

Dikes Overtopping Kernel - Technical documentation

Generated by Doxygen 1.8.9.1

Tue Dec 1 2015 17:53:04

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 | 7 |
| 4.1.2.1 | calculateqo | 7 |
| 4.1.2.2 | calculateqof | 8 |
| 4.1.2.3 | calcvalue | 9 |
| 4.1.2.4 | geometry_c_f | 10 |
| 4.1.2.5 | validateinputc | 10 |
| 4.1.2.6 | validateinputf | 11 |
| 4.1.2.7 | versionnumber | 12 |
| 4.2 | factormodulertooverlapping Module Reference | 12 |
| 4.2.1 | Function/Subroutine Documentation | 12 |
| 4.2.1.1 | calculategammab | 12 |
| 4.2.1.2 | calculategammabeta | 13 |
| 4.2.1.3 | calculategammaf | 13 |
| 4.2.1.4 | calculatetanalpha | 14 |
| 4.3 | formulamodulertooverlapping Module Reference | 15 |
| 4.3.1 | Function/Subroutine Documentation | 16 |
| 4.3.1.1 | adjustinfluencefactors | 16 |
| 4.3.1.2 | calculateanglewaveattack | 16 |
| 4.3.1.3 | calculatebreakerlimit | 17 |
| 4.3.1.4 | calculatebreakerparameter | 17 |

| | | |
|----------|-----------------------------------------------|----|
| 4.3.1.5 | calculatewavelength | 17 |
| 4.3.1.6 | calculatewaveovertoppingdischarge | 18 |
| 4.3.1.7 | calculatewaverunup | 18 |
| 4.3.1.8 | calculatewavesteepness | 19 |
| 4.3.1.9 | cubicroots | 20 |
| 4.3.1.10 | isequalreal | 20 |
| 4.3.1.11 | isequalzero | 20 |
| 4.3.1.12 | realrootscubicfunction | 21 |
| 4.3.1.13 | rootsdepressedcubic | 21 |
| 4.3.1.14 | rootsgeneralcubic | 22 |
| 4.4 | geometrymodulertooverlapping Module Reference | 22 |
| 4.4.1 | Function/Subroutine Documentation | 23 |
| 4.4.1.1 | adjustnonhorizontalberms | 23 |
| 4.4.1.2 | allocatevectorsgeometry | 24 |
| 4.4.1.3 | calculatehorzdistance | 24 |
| 4.4.1.4 | calculatehorzlengths | 25 |
| 4.4.1.5 | calculatesegmentslopes | 25 |
| 4.4.1.6 | checkcrosssection | 26 |
| 4.4.1.7 | copygeometry | 27 |
| 4.4.1.8 | deallocategeometry | 27 |
| 4.4.1.9 | determinesegmenttypes | 28 |
| 4.4.1.10 | initializegeometry | 28 |
| 4.4.1.11 | isequalgeometry | 29 |
| 4.4.1.12 | mergesequentialberms | 30 |
| 4.4.1.13 | removeberms | 31 |
| 4.4.1.14 | removedikesegments | 32 |
| 4.4.1.15 | splitcrosssection | 32 |
| 4.4.1.16 | writecrosssection | 33 |
| 4.5 | mainmodulertooverlapping Module Reference | 33 |
| 4.5.1 | Function/Subroutine Documentation | 34 |
| 4.5.1.1 | calculateovertopping | 34 |
| 4.5.1.2 | calculateovertoppingnegativefreeboard | 35 |
| 4.5.1.3 | calculateovertoppingsection | 35 |
| 4.5.1.4 | calculatewaveovertopping | 36 |
| 4.5.1.5 | checkinputdata | 37 |
| 4.5.1.6 | checkmodelfactors | 38 |
| 4.5.1.7 | converttooverlappinginput | 38 |
| 4.5.1.8 | interpolateresultssections | 38 |
| 4.6 | modulelogging Module Reference | 39 |
| 4.6.1 | Variable Documentation | 39 |

