

Dikes Overtopping Kernel - Technical documentation

Generated by Doxygen 1.8.9.1

Mon Sep 5 2016 11:16:50

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	31
4.4.2.4	calculatehorzdistance	32
4.4.2.5	calculatehorzlengths	32
4.4.2.6	calculatesegmentslopes	33
4.4.2.7	checkcrosssection	34
4.4.2.8	copygeometry	34
4.4.2.9	deallocategeometry	35
4.4.2.10	determinesegmenttypes	35
4.4.2.11	initializegeometry	36
4.4.2.12	mergesequentialberms	37
4.4.2.13	removeberms	38
4.4.2.14	removedikesegments	39
4.4.2.15	splitcrosssection	39
4.5	mainmoduleovertopping Module Reference	40
4.5.1	Detailed Description	41
4.5.2	Function/Subroutine Documentation	41
4.5.2.1	calculateovertopping	41
4.5.2.2	calculateovertoppingnegativefreeboard	42
4.5.2.3	calculateovertoppingsection	42
4.5.2.4	calculatewaveovertopping	43

4.5.2.5	checkinputdata	44
4.5.2.6	checkmodelfactors	45
4.5.2.7	interpolateresultssections	45
4.6	modulelogging Module Reference	46
4.6.1	Detailed Description	46
4.6.2	Variable Documentation	47
4.6.2.1	currentlogging	47
4.6.2.2	maxfilenamelength	47
4.7	omkeervariantmodule Module Reference	47
4.7.1	Detailed Description	47
4.7.2	Function/Subroutine Documentation	47
4.7.2.1	checkifonberm	47
4.7.2.2	iteratetogivendischarge	48
4.7.2.3	iteratetogivendischargevalidprofile	49
4.7.3	Variable Documentation	50
4.7.3.1	dischargeprofile	50
4.7.3.2	isberm	50
4.7.3.3	isvalidz	50
4.7.3.4	minzberm	51
4.7.3.5	toldikeheight	51
4.7.3.6	toldischarge	51
4.7.3.7	zprofile	51
4.8	overtoppinginterface Module Reference	51
4.8.1	Detailed Description	51
4.8.2	Variable Documentation	51
4.8.2.1	varmodelfactorcriticalovertopping	51
4.9	overtoppingmessages Module Reference	51
4.9.1	Detailed Description	52
4.9.2	Function/Subroutine Documentation	52
4.9.2.1	getlanguage	52
4.9.2.2	getovertoppingformat	52
4.9.2.3	getovertoppingmessage	53
4.9.2.4	getovertoppingparameter	53
4.9.2.5	setlanguage	53
4.9.3	Variable Documentation	53
4.9.3.1	language	54
4.9.3.2	maxmsg	54
4.9.3.3	maxpar	54
4.10	typedefinitionsovertopping Module Reference	54
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	56
4.10.2.5	fn_max	56
4.10.2.6	fn_min	56
4.10.2.7	foreshore_max	56
4.10.2.8	foreshore_min	56
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	57
4.10.2.15	margingrad	57
4.10.2.16	mz2_max	57
4.10.2.17	mz2_min	57
4.10.2.18	rfactor_max	57
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	58
4.10.2.24	z2_iter_max2	58
4.10.2.25	z2_margin	58
4.11	waverunup Module Reference	58
4.11.1	Detailed Description	58
4.11.2	Function/Subroutine Documentation	58
4.11.2.1	convergedwithresidu	58
4.11.2.2	determinestartingvalue	59
4.11.2.3	findsmallestresidu	59
4.11.2.4	innercalculation	60
4.11.2.5	iterationwaverunup	61
4.12	zfunctionovertopping Module Reference	62
4.12.1	Detailed Description	62
4.12.2	Function/Subroutine Documentation	63
4.12.2.1	adjustprofile	63
4.12.2.2	calculateqorto	64
4.12.2.3	profileinstructure	65

4.12.2.4	zfunclogratios	66
5	Data Type Documentation	69
5.1	overtoppinginterface::overtoppinggeometrytype Type Reference	69
5.1.1	Detailed Description	70
5.1.2	Member Data Documentation	70
5.1.2.1	normal	70
5.1.2.2	npoints	70
5.1.2.3	roughness	70
5.1.2.4	xcoords	70
5.1.2.5	ycoords	70
5.2	overtoppinginterface::overtoppinggeometrytypef Type Reference	71
5.2.1	Detailed Description	71
5.2.2	Member Data Documentation	71
5.2.2.1	normal	71
5.2.2.2	npoints	71
5.2.2.3	roughness	72
5.2.2.4	xcoords	72
5.2.2.5	ycoords	72
5.3	modulelogging::tlogging Type Reference	72
5.3.1	Detailed Description	72
5.3.2	Member Data Documentation	73
5.3.2.1	filename	73
5.3.2.2	verbosity	73
5.4	typedefinitionsovertopping::tpgeometry Type Reference	73
5.4.1	Detailed Description	74
5.4.2	Member Data Documentation	74
5.4.2.1	nbermsements	74
5.4.2.2	ncoordinates	74
5.4.2.3	psi	74
5.4.2.4	roughnessfactors	74
5.4.2.5	segmentslopes	75
5.4.2.6	segmenttypes	75
5.4.2.7	xcoorddiff	75
5.4.2.8	xcoordinates	75
5.4.2.9	ycoorddiff	75
5.4.2.10	ycoordinates	75
5.5	typedefinitionsovertopping::tpload Type Reference	75
5.5.1	Detailed Description	76
5.5.2	Member Data Documentation	76

5.5.2.1	h	76
5.5.2.2	hm0	77
5.5.2.3	phi	77
5.5.2.4	tm_10	77
5.6	typedefinitionsovertopping::tpovertopping Type Reference	77
5.6.1	Detailed Description	78
5.6.2	Member Data Documentation	78
5.6.2.1	qo	78
5.6.2.2	z2	78
5.7	typedefinitionsovertopping::tpovertoppinginput Type Reference	78
5.7.1	Detailed Description	80
5.7.2	Member Data Documentation	80
5.7.2.1	computedovertopping	80
5.7.2.2	criticalovertopping	80
5.7.2.3	factordeterminationq_b_f_b	80
5.7.2.4	factordeterminationq_b_f_n	80
5.7.2.5	fshallow	80
5.7.2.6	m_z2	80
5.7.2.7	reductionfactorforeshore	80
5.7.2.8	relaxationfactor	81
5.8	overtoppinginterface::tpprofilecoordinate Type Reference	81
5.8.1	Detailed Description	81
5.8.2	Member Data Documentation	81
5.8.2.1	roughness	82
5.8.2.2	xcoordinate	82
5.8.2.3	zcoordinate	82
6	File Documentation	83
6.1	dllOvertopping.f90 File Reference	83
6.1.1	Detailed Description	84
6.2	factorModuleOvertopping.f90 File Reference	84
6.2.1	Detailed Description	84
6.3	formulaModuleOvertopping.f90 File Reference	84
6.3.1	Detailed Description	85
6.4	geometryModuleOvertopping.f90 File Reference	85
6.4.1	Detailed Description	86
6.5	mainModuleOvertopping.f90 File Reference	87
6.5.1	Detailed Description	87
6.6	ModuleLogging.f90 File Reference	87
6.6.1	Detailed Description	88

6.7	omkeerVariantModule.f90 File Reference	88
6.7.1	Detailed Description	88
6.8	overtoppingInterface.f90 File Reference	88
6.8.1	Detailed Description	89
6.9	OvertoppingMessages.f90 File Reference	89
6.9.1	Detailed Description	90
6.10	typeDefinitionsOvertopping.f90 File Reference	90
6.10.1	Detailed Description	91
6.11	waveRunup.f90 File Reference	91
6.11.1	Detailed Description	92
6.12	zFunctionsOvertopping.f90 File Reference	92
6.12.1	Detailed Description	92
Index		93

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	40
modulelogging	Steering the extra logging	46
omkeervariantmodule	Module for the 'omkeerVariant'	47
overtoppinginterface	Module for the interface of dllOvertopping	51
overtoppingmessages	Module for the messages in the overtopping dll, in Dutch or English	51
typedefinitionsovertopping	Type definitions for Dikes Overtopping	54
waverunup	Iteration procedure for 2% wave runup	58
zfunctionsovertopping	Module for the Limit State Functions (Z-functions) for wave overtopping	62

Chapter 2

Data Type Index

2.1 Class List

Here are the data types with brief descriptions:

overtoppinginterface::overtoppinggeometrytype	69
overtoppinginterface::overtoppinggeometrytypef	71
modulelogging::tlogging	
TLogging: structure for steering the logging	72
typedefinitionsovertopping::tpgeometry	
TpGeometry: structure with geometry data	73
typedefinitionsovertopping::tpload	
TpLoad: structure with load parameters	75
typedefinitionsovertopping::tpovertopping	
TpOvertopping: structure with overtopping results	77
typedefinitionsovertopping::tpovertoppinginput	
OvertoppingModelFactors: C-structure with model factors	78
overtoppinginterface::tpprofilecoordinate	81

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:	83
factorModuleOvertopping.f90	This file contains a module with functions for the slope angle and influence factors	84
formulaModuleOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping	84
geometryModuleOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping related to the geometry	85
mainModuleOvertopping.f90	This file contains a module with the core computations for Dikes Overtopping	87
ModuleLogging.f90	Module for steering the extra logging	87
omkeerVariantModule.f90	This file contains the omkeerVariant	88
overtoppingInterface.f90	This file contains the parameters and types (structs) as part of the interface to and from dll↔ Overtopping	88
OvertoppingMessages.f90	This file contains the messages in the overtopping dll, in Dutch or English	89
typeDefinitionsOvertopping.f90	This file contains a module with the type definitions for Dikes Overtopping	90
waveRunup.f90	This file contains a module with the iteration procedure for 2% wave runup	91
zFunctionsOvertopping.f90	This file contains the limit state functions for wave overtopping within VTV	92

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*)

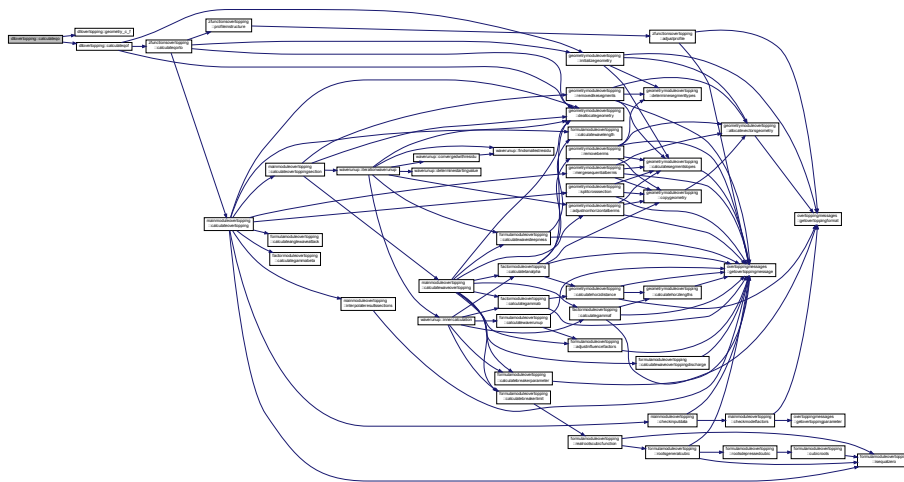
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF: convert C-like input structures to Fortran input structures.

