopping, [24](#)
- calczvalue
 - dllovertopping, [10](#)
- checkcrosssection
 - geometrymoduleovertopping, [35](#)
- checkinputdata
 - mainmoduleovertopping, [45](#)
- checkmodelfactors
 - mainmoduleovertopping, [46](#)
- computedovertopping
 - typedefinitionsovertopping::tpovertoppinginput, [78](#)
- convergedwithresidu
 - waverunup, [58](#)
- copygeometry
 - geometrymoduleovertopping, [35](#)
- criticalovertopping
 - typedefinitionsovertopping::tpovertoppinginput, [78](#)
- cubicroots
 - formulamoduleovertopping, [24](#)
- currentlogging
 - modulelogging, [48](#)
- deallocategeometry
 - geometrymoduleovertopping, [36](#)
- determinesegmenttypes
 - geometrymoduleovertopping, [36](#)
- 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, [13](#)
 - versionnumber, [14](#)
- factorModuleOvertopping.f90, [82](#)
- factordeterminationq_b_f_b

- typedefinitionsovertopping::tpovertoppinginput, 78
- factordeterminationq_b_f_n
 - typedefinitionsovertopping::tpovertoppinginput, 78
- factormoduleovertopping, 14
 - calculategammab, 14
 - calculategammabeta, 15
 - calculategammamf, 15
 - calculatetanalpha, 17
- fb_max
 - typedefinitionsovertopping, 55
- fb_min
 - typedefinitionsovertopping, 55
- filename
 - modulelogging::tlogging, 71
- findsmallestresidu
 - waverunup, 59
- fn_max
 - typedefinitionsovertopping, 55
- fn_min
 - typedefinitionsovertopping, 55
- foreshore_max
 - typedefinitionsovertopping, 55
- foreshore_min
 - typedefinitionsovertopping, 55
- formulaModuleOvertopping.f90, 82
- formulamoduleovertopping, 18
 - adjustinfluencefactors, 19
 - calculateanglewaveattack, 19
 - calculatebreakerlimit, 21
 - calculatebreakerparameter, 21
 - calculatewavelength, 22
 - calculatewaveovertoppingdischarge, 22
 - calculatewaverunup, 23
 - calculatewavesteepness, 24
 - cubicroots, 24
 - isequalreal, 26
 - isequalzero, 26
 - realrootscubicfunction, 26
 - rootsdepressedcubic, 27
 - rootsgeneralcubic, 27
- frunup1
 - typedefinitionsovertopping, 55
- frunup2
 - typedefinitionsovertopping, 56
- frunup3
 - typedefinitionsovertopping, 56
- fs_max
 - typedefinitionsovertopping, 56
- fs_min
 - typedefinitionsovertopping, 56
- fshallow
 - typedefinitionsovertopping::tpovertoppinginput, 78
- geometry_c_f
 - dlovertopping, 11
- geometryModuleOvertopping.f90, 83
- geometrymoduleovertopping, 29
 - adjustnonhorizontalberms, 30
 - allocatevectorsgeometry, 30
 - basicgeometrytest, 32
 - calculatehorzdistance, 33
 - calculatehorzlengths, 33
 - calculatesegmentslopes, 34
 - checkcrosssection, 35
 - copygeometry, 35
 - deallocateggeometry, 36
 - determinesegmenttypes, 36
 - initializegeometry, 37
 - mergesquentialberms, 37
 - removeberms, 39
 - removedikesgments, 40
 - splitcrosssection, 40
- getlanguage
 - dlovertopping, 11
 - overtoppingmessages, 51
- getovertoppingformat
 - overtoppingmessages, 51
- getovertoppingmessage
 - overtoppingmessages, 52
- getovertoppingparameter
 - overtoppingmessages, 52
- h
 - typedefinitionsovertopping::tpload, 74
- hm0
 - typedefinitionsovertopping::tpload, 74
- initializegeometry
 - geometrymoduleovertopping, 37
- innercalculation
 - waverunup, 59
- interpolateresultssections
 - mainmoduleovertopping, 46
- isequalreal
 - formulamoduleovertopping, 26
- isequalzero
 - formulamoduleovertopping, 26
- iteratetogivendischarge
 - omkeervariantmodule, 48
- iteratetogivendischargevalidprofile
 - omkeervariantmodule, 49
- iterationwaverunup
 - waverunup, 60
- language
 - overtoppingmessages, 53
- m_z2
 - typedefinitionsovertopping::tpovertoppinginput, 78
- mainModuleOvertopping.f90, 85
- mainmoduleovertopping, 41
 - calculateovertopping, 42
 - calculateovertoppingnegativefreeboard, 42
 - calculateovertoppingsection, 43
 - calculatewaveovertopping, 44
 - checkinputdata, 45
 - checkmodelfactors, 46
 - interpolateresultssections, 46