| | | |
|----------|------------------------------------------------|----|
| 4.6.1.1 | currentlogging | 39 |
| 4.6.1.2 | maxfilenamelength | 39 |
| 4.7 | overtoppinginterface Module Reference | 39 |
| 4.7.1 | Variable Documentation | 40 |
| 4.7.1.1 | varmodelfactorcriticalovertopping | 40 |
| 4.8 | typedefinitionsrtooverlapping Module Reference | 40 |
| 4.8.1 | Variable Documentation | 41 |
| 4.8.1.1 | berm_max | 41 |
| 4.8.1.2 | berm_min | 41 |
| 4.8.1.3 | fb_max | 42 |
| 4.8.1.4 | fb_min | 42 |
| 4.8.1.5 | fn_max | 42 |
| 4.8.1.6 | fn_min | 42 |
| 4.8.1.7 | foreshore_max | 42 |
| 4.8.1.8 | foreshore_min | 42 |
| 4.8.1.9 | frunup1_max | 42 |
| 4.8.1.10 | frunup1_min | 42 |
| 4.8.1.11 | frunup2_max | 42 |
| 4.8.1.12 | frunup2_min | 43 |
| 4.8.1.13 | frunup3_max | 43 |
| 4.8.1.14 | frunup3_min | 43 |
| 4.8.1.15 | fs_max | 43 |
| 4.8.1.16 | fs_min | 43 |
| 4.8.1.17 | margindiff | 43 |
| 4.8.1.18 | margingrad | 43 |
| 4.8.1.19 | mz2_max | 43 |
| 4.8.1.20 | mz2_min | 43 |
| 4.8.1.21 | rfactor_max | 44 |
| 4.8.1.22 | rfactor_min | 44 |
| 4.8.1.23 | slope_max | 44 |
| 4.8.1.24 | slope_min | 44 |
| 4.8.1.25 | xdiff_min | 44 |
| 4.8.1.26 | z2_iter_max1 | 44 |
| 4.8.1.27 | z2_iter_max2 | 44 |
| 4.8.1.28 | z2_margin | 44 |
| 4.9 | waverunup Module Reference | 44 |
| 4.9.1 | Function/Subroutine Documentation | 45 |
| 4.9.1.1 | convergedwithresidu | 45 |
| 4.9.1.2 | determinestartingvalue | 45 |
| 4.9.1.3 | findsmallestresidu | 45 |

| | | |
|----------|---------------------------------------------------------------|-----------|
| 4.9.1.4 | innercalculation | 46 |
| 4.9.1.5 | iterationwaverunup | 47 |
| 4.10 | zfunctionswtiovertopping Module Reference | 48 |
| 4.10.1 | Detailed Description | 48 |
| 4.10.2 | Function/Subroutine Documentation | 48 |
| 4.10.2.1 | adjustprofile | 49 |
| 4.10.2.2 | calculateqorto | 50 |
| 4.10.2.3 | profileinstructure | 51 |
| 4.10.2.4 | zfunclogratios | 52 |
| 5 | Data Type Documentation | 55 |
| 5.1 | overtoppinginterface::overtoppinggeometrytype Type Reference | 55 |
| 5.1.1 | Detailed Description | 56 |
| 5.1.2 | Member Data Documentation | 56 |
| 5.1.2.1 | normal | 56 |
| 5.1.2.2 | npoints | 56 |
| 5.1.2.3 | roughness | 56 |
| 5.1.2.4 | xcoords | 56 |
| 5.1.2.5 | ycoords | 56 |
| 5.2 | overtoppinginterface::overtoppinggeometrytypef Type Reference | 57 |
| 5.2.1 | Detailed Description | 57 |
| 5.2.2 | Member Data Documentation | 57 |
| 5.2.2.1 | normal | 57 |
| 5.2.2.2 | npoints | 57 |
| 5.2.2.3 | roughness | 58 |
| 5.2.2.4 | xcoords | 58 |
| 5.2.2.5 | ycoords | 58 |
| 5.3 | modulelogging::tlogging Type Reference | 58 |
| 5.3.1 | Detailed Description | 58 |
| 5.3.2 | Member Data Documentation | 59 |
| 5.3.2.1 | filename | 59 |
| 5.3.2.2 | verbosity | 59 |
| 5.4 | typedefinitionsrtoovertopping::tpgeometry Type Reference | 59 |
| 5.4.1 | Detailed Description | 60 |
| 5.4.2 | Member Data Documentation | 60 |
| 5.4.2.1 | nbermsements | 60 |
| 5.4.2.2 | ncoordinates | 60 |
| 5.4.2.3 | psi | 60 |
| 5.4.2.4 | roughnessfactors | 60 |
| 5.4.2.5 | segmentslopes | 61 |