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 65 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*)

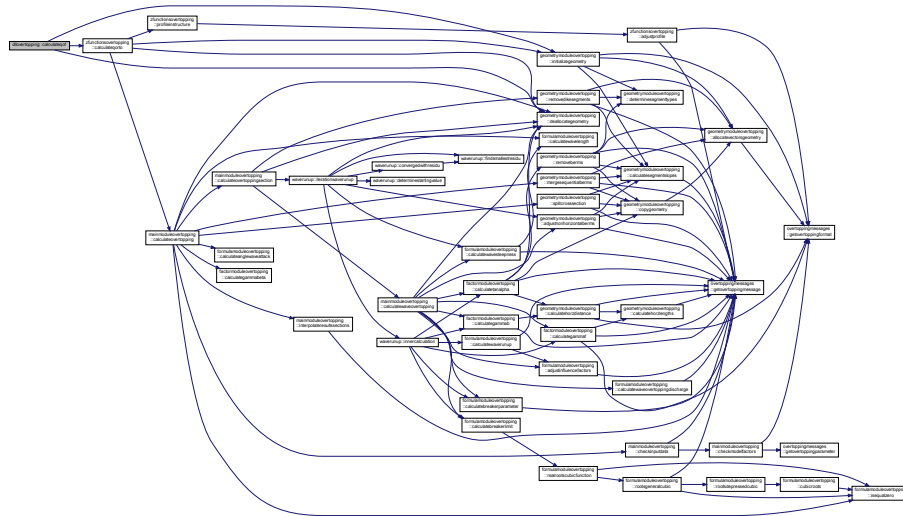
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.

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 96 of file dllovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

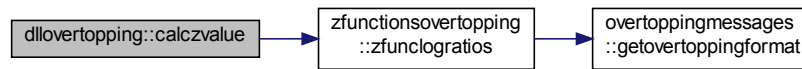
Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.

Parameters

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

Definition at line 129 of file dllovertopping.f90.

Here is the call graph for this function:



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

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

Parameters

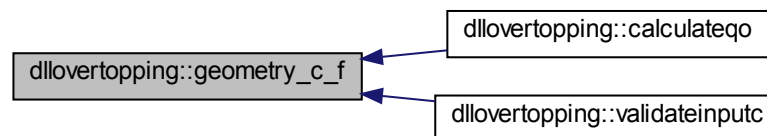
<code>in</code>	<code>geometryInput</code>	struct with geometry and roughness as c-pointers
-----------------	----------------------------	--

Returns

fortran struct with geometry and roughness

Definition at line 350 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 316 of file dllovertopping.f90.

4.1.2.6 `subroutine, public dllovertopping::omkeervariantf (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)`

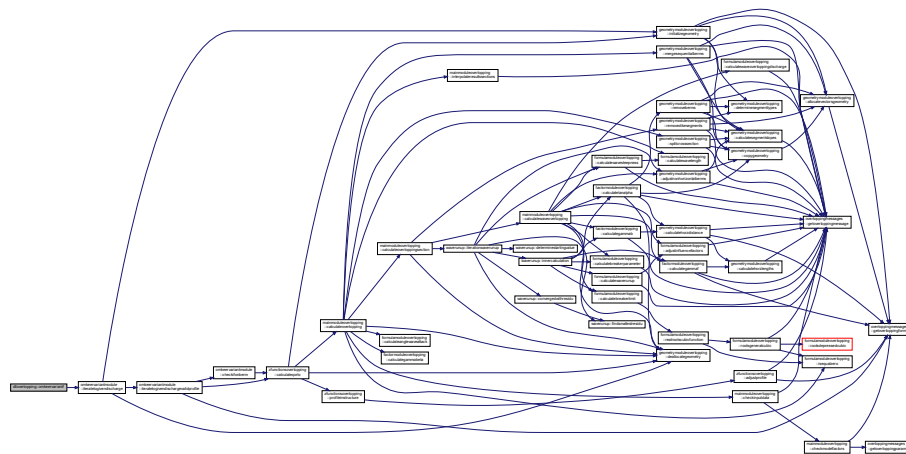
Subroutine with omkeerVariant.

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 280 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 303 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)

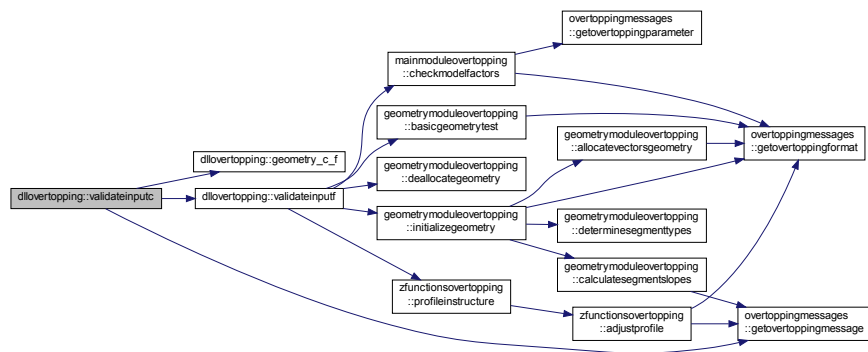
Subroutine that validates the geometry Wrapper for ValidateInputFold: convert C-like input structures to Fortran input structures.

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 149 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*)

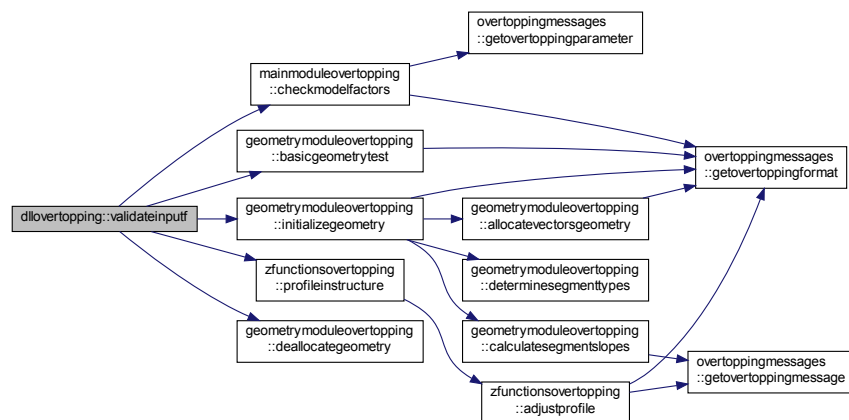
Subroutine that validates the geometry.

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 200 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 328 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*)

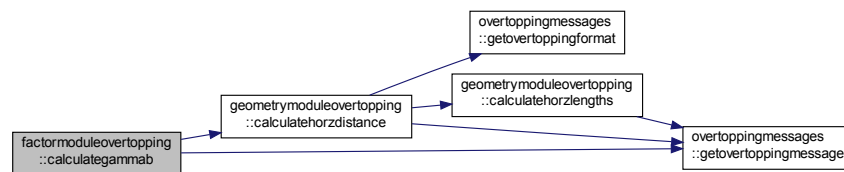
calculateGammaB influence factor berms

Parameters

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

Definition at line 310 of file factorModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.2 subroutine, public factormoduleovertopping::calculategammabeta (real(kind=wp), intent(inout) *Hm0*, real(kind=wp), intent(inout) *Tm_10*, real(kind=wp), intent(in) *beta*, real(kind=wp), intent(out) *gammaBeta_z*, real(kind=wp), intent(out) *gammaBeta_o*)

calculateGammaBeta influence factor angle of wave attack

Parameters

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

Definition at line 139 of file factorModuleOvertopping.f90.

Here is the caller graph for this function:



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*)

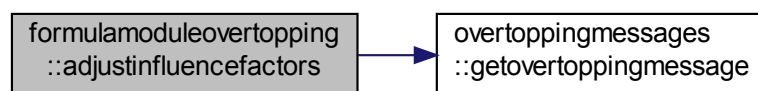
`adjustInfluenceFactors`: adjust the influence factors

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 403 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*)

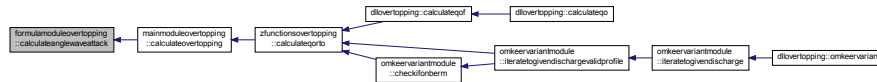
calculateAngleWaveAttack: calculate the angle of wave attack

Parameters

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

Definition at line 309 of file formulaModuleOvertopping.f90.

Here is the caller graph for this function:

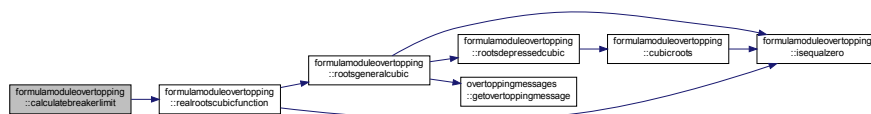


4.3.2.3 subroutine, public formulamoduleovertopping::calculatebreakerlimit (real(kind=wp), intent(in) *gammaB*, real(kind=wp), intent(out) *ksi0Limit*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

calculateBreakerLimit: calculate the breaker limit

Definition at line 332 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

calculateBreakerParameter: calculate the breaker parameter

Parameters

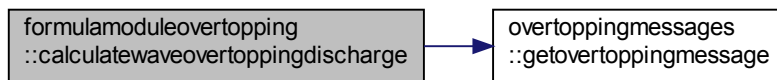
in	<i>tanalpha</i>	representative slope angle
in	<i>s0</i>	wave steepness
out	<i>ksi0</i>	breaker parameter
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 268 of file formulaModuleOvertopping.f90.

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 108 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 59 of file formulaModuleOvertopping.f90.

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

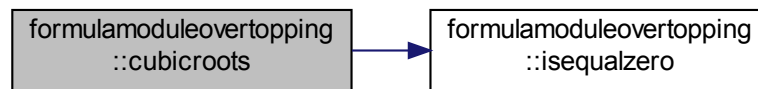
cubicRoots: calculate the roots of a cubic function

Parameters

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

Definition at line 651 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.10 logical function, public formulaModuleOvertopping::isequalreal (real(kind=wp), intent(in) *x1*, real(kind=wp), intent(in) *x2*)

isEqualReal: are two reals (almost) equal

Parameters

in	<i>x1</i>	first real
in	<i>x2</i>	second real

Definition at line 692 of file formulaModuleOvertopping.f90.

4.3.2.11 logical function, public formulaModuleOvertopping::isequalzero (real(kind=wp), intent(in) *x*)

isEqualZero: is a real (almost) zero

Parameters

in	<i>x</i>	real number
----	----------	-------------

Definition at line 716 of file formulaModuleOvertopping.f90.