- margindiff
 - typedefinitionsovertopping, 56
- maringrad
 - typedefinitionsovertopping, 56
- maxfilenamelenh
 - modulelogging, 48
- maxmsg
 - overtoppingmessages, 53
- maxpar
 - overtoppingmessages, 53
- mergesequentialberms
 - geometrymoduleovertopping, 37
- ModuleLogging.f90, 85
- modulelogging, 47
 - currentlogging, 48
 - maxfilenamelenh, 48
- modulelogging::tlogging, 70
 - filename, 71
 - verbosity, 71
- mz2_max
 - typedefinitionsovertopping, 56
- mz2_min
 - typedefinitionsovertopping, 56
- nbermssegments
 - typedefinitionsovertopping::tpgeometry, 72
- ncoordinates
 - typedefinitionsovertopping::tpgeometry, 72
- normal
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
- npoints
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
- omkeerVariantModule.f90, 86
- omkeervariantf
 - dlovertopping, 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
 - typedefinitionsovertopping::tpload, 75
- profileinstructure
 - zfunctionsovertopping, 64
- psi
 - typedefinitionsovertopping::tpgeometry, 72
- qo
 - typedefinitionsovertopping::tpovertopping, 76
- realrootscubicfunction
 - formulamoduleovertopping, 26
- reductionfactorforeshore
 - typedefinitionsovertopping::tpovertoppinginput, 78
- relaxationfactor
 - typedefinitionsovertopping::tpovertoppinginput, 78
- removeberms
 - geometrymoduleovertopping, 39
- removedikessegments
 - geometrymoduleovertopping, 40
- rfactor_max
 - typedefinitionsovertopping, 56
- rfactor_min
 - typedefinitionsovertopping, 56
- rootsdepressedcubic
 - formulamoduleovertopping, 27
- rootsgeneralcubic
 - formulamoduleovertopping, 27
- roughness
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 69
 - overtoppinginterface::tpprofilecoordinate, 79
- roughnessfactors
 - typedefinitionsovertopping::tpgeometry, 72
- segmentslopes
 - typedefinitionsovertopping::tpgeometry, 72
- segmenttypes
 - typedefinitionsovertopping::tpgeometry, 73
- setlanguage
 - dlovertopping, 12

- overtoppingmessages, 53
- slope_max
 - typedefinitionsovertopping, 57
- slope_min
 - typedefinitionsovertopping, 57
- splitcrosssection
 - geometrymoduleovertopping, 40
- tm_10
 - typedefinitionsovertopping::tload, 75
- typeDefinitionsOvertopping.f90, 87
- typedefinitionsovertopping, 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
- typedefinitionsovertopping::tpgeometry, 71
 - nbermsegs, 72
 - ncoordinates, 72
 - psi, 72
 - roughnessfactors, 72
 - segmentslopes, 72
 - segmenttypes, 73
 - xcoorddiff, 73
 - xcoordinates, 73
 - ycoorddiff, 73
 - ycoordinates, 73
- typedefinitionsovertopping::tload, 73
 - h, 74
 - hm0, 74
 - phi, 75
 - tm_10, 75
- typedefinitionsovertopping::tpovertopping, 75
 - qo, 76
 - z2, 76
- typedefinitionsovertopping::tpovertoppinginput, 76
 - computedovertopping, 78
 - criticalovertopping, 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, 13
- varmodelfactorcriticalovertopping
 - overtoppinginterface, 51
- verbosity
 - modulelogging::tlogging, 71
- versionnumber
 - dllovertopping, 14
- waveRunup.f90, 89
- waverunup, 57
 - convergedwithresidu, 58
 - determinestartingvalue, 58
 - findsmallestresidu, 59
 - innercalculation, 59
 - iterationwaverunup, 60
- xcoorddiff
 - typedefinitionsovertopping::tpgeometry, 73
- xcoordinate
 - overtoppinginterface::tpprofilecoordinate, 80
- xcoordinates
 - typedefinitionsovertopping::tpgeometry, 73
- xcoords
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 70
- xdiff_min
 - typedefinitionsovertopping, 57
- ycoorddiff
 - typedefinitionsovertopping::tpgeometry, 73
- ycoordinates
 - typedefinitionsovertopping::tpgeometry, 73
- ycoords
 - overtoppinginterface::overtoppinggeometrytype, 68
 - overtoppinginterface::overtoppinggeometrytypef, 70
- z2
 - typedefinitionsovertopping::tpovertopping, 76
- z2_iter_max1
 - typedefinitionsovertopping, 57
- z2_iter_max2
 - typedefinitionsovertopping, 57
- z2_margin
 - typedefinitionsovertopping, 57
- zFunctionsOvertopping.f90, 90

- zcoordinate
 - overtoppinginterface::tpprofilecoordinate, [80](#)
- zfunclogratios
 - zfunctionsovertopping, [65](#)
- zfunctionsovertopping, [61](#)
 - adjustprofile, [62](#)
 - calculateqorto, [62](#)
 - profileinstructure, [64](#)
 - zfunclogratios, [65](#)