| | | |
|----------|----------------------------------------------------------------|----|
| 5.4.2.6 | segmenttypes | 61 |
| 5.4.2.7 | xcoorddiff | 61 |
| 5.4.2.8 | xcoordinates | 61 |
| 5.4.2.9 | ycoorddiff | 61 |
| 5.4.2.10 | ycoordinates | 61 |
| 5.5 | typedefinitionsrtovertopping::tpload Type Reference | 61 |
| 5.5.1 | Detailed Description | 62 |
| 5.5.2 | Member Data Documentation | 62 |
| 5.5.2.1 | h | 62 |
| 5.5.2.2 | hm0 | 63 |
| 5.5.2.3 | phi | 63 |
| 5.5.2.4 | tm_10 | 63 |
| 5.6 | typedefinitionsrtovertopping::tpvertopping Type Reference | 63 |
| 5.6.1 | Detailed Description | 64 |
| 5.6.2 | Member Data Documentation | 64 |
| 5.6.2.1 | qo | 64 |
| 5.6.2.2 | z2 | 64 |
| 5.7 | typedefinitionsrtovertopping::tpvertoppinginput Type Reference | 64 |
| 5.7.1 | Detailed Description | 66 |
| 5.7.2 | Member Data Documentation | 66 |
| 5.7.2.1 | computedvertopping | 66 |
| 5.7.2.2 | criticalvertopping | 66 |
| 5.7.2.3 | factordeterminationq_b_f_b | 66 |
| 5.7.2.4 | factordeterminationq_b_f_n | 66 |
| 5.7.2.5 | frunup1 | 66 |
| 5.7.2.6 | frunup2 | 66 |
| 5.7.2.7 | frunup3 | 67 |
| 5.7.2.8 | fshallow | 67 |
| 5.7.2.9 | m_z2 | 67 |
| 5.7.2.10 | reductionfactorforeshore | 67 |
| 5.7.2.11 | relaxationfactor | 67 |
| 5.7.2.12 | typerunup | 67 |
| 5.8 | vertoppinginterface::tpprofilecoordinate Type Reference | 68 |
| 5.8.1 | Detailed Description | 68 |
| 5.8.2 | Member Data Documentation | 68 |
| 5.8.2.1 | roughness | 68 |
| 5.8.2.2 | xcoordinate | 69 |
| 5.8.2.3 | zcoordinate | 69 |