Here is the caller graph for this function:



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

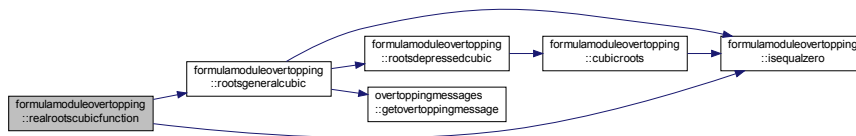
realRootsCubicFunction: calculate the roots of a cubic function

Parameters

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

Definition at line 501 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]
```

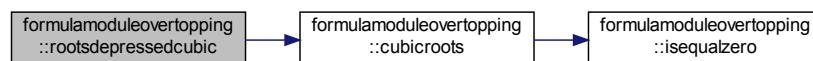
rootsDepressedCubic: calculate the roots of a depressed cubic function

Parameters

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

Definition at line 610 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]

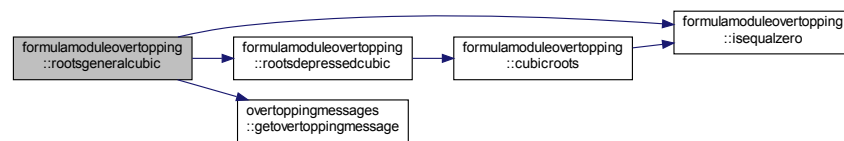
rootsGeneralCubic: calculate the roots of a generic cubic function

Parameters

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

Definition at line 557 of file formulaModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4 geometrymoduleovertopping 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 [calculateSegmentSlopes](#) (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*)

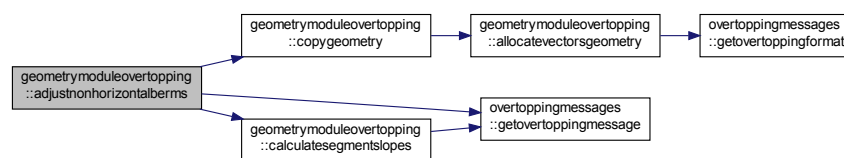
adjustNonHorizontalBerms: adjust non-horizontal berms

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 550 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) *success*, character(len=*) , intent(inout) *errorMessage*)

allocateVectorsGeometry: allocate the geometry vectors

Parameters

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

Definition at line 246 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`)

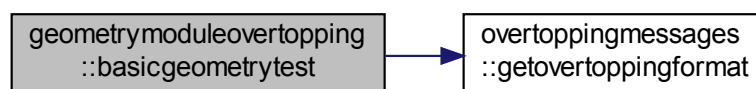
basicGeometryTest: test the input geometry (the adjusted geometry is checked elsewhere)

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 1053 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:

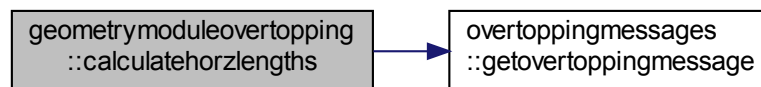


Parameters

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

Definition at line 908 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

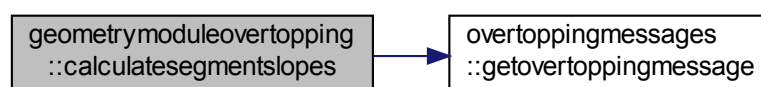
calculateSegmentSlopes: calculate the segment slopes

Parameters

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

Definition at line 308 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.7 subroutine, public geometrymoduleoverlapping::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*)

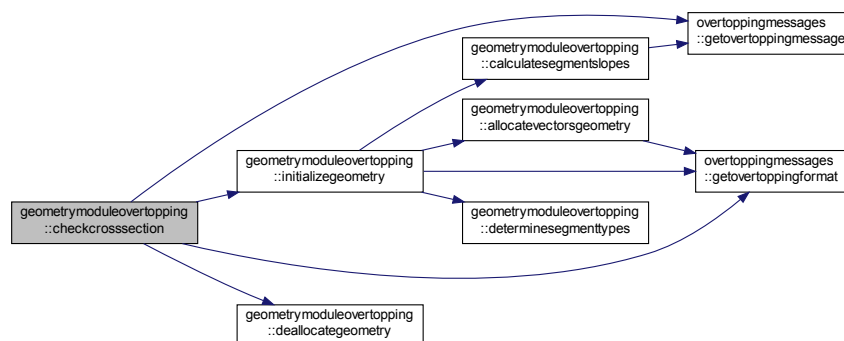
checkCrossSection: check cross section

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 59 of file geometryModuleOverlapping.f90.

Here is the call graph for this function:



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

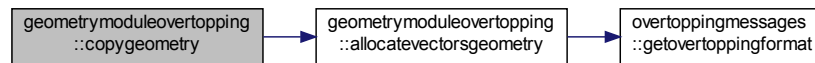
copyGeometry: copy a geometry structure

Parameters

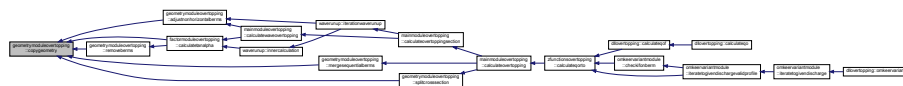
in	<i>geometry</i>	structure with geometry data
in, out	<i>geometrycopy</i>	structure with geometry data copy
out	<i>succes</i>	succes flag
in, out	<i>errormessage</i>	error message, only set in case of error

Definition at line 390 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

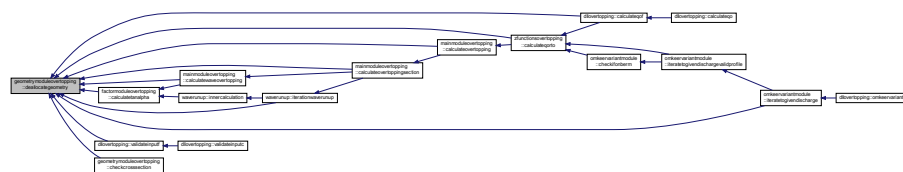
deallocateGeometry: deallocate the geometry vectors

Parameters

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

Definition at line 285 of file geometryModuleOvertopping.f90.

Here is the caller graph for this function:



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

determineSegmentTypes: determine the segment types

Parameters

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

Definition at line 347 of file geometryModuleOvertopping.f90.

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)

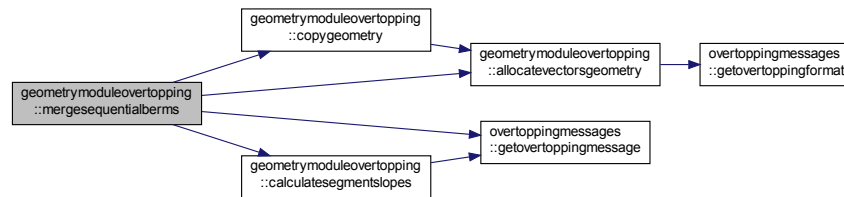
mergeSequentialBerms: merge sequential berms

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 441 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 geometrymoduleoverlapping::removeberms (type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(out) *geometryNoBerms*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

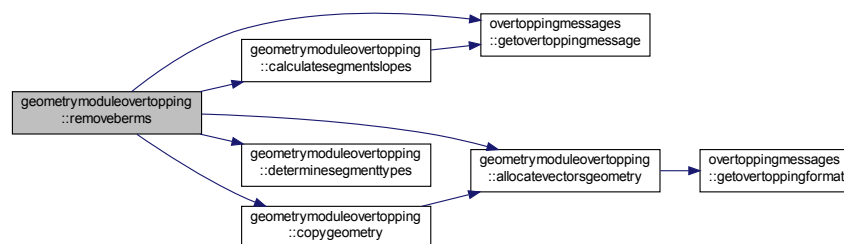
removeBerms: remove berms

Parameters

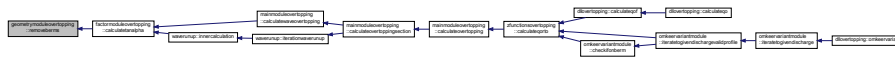
in	<i>geometry</i>	structure with geometry data
out	<i>geometrynoberms</i>	geometry data without berms
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 639 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

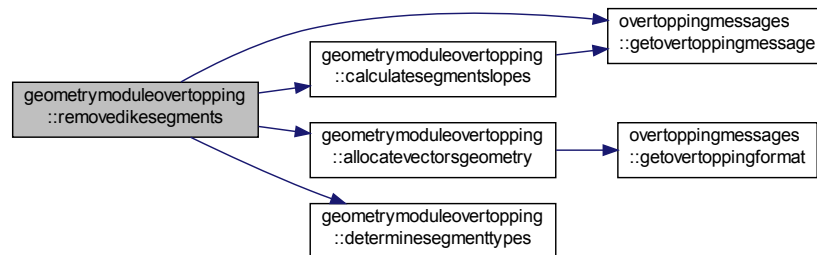
removeDikeSegments: remove dike segments

Parameters

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

Definition at line 739 of file geometryModuleOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

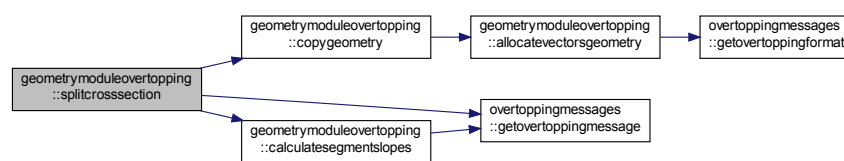
splitCrossSection: split a cross section

Parameters

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

Definition at line 803 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

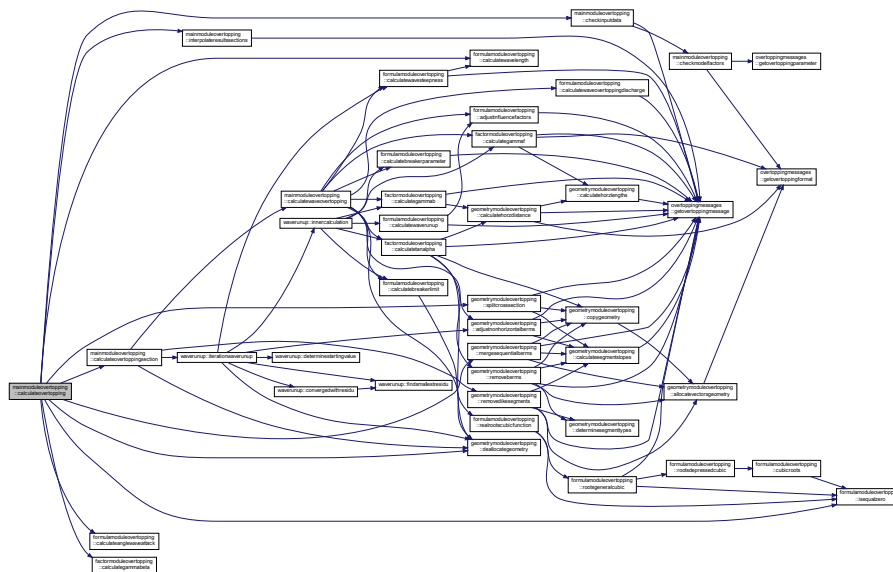
- #### 4.5.1 Detailed Description

4.5.2 Function/Subroutine Documentation

calculateOvertopping: calculate the overtopping

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

Here is the call graph for this function:



```
graph LR; dlo[diOverlapping::calculateQ] --> zfo[zfunctionOverlapping::calculateQto]; dlo --> ovr[omkevar::variantmodule::iterateChargeValidProfile]; ovr --> ovr2[omkevar::variantmodule::iterateVendicharge]; ovr2 --> dlo2[diOverlapping::omkevarVariant]; zfo --> mmo[mainModuleOverlapping::calculateOverlapping]
```

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 502 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 189 of file mainModuleOvertopping.f90.

[illegible]

calculateWaveOvertopping: calculate wave overtopping

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

Generated on Mon Sep 5 2016 11:16:50 for Dikes Overtopping Kernel - Technical documentation by Doxygen

```

graph TD
    mainmodule[mainmodule:everlopping  
:calculatewaveoverlopping]
    geometrymodule[geometrymodule:everlopping  
:adjustnonhorizontalberms]
    geometrymodule2[geometrymodule:everlopping  
:copygeometry]
    formulamodule1[formulamodule:everlopping  
:calculatewaveheight]
    geometrymodule3[geometrymodule:everlopping  
:calculatesegmenttypes]
    geometrymodule4[geometrymodule:everlopping  
:allocatevectorgeometry]
    formulamodule2[formulamodule:everlopping  
:removeberms]
    geometrymodule5[geometrymodule:everlopping  
:determinesegmenttypes]
    formulamodule3[formulamodule:everlopping  
:calculatewavevelocity]
    formulamodule4[formulamodule:everlopping  
:calculatebeta]
    geometrymodule6[geometrymodule:everlopping  
:deallocategeometry]
    formulamodule5[formulamodule:everlopping  
:calculatewaveoverloppingdischarge]
    formulamodule6[formulamodule:everlopping  
:calculatebreakerparameter]
    overloppingmessages1[overloppingmessages  
:getoverloppingformat]
    formulamodule7[formulamodule:everlopping  
:adjustinfluencefactors]
    formulamodule8[formulamodule:everlopping  
:calculategammaab]
    geometrymodule7[geometrymodule:everlopping  
:calculatehorzdistance]
    geometrymodule8[geometrymodule:everlopping  
:calculatehorzlengths]
    formulamodule9[formulamodule:everlopping  
:calculategammaef]
    formulamodule10[formulamodule:everlopping  
:calculatebreakerlimit]
    formulamodule11[formulamodule:everlopping  
:realrootcubicfunction]
    formulamodule12[formulamodule:everlopping  
:rootgeneralcubic]
    formulamodule13[formulamodule:everlopping  
:rootdepressedcubic]
    formulamodule14[formulamodule:everlopping  
:cubicroots]
    formulamodule15[formulamodule:everlopping  
:isquagzero]
    overloppingmessages2[overloppingmessages  
:getoverloppingmessage]

    mainmodule --> geometrymodule
    mainmodule --> geometrymodule2
    mainmodule --> formulamodule1
    mainmodule --> geometrymodule3
    mainmodule --> geometrymodule4
    mainmodule --> formulamodule2
    mainmodule --> geometrymodule5
    mainmodule --> formulamodule3
    mainmodule --> formulamodule4
    mainmodule --> geometrymodule6
    mainmodule --> formulamodule5
    mainmodule --> formulamodule6
    mainmodule --> overloppingmessages1
    mainmodule --> formulamodule7
    mainmodule --> formulamodule8
    mainmodule --> geometrymodule7
    mainmodule --> geometrymodule8
    mainmodule --> formulamodule9
    mainmodule --> formulamodule10
    mainmodule --> formulamodule11
    mainmodule --> formulamodule12
    mainmodule --> formulamodule13
    mainmodule --> formulamodule14
    mainmodule --> formulamodule15
    mainmodule --> overloppingmessages2

    geometrymodule --> formulamodule1
    geometrymodule2 --> formulamodule1
    geometrymodule3 --> formulamodule3
    geometrymodule4 --> formulamodule3
    geometrymodule5 --> formulamodule3
    geometrymodule6 --> formulamodule4
    geometrymodule7 --> formulamodule7
    geometrymodule8 --> formulamodule8
    formulamodule1 --> formulamodule2
    formulamodule2 --> formulamodule3
    formulamodule3 --> formulamodule4
    formulamodule4 --> formulamodule5
    formulamodule5 --> formulamodule6
    formulamodule6 --> formulamodule7
    formulamodule7 --> formulamodule8
    formulamodule8 --> formulamodule9
    formulamodule9 --> formulamodule10
    formulamodule10 --> formulamodule11
    formulamodule11 --> formulamodule12
    formulamodule12 --> formulamodule13
    formulamodule13 --> formulamodule14
    formulamodule14 --> formulamodule15
    overloppingmessages1 --> overloppingmessages2
    overloppingmessages2 --> formulamodule15
  
```

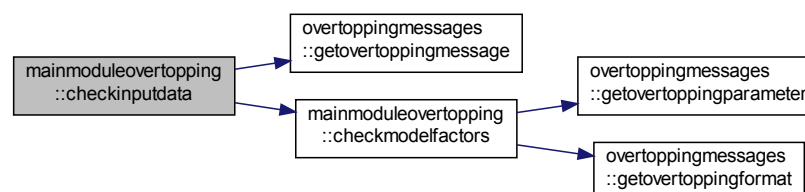
```

graph LR
    A[mainmodule:ontwikopping] --> B[mainmodule:ontwikopping:calculatewervingspectra]
    B --> C[mainmodule:ontwikopping:calculatewervingspectra]
    C --> D[functions:ontwikopping:calculateprob]
    D --> E[functions:ontwikopping:calculateprob]
    E --> F[diwertopping:calculateprob]
    E --> G[diwertopping:calculateprob]
    F --> H[onkier:variantmodule:iteratingwervingspectra]
    G --> H
    H --> I[onkier:variantmodule:iteratingwervingspectra]
    I --> J[onkier:variantmodule:iteratingwervingspectra]
    J --> K[diwertopping:ontkier:variantmodule:iteratingwervingspectra]
    K --> L[diwertopping:ontkier:variantmodule:iteratingwervingspectra]
  
```

checkInputdata: check the input data

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

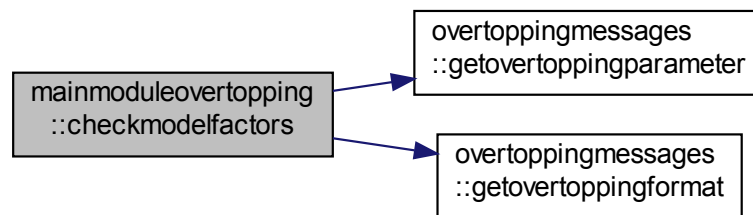
checkModelFactors: check the input data

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 681 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*)

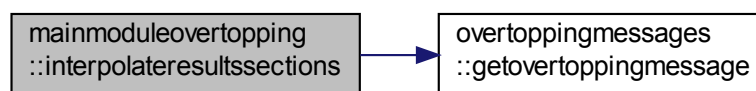
interpolateResultsSections: interpolate results for split cross sections

Parameters

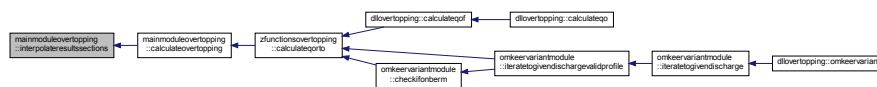
in	<i>geometry</i>	structure with geometry data
in	<i>l0</i>	wave length (m)
in	<i>nwideberms</i>	number of wide berms
in	<i>overtoppingb</i>	structure with overtopping results ordinary berms
in	<i>overtoppingf</i>	structure with overtopping results foreshores
out	<i>overtopping</i>	structure with combined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

Definition at line 538 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 43 of file ModuleLogging.f90.

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

maximum length of filename

Definition at line 35 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.
- subroutine `checkifonberm` (geometry, load, modelFactors, overtopping, givenDischarge, dikeHeight, iUp, iLow, dis1, dis2, minDikeHeight, maxDikeHeight, foundValue, success, errorText)

Variables

- real(kind=wp), parameter `toldischarge` = 1d-3
- real(kind=wp), parameter `toldikeheight` = 1d-3
- real(kind=wp), parameter `minzberm` = 0.1_wp
- logical, dimension(:), allocatable `isberm`
- logical, dimension(:), allocatable `isvalidz`
- real(kind=wp), dimension(:), allocatable `dischargeprofile`
- real(kind=wp), dimension(:), allocatable `zprofile`

4.7.1 Detailed Description

Module for the 'omkeerVariant'.

4.7.2 Function/Subroutine Documentation

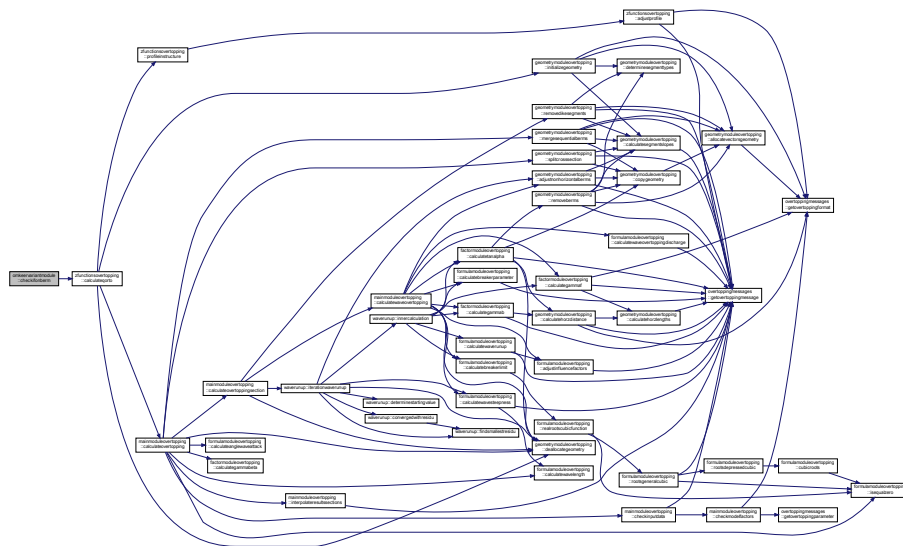
- 4.7.2.1 subroutine `omkeervariantmodule::checkifonberm` (type(tpgeometry), intent(in) *geometry*, type(tpload), intent(in) *load*, type(tpovertoppinginput), intent(inout) *modelFactors*, type(tpovertopping), intent(inout) *overtopping*, real(kind=wp), intent(in) *givenDischarge*, real(kind=wp), intent(inout) *dikeHeight*, integer, intent(in) *iUp*, integer, intent(in) *iLow*, real(kind=wp), intent(inout) *dis1*, real(kind=wp), intent(inout) *dis2*, real(kind=wp), intent(inout) *minDikeHeight*, real(kind=wp), intent(inout) *maxDikeHeight*, logical, intent(inout) *foundValue*, logical, intent(inout) *success*, character(len=*), intent(inout) *errorText*)

Parameters

in	<i>geometry</i>	internal structure with geometry data
in	<i>load</i>	struct with waterlevel and wave parameters
in, out	<i>modelfactors</i>	struct with modelFactors
in, out	<i>overtopping</i>	structure with overtopping results
in	<i>iup</i>	upper bound on profile
in	<i>ilow</i>	lower bound on profile
in	<i>givendischarge</i>	discharge to iterate to
in, out	<i>dis1</i>	discharge at minDikeHeight
in, out	<i>dis2</i>	discharge at maxDikeHeight
in, out	<i>mindikeheight</i>	lower bound dike heigth
in, out	<i>maxdikeheight</i>	upper bound dike heigth
in, out	<i>dikeheight</i>	dike height
in, out	<i>foundvalue</i>	flag for early succesfull return
in, out	<i>success</i>	flag for success
in, out	<i>errortext</i>	error message (only set if not successful)

Definition at line 245 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, 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*)

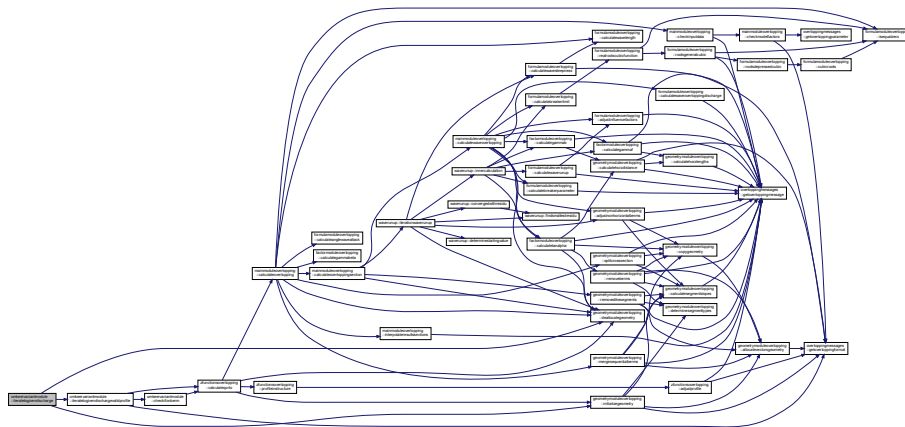
Subroutine with omkeerVariant.

Parameters

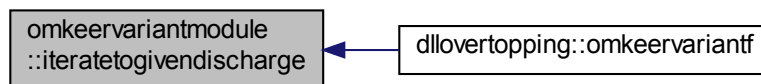
in	<i>geometryf</i>	struct with geometry and roughness
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>givedischarge</i>	discharge to iterate to
out	<i>dikeheight</i>	dike height
in,out	<i>modelfactors</i>	struct with modelFactors
in,out	<i>overtopping</i>	structure with overtopping results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

Definition at line 61 of file omkeerVariantModule.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



```
4.7.2.3 subroutine omkeervariantmodule::iteratetogivedischargevalidprofile ( 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=*) errorText ) [private]
```

Subroutine with iterateToGivenDischarge, with already checked profile.

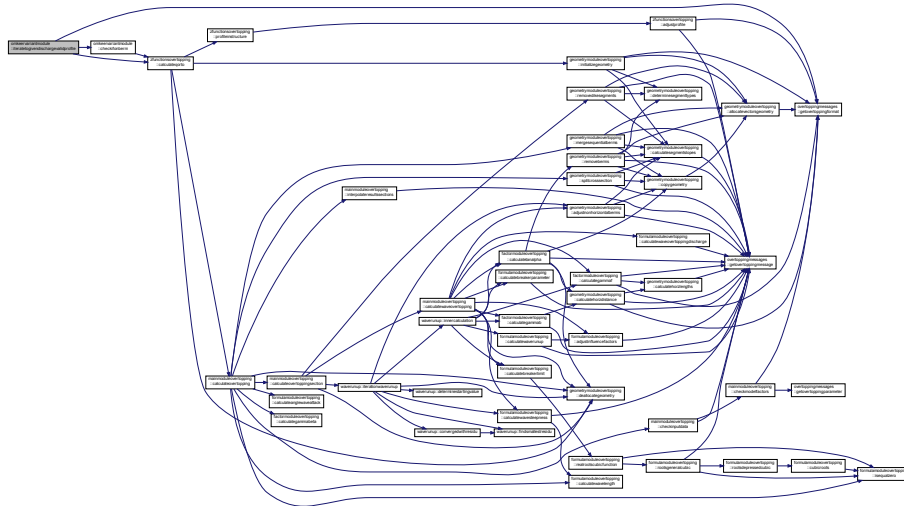
Parameters

in	<i>geometry</i>	internal structure with geometry data
----	-----------------	---------------------------------------

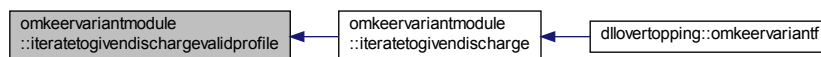
in	<i>load</i>	struct with waterlevel and wave parameters
in	<i>givendischarge</i>	discharge to iterate to
out	<i>dikeheight</i>	dike height
in,out	<i>modelfactors</i>	struct with modelFactors
in,out	<i>overtopping</i>	structure with overtopping results
out	<i>success</i>	flag for success
out	<i>errortext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

Definition at line 102 of file omkeerVariantModule.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3 Variable Documentation

4.7.3.1 `real(kind=wp), dimension(:), allocatable omkeervariantmodule::dischargeprofile`

Definition at line 50 of file omkeerVariantModule.f90.

4.7.3.2 `logical, dimension(:), allocatable omkeervariantmodule::isberm`

Definition at line 48 of file omkeerVariantModule.f90.

4.7.3.3 `logical, dimension(:), allocatable omkeervariantmodule::isvalidz`

Definition at line 49 of file omkeerVariantModule.f90.

4.7.3.4 `real(kind=wp), parameter omkeervariantmodule::minzberm = 0.1_wp`

Definition at line 46 of file omkeerVariantModule.f90.

4.7.3.5 `real(kind=wp), parameter omkeervariantmodule::toldikeheight = 1d-3`

Definition at line 45 of file omkeerVariantModule.f90.

4.7.3.6 `real(kind=wp), parameter omkeervariantmodule::toldischarge = 1d-3`

Definition at line 44 of file omkeerVariantModule.f90.

4.7.3.7 `real(kind=wp), dimension(:), allocatable omkeervariantmodule::zprofile`

Definition at line 51 of file omkeerVariantModule.f90.

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 38 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	<i>lang</i>	filled with current language ID
-----	-------------	---------------------------------

Definition at line 121 of file `OvertoppingMessages.f90`.

4.9.2.2 character(len=`maxmsg`) function `overtoppingmessages::getovertoppingformat` (integer, intent(in) *ID*)

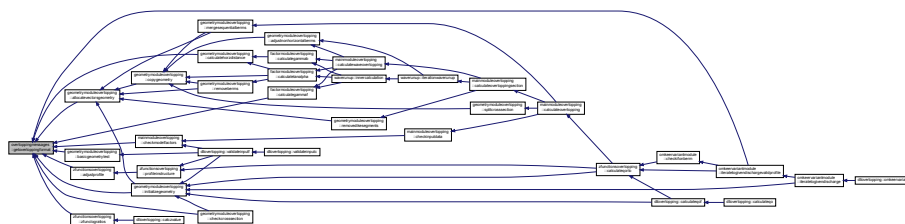
Subroutine that returns a Fortran format string with the corresponding ID in the current language.

Parameters

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

Definition at line 281 of file `OvertoppingMessages.f90`.

Here is the caller graph for this function:



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

default : Dutch

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

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

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

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

Definition at line 36 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 margindiff = 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 margingrad = 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 91 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 90 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 100 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 99 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 102 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 101 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 106 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 105 of file typeDefinitionsOvertopping.f90.

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

Definition at line 63 of file typeDefinitionsOvertopping.f90.

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

Definition at line 64 of file typeDefinitionsOvertopping.f90.

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

Definition at line 65 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 104 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 103 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 89 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 94 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 98 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 97 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 96 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 95 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 93 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 92 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 88 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 107 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 108 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 109 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	z2_start	array with z2 values at the start of the iteration i
in, out	z2_end	array with z2 values at the end of iteration i

Definition at line 373 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 292 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 330 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 (tpovertoppinginput), 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

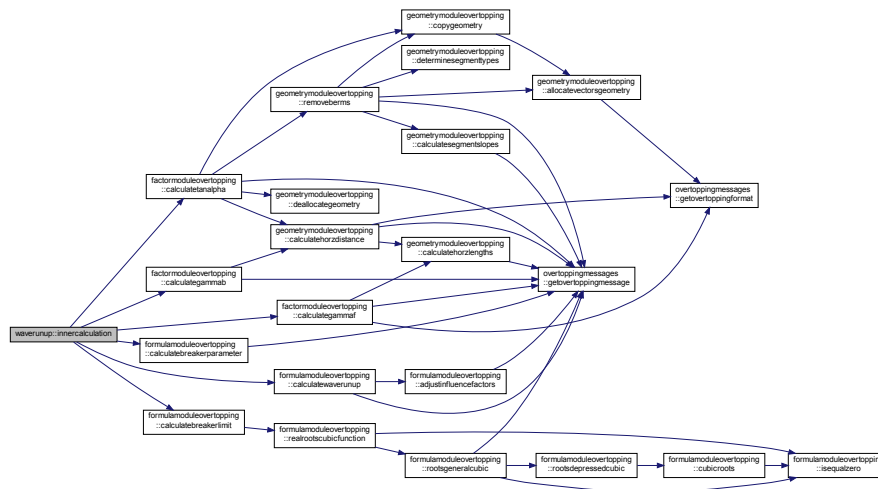
in	<i>geometry</i>	structure with geometry data
in	<i>h</i>	local water level (m+NAP)
in	<i>hm0</i>	significant wave height (m)
in, out	<i>gammabeta_z</i>	influence factor angle wave attack 2% run-up
in	<i>modelfactors</i>	structure with model factors
in	<i>z2</i>	2% wave run-up (m)
in	<i>s0</i>	wave 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

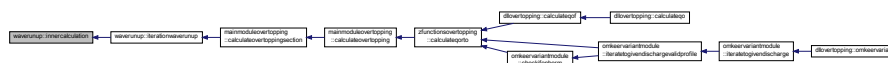
2% wave run-up at end of inner calculation

Definition at line 181 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 (tpovertoppinginput), intent(in) *modelFactors*, real(kind=wp), intent(out) *z2*, logical, intent(out) *success*, 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 59 of file waveRunup.f90.

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]

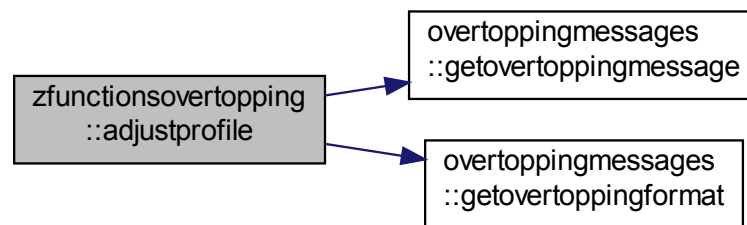
Subroutine adjust the profile due to a desired dike height.

Parameters

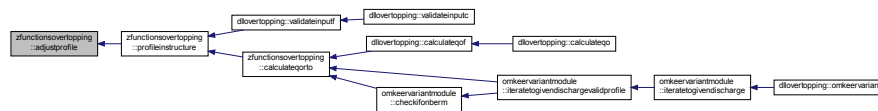
in	<i>nrcoordinates</i>	number of coordinates of the profile
in	<i>coordinates</i>	structure for the profile
in	<i>dikeheight</i>	dike height
out	<i>nrcoordsadjusted</i>	number of coordinates in the adjusted profile
	<i>xcoordsadjusted</i>	vector with x-coordinates of the adjusted profile
	<i>zcoordsadjusted</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 129 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.

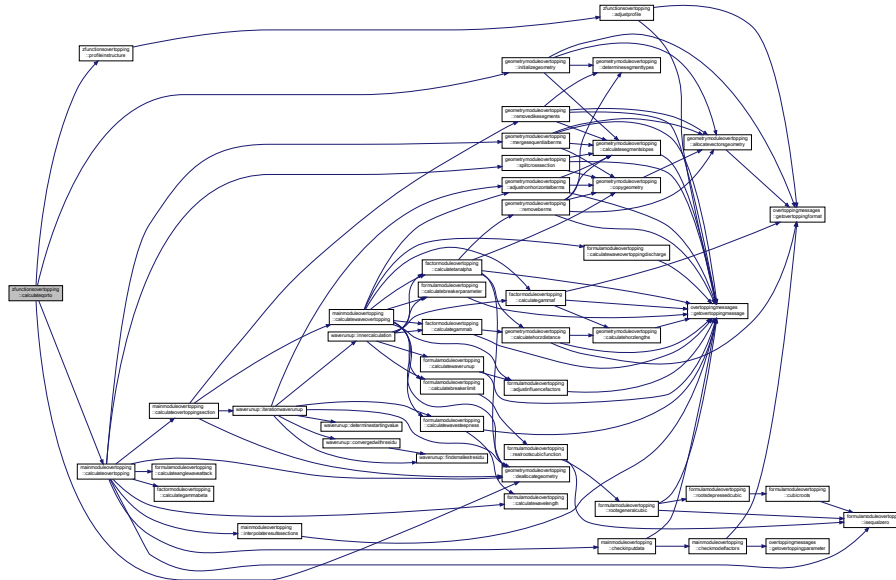
Parameters

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