| | | |
|--------|------------------------------------------------------------------|-----------|
| 6.1 | dllOvertopping.f90 File Reference | 71 |
| 6.1.1 | Detailed Description | 71 |
| 6.2 | factorModuleRTOovertopping.f90 File Reference | 72 |
| 6.2.1 | Detailed Description | 72 |
| 6.3 | formulaModuleRTOovertopping.f90 File Reference | 72 |
| 6.3.1 | Detailed Description | 73 |
| 6.4 | geometryModuleRTOovertopping.f90 File Reference | 73 |
| 6.4.1 | Detailed Description | 74 |
| 6.5 | mainModuleRTOovertopping.f90 File Reference | 75 |
| 6.5.1 | Detailed Description | 75 |
| 6.6 | ModuleLogging.f90 File Reference | 75 |
| 6.6.1 | Detailed Description | 76 |
| 6.7 | overtoppingInterface.f90 File Reference | 76 |
| 6.7.1 | Detailed Description | 76 |
| 6.8 | typeDefinitionsRTOovertopping.f90 File Reference | 76 |
| 6.8.1 | Detailed Description | 78 |
| 6.9 | waveRunup.f90 File Reference | 78 |
| 6.9.1 | Detailed Description | 78 |
| 6.10 | zFunctionsWTIOovertopping.f90 File Reference | 78 |
| 6.10.1 | Detailed Description | 79 |
| | Index | 81 |

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

| | |
|-----------------------------------------------------------------------------------|----|
| dllovertopping | |
| Calculate one type of overtopping | 7 |
| factormodulertoovertopping | 12 |
| formulamodulertoovertopping | 15 |
| geometrymodulertoovertopping | 22 |
| mainmodulertoovertopping | 33 |
| modulelogging | 39 |
| overtoppinginterface | 39 |
| typedefinitionsrtoovertopping | 40 |
| waverunup | 44 |
| zfunctionswtiovertopping | |
| Module for the Limit State Functions (Z-functions) for wave overtopping | 48 |

Chapter 2

Data Type Index

2.1 Class List

Here are the data types with brief descriptions:

| | |
|-------------------------------------------------------------------|----|
| overtoppinginterface::overtoppinggeometrytype | 55 |
| overtoppinginterface::overtoppinggeometrytypef | 57 |
| modulelogging::tlogging | |
| TLogging: structure for steering the logging | 58 |
| typedefinitionsrtoovertopping::tpgeometry | |
| TpGeometry: structure with geometry data | 59 |
| typedefinitionsrtoovertopping::tpload | |
| TpLoad: structure with load parameters | 61 |
| typedefinitionsrtoovertopping::tpovertopping | |
| TpOvertopping: structure with overtopping results | 63 |
| typedefinitionsrtoovertopping::tpovertoppinginput | |
| OvertoppingModelFactors: C-structure with model factors | 64 |
| overtoppinginterface::tpprofilecoordinate | 68 |

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: | 71 |
| factorModuleRTOvertopping.f90 | This file contains a module with functions for the slope angle and influence factors | 72 |
| formulaModuleRTOvertopping.f90 | This file contains a module with the core computations for Dikes Overtopping | 72 |
| geometryModuleRTOvertopping.f90 | This file contains a module with the core computations for Dikes Overtopping related to the geometry | 73 |
| mainModuleRTOvertopping.f90 | This file contains a module with the core computations for Dikes Overtopping | 75 |
| ModuleLogging.f90 | Module for steering the extra logging | 75 |
| overtoppingInterface.f90 | This file contains the parameters and types (structs) as part of the interface to and from dll↔ Overtopping | 76 |
| typeDefinitionsRTOvertopping.f90 | This file contains a module with the type definitions for Dikes Overtopping | 76 |
| waveRunup.f90 | This file contains a module with the core computations for Dikes Overtopping | 78 |
| zFunctionsWTIOvertopping.f90 | This file contains the limit state functions for wave overtopping within WTl | 78 |

Chapter 4

Module Documentation

4.1 dllovertopping Module Reference

Calculate one type of overtopping.

Functions/Subroutines

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

4.1.1 Detailed Description

Calculate one type of overtopping.