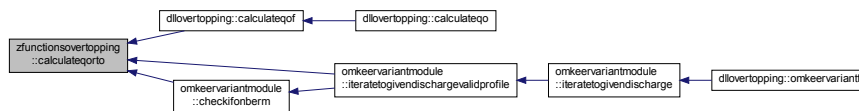
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 55 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

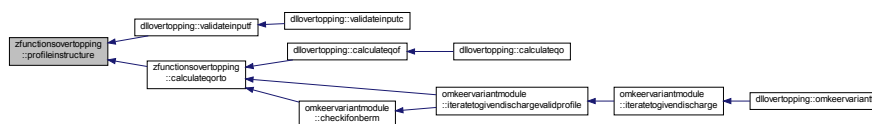
	<i>xcoordsadjusted</i>	vector with x-coordinates of the adjusted profile
	<i>zcoordsadjusted</i>	vector with y-coordinates of the adjusted profile
out	<i>succes</i>	flag for succes
out	<i>errormessage</i>	error message

Definition at line 104 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) mgo, real (kind=wp), intent(in) mqc, logical, intent(out) success, character(len=*), intent(out) errorMessage)`

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

Parameters

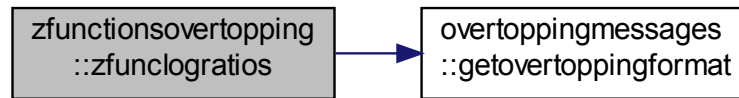
in	<i>qo</i>	computed overtopping discharge
in	<i>qc</i>	Critical overtopping discharge
in	<i>mgo</i>	Model factor computed overtopping discharge
in	<i>mqc</i>	Model factor Critical overtopping discharge
out	<i>success</i>	Flag for succes
out	<i>errormessage</i>	error message, only set if not successful

Returns

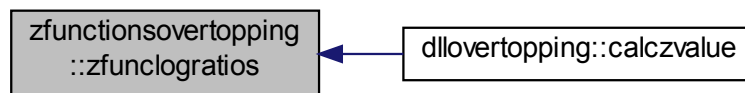
Value z-function

Definition at line 233 of file zFunctionsOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

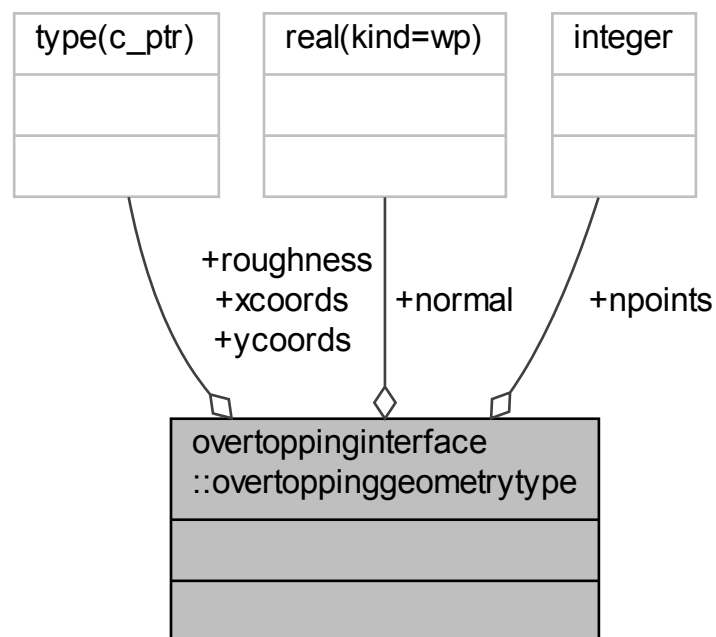


Chapter 5

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 46 of file overtoppingInterface.f90.

5.1.2 Member Data Documentation

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

Definition at line 47 of file overtoppingInterface.f90.

5.1.2.2 `integer overtoppinginterface::overtoppinggeometrytype::npoints`

Definition at line 48 of file overtoppingInterface.f90.

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

Definition at line 51 of file overtoppingInterface.f90.

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

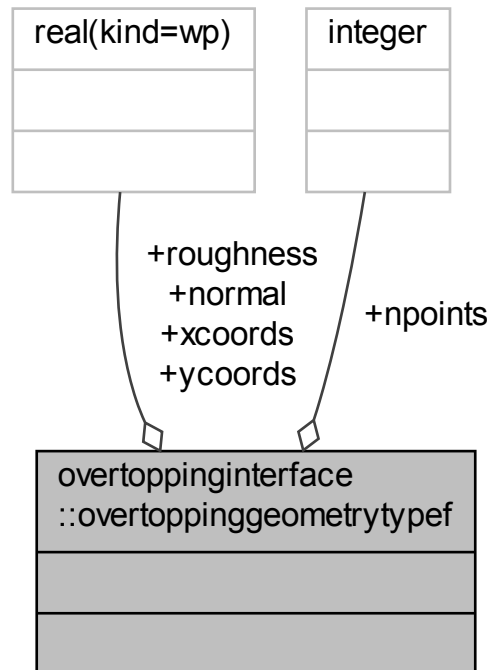
Definition at line 49 of file overtoppingInterface.f90.

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

Definition at line 50 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 54 of file `overtoppingInterface.f90`.

5.2.2 Member Data Documentation

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

Definition at line 55 of file `overtoppingInterface.f90`.

5.2.2.2 `integer` `overtoppinginterface::overtoppinggeometrytypef::npoints`

Definition at line 56 of file `overtoppingInterface.f90`.

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

Definition at line 59 of file `overtoppingInterface.f90`.

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

Definition at line 57 of file `overtoppingInterface.f90`.

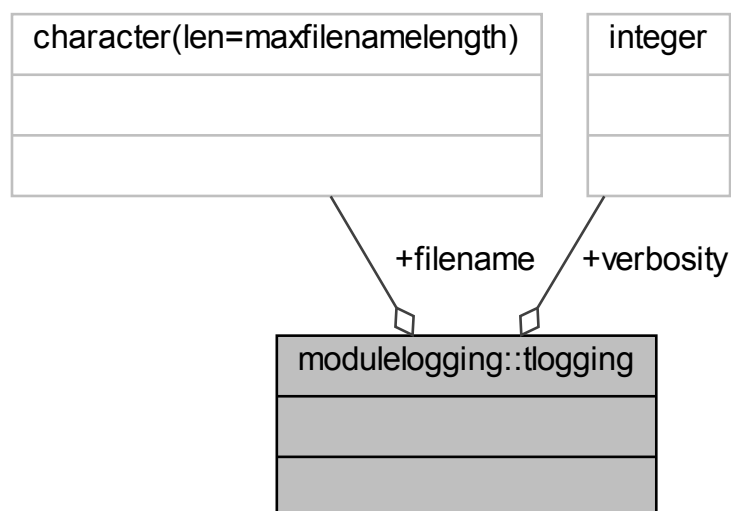
5.2.2.5 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::ycoords`

Definition at line 58 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 38 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 40 of file ModuleLogging.f90.

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

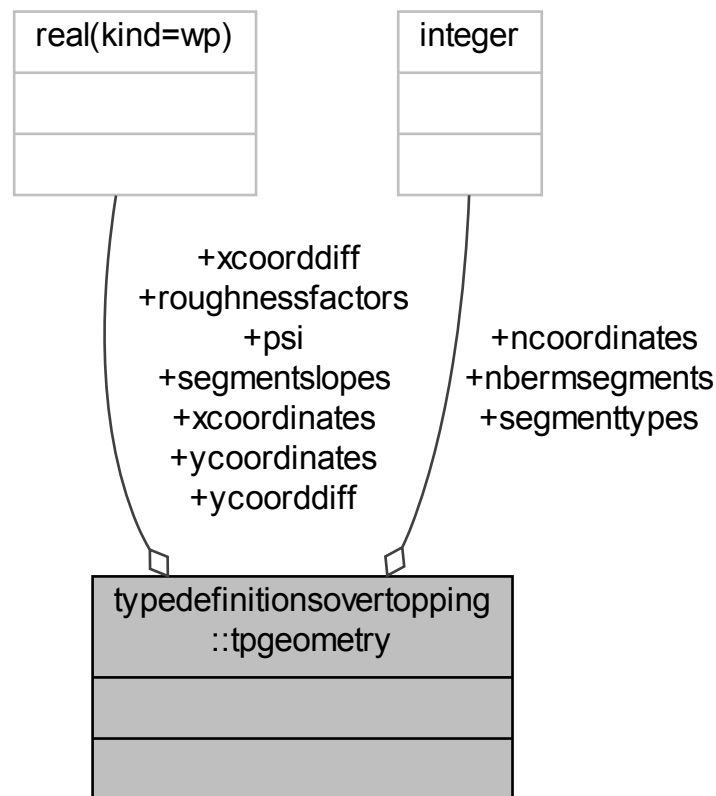
level of verbosity: one of verboseNone, verboseBasic, verboseDetailed, verboseDebugging

Definition at line 39 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 42 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 52 of file typeDefinitionsOvertopping.f90.

5.4.2.2 integer typedefinitionsovertopping::tpgeometry::ncoordinates

number of coordinates cross section

Definition at line 44 of file typeDefinitionsOvertopping.f90.

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

dike normal (degrees)

Definition at line 43 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 47 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 50 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 51 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 48 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 45 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 49 of file typeDefinitionsOvertopping.f90.

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

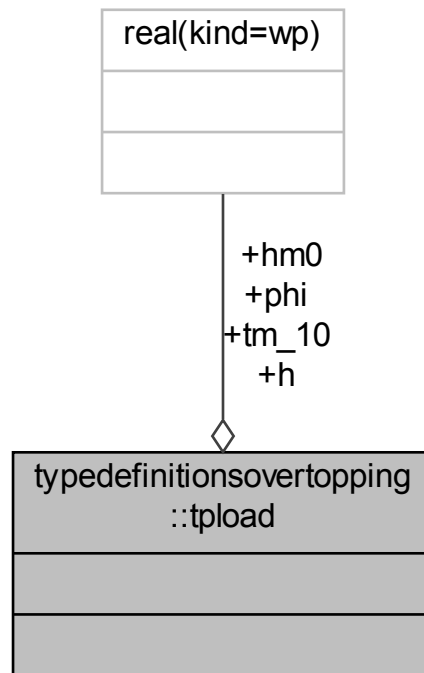
vector with y-coordinates cross section (m+NAP)

Definition at line 46 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 56 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 57 of file `typeDefinitionsOvertopping.f90`.

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

significant wave height (m)

Definition at line 58 of file typeDefinitionsOvertopping.f90.

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

wave direction (degrees)

Definition at line 60 of file typeDefinitionsOvertopping.f90.

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

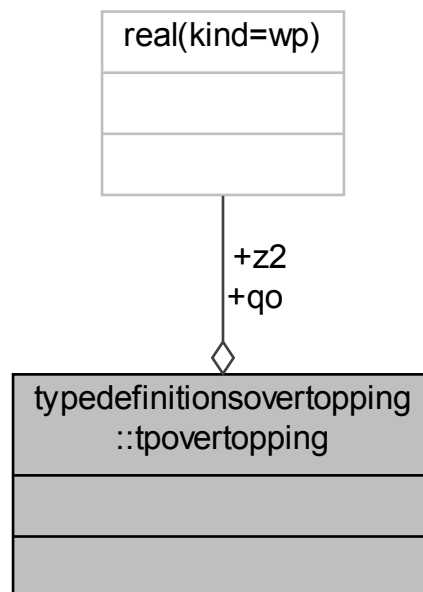
spectral wave period (s)

Definition at line 59 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 80 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 82 of file typeDefinitionsOvertopping.f90.

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

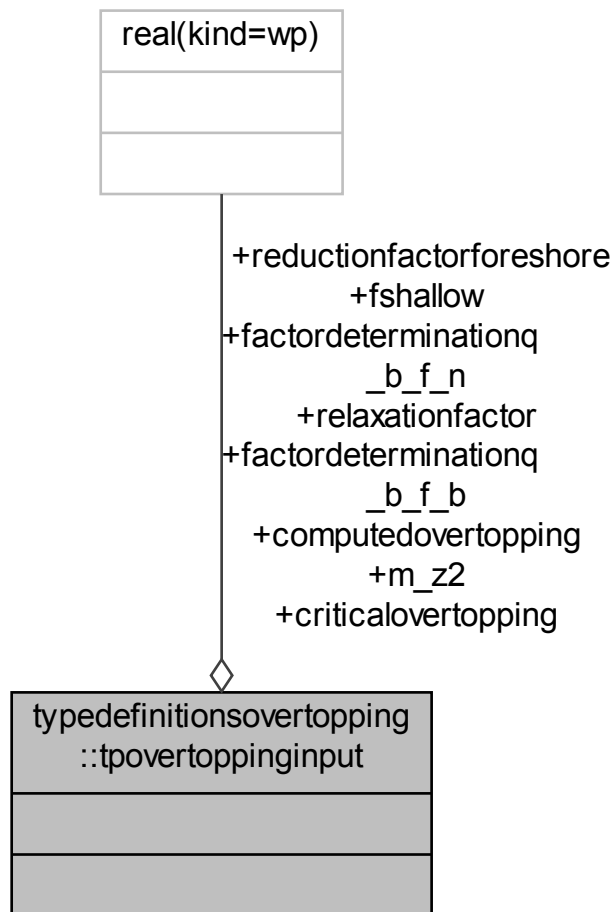
2% wave run-up (m)

Definition at line 81 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 68 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 73 of file typeDefinitionsOvertopping.f90.

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

model factor critical overtopping

Definition at line 74 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 70 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 69 of file typeDefinitionsOvertopping.f90.

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

model factor for shallow waves

Definition at line 72 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 71 of file typeDefinitionsOvertopping.f90.

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

reduction factor foreshore

Definition at line 76 of file typeDefinitionsOvertopping.f90.

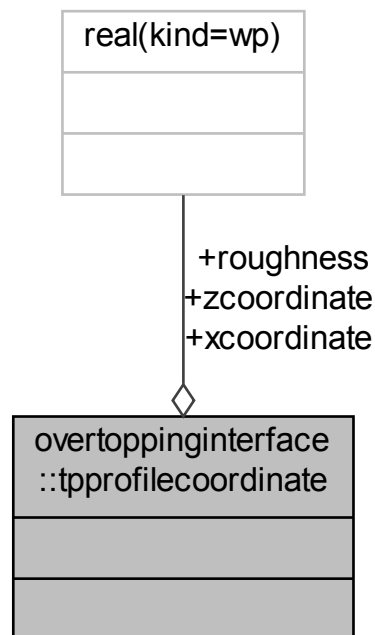
5.7.2.8 `real(kind=wp) typedefinitionsovertopping::tpovertoppinginput::relaxationfactor`

relaxation factor iteration procedure wave runup

Definition at line 75 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 40 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 43 of file `overtoppingInterface.f90`.

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

X-coordinate foreland profile.

Definition at line 41 of file `overtoppingInterface.f90`.

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

Z-coordinate foreland profile.

Definition at line 42 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, B, geometrysectionF, 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::maxfilenamelen` = 256
maximum length of filename
- type(tlogging) `modulelogging::currentlogging`
copy of argument logging

6.6.1 Detailed Description

Module for steering the extra logging.

6.7 omkeerVariantModule.f90 File Reference

This file contains the omkeerVariant.

Modules

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

Functions/Subroutines

- subroutine, public `omkeervariantmodule::iteratetogivendischarge` (load, geometryF, givenDischarge, dikeHeight, modelFactors, overtopping, success, errorText, logging)
Subroutine with omkeerVariant.
- subroutine `omkeervariantmodule::iteratetogivendischargevalidprofile` (load, geometry, givenDischarge, dikeHeight, modelFactors, overtopping, success, errorText)
Subroutine with iterateToGivenDischarge, with already checked profile.
- subroutine `omkeervariantmodule::checkifonberm` (geometry, load, modelFactors, overtopping, givenDischarge, dikeHeight, iUp, iLow, dis1, dis2, minDikeHeight, maxDikeHeight, foundValue, success, errorText)

Variables

- real(kind=wp), parameter `omkeervariantmodule::toldischarge` = 1d-3
- real(kind=wp), parameter `omkeervariantmodule::toldikeheight` = 1d-3
- real(kind=wp), parameter `omkeervariantmodule::minzberm` = 0.1_wp
- logical, dimension(:), allocatable `omkeervariantmodule::isberm`
- logical, dimension(:), allocatable `omkeervariantmodule::isvalidz`
- real(kind=wp), dimension(:), allocatable `omkeervariantmodule::dischargeprofile`
- real(kind=wp), dimension(:), allocatable `omkeervariantmodule::zprofile`

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, [63](#)
- allocatevectorsgeometry
 - geometrymoduleovertopping, [30](#)
- basicgeometrytest
 - geometrymoduleovertopping, [31](#)
- berm_max
 - typedefinitionsovertopping, [55](#)
- berm_min
 - typedefinitionsovertopping, [55](#)
- calculateanglewaveattack
 - formulamoduleovertopping, [19](#)
- calculatebreakerlimit
 - formulamoduleovertopping, [21](#)
- calculatebreakerparameter
 - formulamoduleovertopping, [21](#)
- calculategammab
 - factormoduleovertopping, [14](#)
- calculategammabeta
 - factormoduleovertopping, [15](#)
- calculategammalf
 - factormoduleovertopping, [15](#)
- calculatehorzdistance
 - geometrymoduleovertopping, [32](#)
- calculatehorzlengths
 - geometrymoduleovertopping, [32](#)
- calculateovertopping
 - mainmoduleovertopping, [41](#)
- calculateovertoppingnegativefreeboard
 - mainmoduleovertopping, [41](#)
- calculateovertoppingsection
 - mainmoduleovertopping, [42](#)
- calculateqo
 - dlllovertopping, [8](#)
- calculateqof
 - dlllovertopping, [9](#)
- calculateqorto
 - zfunctionsovertopping, [64](#)
- calculatesegmentslopes
 - geometrymoduleovertopping, [33](#)
- calculatetanalpha
 - factormoduleovertopping, [17](#)
- calculatewavelength
 - formulamoduleovertopping, [22](#)
- calculatewaveovertopping
 - mainmoduleovertopping, [43](#)
- calculatewaveovertoppingdischarge
 - formulamoduleovertopping, [22](#)
- calculatewaverunup
 - formulamoduleovertopping, [23](#)
- calculatewavesteepness
 - formulamoduleovertopping, [24](#)
- calczvalue
 - dlllovertopping, [10](#)
- checkcrosssection
 - geometrymoduleovertopping, [34](#)
- checkifonberm
 - omkeervariantmodule, [47](#)
- checkinputdata
 - mainmoduleovertopping, [44](#)
- checkmodelfactors
 - mainmoduleovertopping, [45](#)
- computedovertopping
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- convergedwithresidu
 - waverunup, [58](#)
- copygeometry
 - geometrymoduleovertopping, [34](#)
- criticalovertopping
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- cubicroots
 - formulamoduleovertopping, [24](#)
- currentlogging
 - modulelogging, [47](#)
- deallocateggeometry
 - geometrymoduleovertopping, [35](#)
- determinesegmenttypes
 - geometrymoduleovertopping, [35](#)
- determinestartingvalue
 - waverunup, [59](#)
- dischargeprofile
 - omkeervariantmodule, [50](#)
- dllOvertopping.f90, [83](#)
- dlllovertopping, [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, [84](#)
- factordeterminationq_b_f_b
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- factordeterminationq_b_f_n
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- factormoduleovertopping, [14](#)
 - calculategammab, [14](#)
 - calculategammabeta, [15](#)
 - calculategammaf, [15](#)
 - calculatetanalpha, [17](#)
- fb_max
 - typedefinitionsovertopping, [55](#)
- fb_min
 - typedefinitionsovertopping, [55](#)
- filename
 - modulelogging::tlogging, [73](#)
- findsmallestresidu
 - waverunup, [59](#)
- fn_max
 - typedefinitionsovertopping, [56](#)
- fn_min
 - typedefinitionsovertopping, [56](#)
- foreshore_max
 - typedefinitionsovertopping, [56](#)
- foreshore_min
 - typedefinitionsovertopping, [56](#)
- formulaModuleOvertopping.f90, [84](#)
- 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, [56](#)
- frunup2
 - typedefinitionsovertopping, [56](#)
- frunup3
 - typedefinitionsovertopping, [56](#)
- fs_max
 - typedefinitionsovertopping, [56](#)
- fs_min
 - typedefinitionsovertopping, [56](#)
- fshallow
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- geometry_c_f
 - dllovertopping, [11](#)
- geometryModuleOvertopping.f90, [85](#)
- geometrymoduleovertopping, [29](#)
 - adjustnonhorizontalberms, [30](#)
 - allocatevectorsgeometry, [30](#)
 - basicgeometrytest, [31](#)
 - calculatehorzdistance, [32](#)
 - calculatehorzlengths, [32](#)
 - calculatesegmentslopes, [33](#)
 - checkcrosssection, [34](#)
 - copygeometry, [34](#)
 - deallocategeometry, [35](#)
 - determinesegmenttypes, [35](#)
 - initializegeometry, [36](#)
 - mergesequentialberms, [36](#)
 - removeberms, [38](#)
 - removedikesegments, [39](#)
 - splitcrosssection, [39](#)
- getlanguage
 - dllovertopping, [11](#)
 - overtoppingmessages, [52](#)
- getovertoppingformat
 - overtoppingmessages, [52](#)
- getovertoppingmessage
 - overtoppingmessages, [52](#)
- getovertoppingparameter
 - overtoppingmessages, [53](#)
- h
 - typedefinitionsovertopping::tpload, [76](#)
- hm0
 - typedefinitionsovertopping::tpload, [76](#)
- initializegeometry
 - geometrymoduleovertopping, [36](#)
- innercalculation
 - waverunup, [60](#)
- interpolateresultssections
 - mainmoduleovertopping, [45](#)
- isberm
 - omkeervariantmodule, [50](#)
- isequalreal
 - formulamoduleovertopping, [26](#)
- isequalzero
 - formulamoduleovertopping, [26](#)
- isvalidz
 - omkeervariantmodule, [50](#)
- iteratetogivendischarge
 - omkeervariantmodule, [48](#)
- iteratetogivendischargevalidprofile
 - omkeervariantmodule, [49](#)
- iterationwaverunup
 - waverunup, [61](#)
- language
 - overtoppingmessages, [53](#)
- m_z2
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- mainModuleOvertopping.f90, [87](#)

- mainmoduleovertopping, [40](#)
 - calculateovertopping, [41](#)