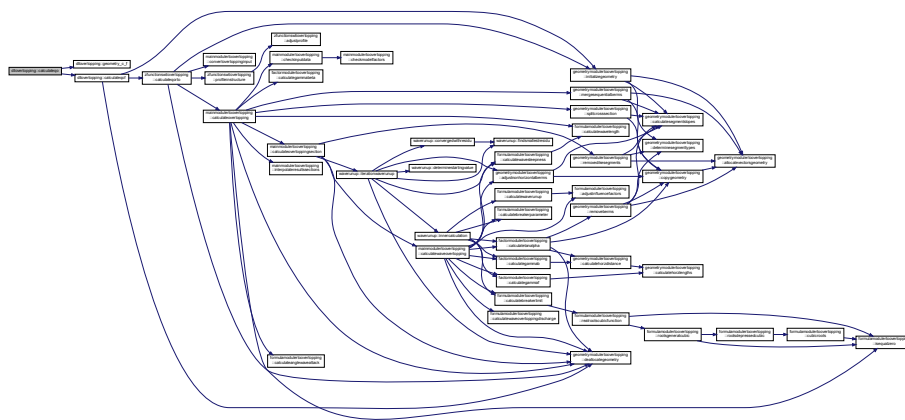
4.1.2 Function/Subroutine Documentation

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

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

| | | |
|---------|----------------------|--------------------------------------------------|
| 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 |

Here is the call graph for this function:

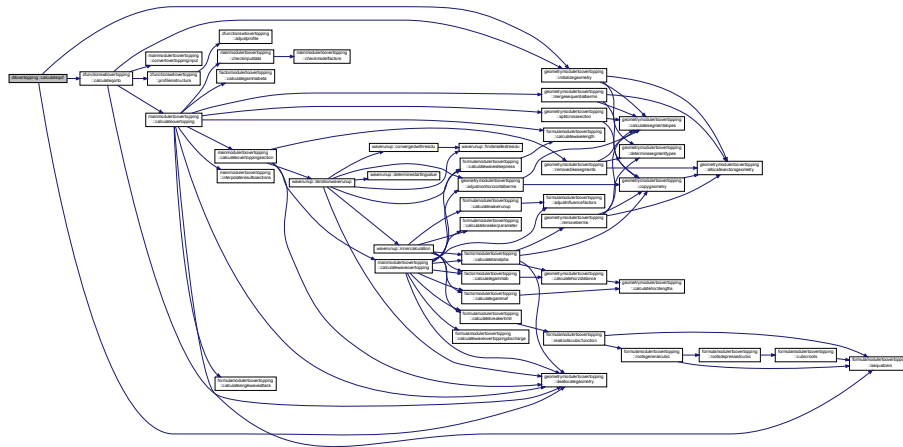


Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.

| | | |
|---------|---------------------|--------------------------------------------|
| 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 |

Generated on Tue Dec 1 2015 17:53:04 for Dikes Overtopping Kernel - Technical documentation by Doxygen

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.2.3 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*)

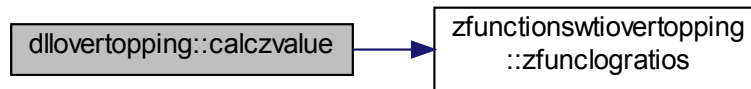
Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.

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 104 of file dllovertopping.f90.

Here is the call graph for this function:



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

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

Parameters

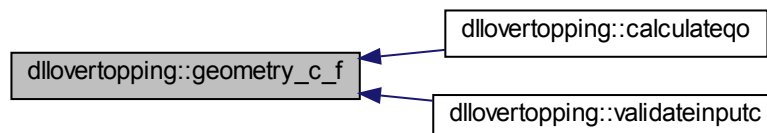
| | | |
|----|----------------------|--------------------------------------------------|
| in | <i>geometryinput</i> | struct with geometry and roughness as c-pointers |
|----|----------------------|--------------------------------------------------|

Returns

fortran struct with geometry and roughness

Definition at line 225 of file `dllOvertopping.f90`.

Here is the caller graph for this function:



4.1.2.5 `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 ValidateInputF: convert C-like input structures to Fortran input structures.