 - calculateovertoppingnegativefreeboard, [41](#)
 - calculateovertoppingsection, [42](#)
 - calculatewaveovertopping, [43](#)
 - checkinputdata, [44](#)
 - checkmodelfactors, [45](#)
 - interpolateresultssections, [45](#)
- margindiff
 - typedefinitionsovertopping, [56](#)
- maringrad
 - typedefinitionsovertopping, [57](#)
- maxfilenamelength
 - modulelogging, [47](#)
- maxmsg
 - overtoppingmessages, [54](#)
- maxpar
 - overtoppingmessages, [54](#)
- mergesequentialberms
 - geometrymoduleovertopping, [36](#)
- minzberm
 - omkeervariantmodule, [50](#)
- ModuleLogging.f90, [87](#)
- modulelogging, [46](#)
 - currentlogging, [47](#)
 - maxfilenamelength, [47](#)
- modulelogging::tlogging, [72](#)
 - filename, [73](#)
 - verbosity, [73](#)
- mz2_max
 - typedefinitionsovertopping, [57](#)
- mz2_min
 - typedefinitionsovertopping, [57](#)
- nbermssegments
 - typedefinitionsovertopping::tpgeometry, [74](#)
- ncoordinates
 - typedefinitionsovertopping::tpgeometry, [74](#)
- normal
 - overtoppinginterface::overtoppinggeometrytype, [70](#)
 - overtoppinginterface::overtoppinggeometrytypef, [71](#)
- npoints
 - overtoppinginterface::overtoppinggeometrytype, [70](#)
 - overtoppinginterface::overtoppinggeometrytypef, [71](#)
- omkeerVariantModule.f90, [88](#)
- omkeervariantf
 - dlovertopping, [11](#)
- omkeervariantmodule, [47](#)
 - checkifonberm, [47](#)
 - dischargeprofile, [50](#)
 - isberm, [50](#)
 - isvalidz, [50](#)
 - iteratetogivendischarge, [48](#)
 - iteratetogivendischargevalidprofile, [49](#)
- minzberm, [50](#)
- toldikeheight, [51](#)
- toldischarge, [51](#)
- zprofile, [51](#)
- overtoppingInterface.f90, [88](#)
- OvertoppingMessages.f90, [89](#)
- overtoppinginterface, [51](#)
 - varmodelfactorcriticalovertopping, [51](#)
- overtoppinginterface::overtoppinggeometrytype, [69](#)
 - normal, [70](#)
 - npoints, [70](#)
 - roughness, [70](#)
 - xcoords, [70](#)
 - ycoords, [70](#)
- overtoppinginterface::overtoppinggeometrytypef, [71](#)
 - normal, [71](#)
 - npoints, [71](#)
 - roughness, [71](#)
 - xcoords, [72](#)
 - ycoords, [72](#)
- overtoppinginterface::tpprofilecoordinate, [81](#)
 - roughness, [81](#)
 - xcoordinate, [82](#)
 - zcoordinate, [82](#)
- overtoppingmessages, [51](#)
 - getlanguage, [52](#)
 - getovertoppingformat, [52](#)
 - getovertoppingmessage, [52](#)
 - getovertoppingparameter, [53](#)
 - language, [53](#)
 - maxmsg, [54](#)
 - maxpar, [54](#)
 - setlanguage, [53](#)
- phi
 - typedefinitionsovertopping::tpload, [77](#)
- profileinstructure
 - zfunctionsovertopping, [65](#)
- psi
 - typedefinitionsovertopping::tpgeometry, [74](#)
- qo
 - typedefinitionsovertopping::tpovertopping, [78](#)
- realrootscubicfunction
 - formulamoduleovertopping, [26](#)
- reductionfactorforeshore
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- relaxationfactor
 - typedefinitionsovertopping::tpovertoppinginput, [80](#)
- removeberms
 - geometrymoduleovertopping, [38](#)
- removedikessegments
 - geometrymoduleovertopping, [39](#)
- rfactor_max
 - typedefinitionsovertopping, [57](#)
- rfactor_min
 - typedefinitionsovertopping, [57](#)
- rootsdepressedcubic

- formulamoduleovertopping, 27
- rootsgeneralcubic
 - formulamoduleovertopping, 27
- roughness
 - overtoppinginterface::overtoppinggeometrytype, 70
 - overtoppinginterface::overtoppinggeometrytypef, 71
 - overtoppinginterface::tpprofilecoordinate, 81
- roughnessfactors
 - typedefinitionsovertopping::tpgeometry, 74
- segmentslopes
 - typedefinitionsovertopping::tpgeometry, 74
- segmenttypes
 - typedefinitionsovertopping::tpgeometry, 75
- setlanguage
 - dllovertopping, 12
 - overtoppingmessages, 53
- slope_max
 - typedefinitionsovertopping, 57
- slope_min
 - typedefinitionsovertopping, 57
- splitcrosssection
 - geometrymoduleovertopping, 39
- tm_10
 - typedefinitionsovertopping::tpload, 77
- toldikeheight
 - omkeervariantmodule, 51
- toldischarge
 - omkeervariantmodule, 51
- typeDefinitionsOvertopping.f90, 90
- typedefinitionsovertopping, 54
 - berm_max, 55
 - berm_min, 55
 - fb_max, 55
 - fb_min, 55
 - fn_max, 56
 - fn_min, 56
 - foreshore_max, 56
 - foreshore_min, 56
 - frunup1, 56
 - frunup2, 56
 - frunup3, 56
 - fs_max, 56
 - fs_min, 56
 - margindiff, 56
 - marginrad, 57
 - mz2_max, 57
 - mz2_min, 57
 - rfactor_max, 57
 - rfactor_min, 57
 - slope_max, 57
 - slope_min, 57
 - xdiff_min, 57
 - z2_iter_max1, 57
 - z2_iter_max2, 58
 - z2_margin, 58
- typedefinitionsovertopping::tpgeometry, 73
 - nbermsegments, 74
 - ncoordinates, 74
 - psi, 74
 - roughnessfactors, 74
 - segmentslopes, 74
 - segmenttypes, 75
 - xcoorddiff, 75
 - xcoordinates, 75
 - ycoorddiff, 75
 - ycoordinates, 75
- typedefinitionsovertopping::tpload, 75
 - h, 76
 - hm0, 76
 - phi, 77
 - tm_10, 77
- typedefinitionsovertopping::tpovertopping, 77
 - qo, 78
 - z2, 78
- typedefinitionsovertopping::tpovertoppinginput, 78
 - computedovertopping, 80
 - criticalovertopping, 80
 - factordeterminationq_b_f_b, 80
 - factordeterminationq_b_f_n, 80
 - fshallow, 80
 - m_z2, 80
 - reductionfactorforeshore, 80
 - relaxationfactor, 80
- validateinputc
 - dllovertopping, 12
- validateinputf
 - dllovertopping, 13
- varmodelfactorcriticalovertopping
 - overtoppinginterface, 51
- verbosity
 - modulelogging::tlogging, 73
- versionnumber
 - dllovertopping, 14
- waveRunup.f90, 91
- waverunup, 58
 - convergedwithresidu, 58
 - determinestartingvalue, 59
 - findsmallestresidu, 59
 - innercalculation, 60
 - iterationwaverunup, 61
- xcoorddiff
 - typedefinitionsovertopping::tpgeometry, 75
- xcoordinate
 - overtoppinginterface::tpprofilecoordinate, 82
- xcoordinates
 - typedefinitionsovertopping::tpgeometry, 75
- xcoords
 - overtoppinginterface::overtoppinggeometrytype, 70
 - overtoppinginterface::overtoppinggeometrytypef, 72

- xdiff_min
 - typedefinitionsovertopping, [57](#)
- ycoorddiff
 - typedefinitionsovertopping::tpgeometry, [75](#)
- ycoordinates
 - typedefinitionsovertopping::tpgeometry, [75](#)
- ycoords
 - overtoppinginterface::overtoppinggeometrytype, [70](#)
 - overtoppinginterface::overtoppinggeometrytypef, [72](#)
- z2
 - typedefinitionsovertopping::tpovertopping, [78](#)
- z2_iter_max1
 - typedefinitionsovertopping, [57](#)
- z2_iter_max2
 - typedefinitionsovertopping, [58](#)
- z2_margin
 - typedefinitionsovertopping, [58](#)
- zFunctionsOvertopping.f90, [92](#)
- zcoordinate
 - overtoppinginterface::tpprofilecoordinate, [82](#)
- zfunclogratios
 - zfunctionsovertopping, [66](#)
- zfunctionsovertopping, [62](#)
 - adjustprofile, [63](#)
 - calculateqorto, [64](#)
 - profileinstructure, [65](#)
 - zfunclogratios, [66](#)
- zprofile
 - omkeervariantmodule, [51](#)