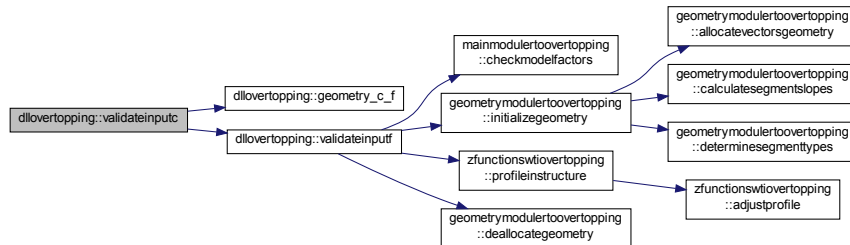
Parameters

| | | |
|---------|----------------------|--------------------------------------------------|
| 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 124 of file dllOvertopping.f90.

Here is the call graph for this function:



4.1.2.6 subroutine, public dlovertopping::validateinputf (type(overtoppinggeometrytypef), intent(in) *geometryF*, real(kind=wp), intent(in) *dikeHeight*, type(tpovertoppinginput), intent(inout) *modelFactors*, logical, intent(out) *success*, character(len=*) , intent(out) *errorText*)

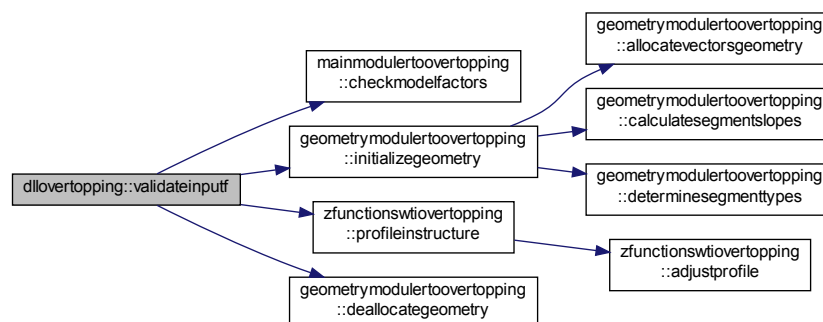
Subroutine that validates the geometry.

Parameters

| | | |
|---------|---------------------|--------------------------------------------|
| in | <i>geometryf</i> | struct with geometry and roughness |
| 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 146 of file dllOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

Subroutine that delivers the version number.

Parameters

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

Definition at line 203 of file dllovertopping.f90.

4.2 factormodulertoovertopping Module Reference

Functions/Subroutines

- subroutine, public [calculatetanalpha](#) (h, Hm0, z2, geometry, tanAlpha, succes, errorMessage)
calculateTanAlpha representative slope angle
- subroutine, public [calculategammabeta](#) (Hm0, Tm_10, beta, gammaBeta_z, gammaBeta_o)
calculateGammaBeta influence factor angle of wave attack
- subroutine, public [calculategammaf](#) (h, ksi0, ksi0Limit, gammaB, z2, geometry, gammaF, succes, error↔Message)
calculateGammaF influence factor roughness
- subroutine, public [calculategammab](#) (h, Hm0, z2, geometry, gammaB, succes, errorMessage)
calculateGammaB influence factor berms

4.2.1 Function/Subroutine Documentation

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

calculateGammaB influence factor berms

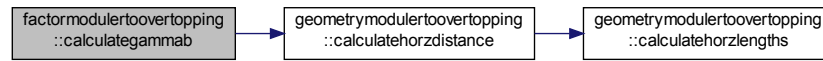
Parameters

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

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

Definition at line 279 of file factorModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.1.2 subroutine, public factormodulertoovertopping::calculategammabeta (real(wp), intent(inout) *Hm0*, real(wp), intent(inout) *Tm_10*, real(wp), intent(in) *beta*, real(wp), intent(out) *gammaBeta_z*, real(wp), intent(out) *gammaBeta_o*)

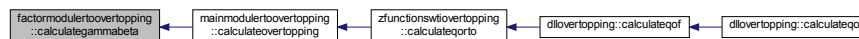
calculateGammaBeta influence factor angle of wave attack

Parameters

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

Definition at line 114 of file factorModuleRTOovertopping.f90.

Here is the caller graph for this function:



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

calculateGammaF influence factor roughness

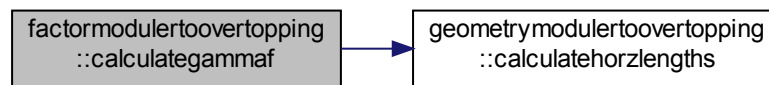
Parameters

| | | |
|----|-------------|---------------------------|
| in | <i>h</i> | local water level (m+NAP) |
| in | <i>ksi0</i> | breaker parameter |

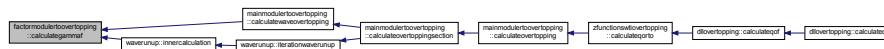
| | | |
|-----|---------------------|-------------------------------|
| in | <i>ksi0limit</i> | limit value breaker parameter |
| in | <i>gammab</i> | influence factor berms |
| in | <i>z2</i> | 2% wave run-up (m) |
| in | <i>geometry</i> | structure with geometry data |
| out | <i>gammaf</i> | influence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

Definition at line 154 of file factorModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

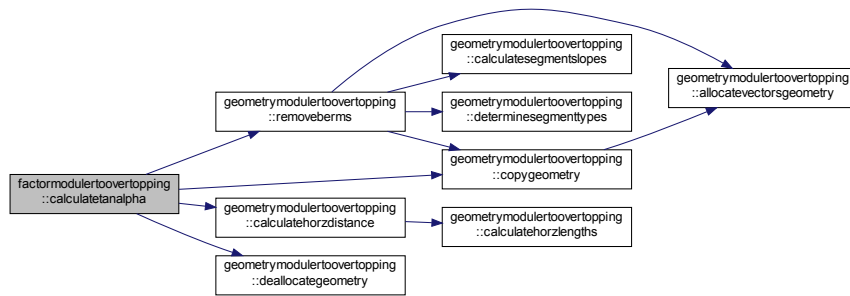
calculateTanAlpha representative slope angle

Parameters

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

Definition at line 35 of file factorModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3 formulamodulertoovertopping Module Reference

Functions/Subroutines

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

cubicRoots: calculate the roots of a cubic function

- logical function, public [isequalreal](#) (x1, x2)

isEqualReal: are two reals (almost) equal

- logical function, public [isequalzero](#) (x)

isEqualZero: is a real (almost) zero

4.3.1 Function/Subroutine Documentation

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

`adjustInfluenceFactors`: adjust the influence factors

Parameters

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

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

Here is the caller graph for this function:



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

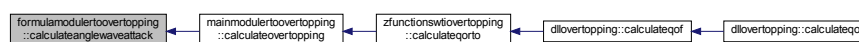
`calculateAngleWaveAttack`: calculate the angle of wave attack

Parameters

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

Definition at line 285 of file `formulaModuleRTOOvertopping.f90`.

Here is the caller graph for this function:

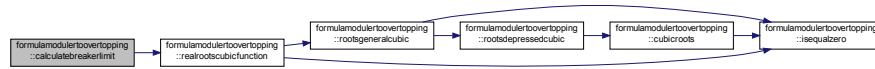


4.3.1.3 subroutine, public formulamodulertoovertopping::calculatebreakerlimit (type (tpovertoppinginput), intent(in) *modelFactors*, real(wp), intent(in) *gammaB*, real(wp), intent(out) *ksi0Limit*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

calculateBreakerLimit: calculate the breaker limit

Definition at line 308 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

calculateBreakerParameter: calculate the breaker parameter

Parameters

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

Definition at line 244 of file formulaModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.3.1.5 subroutine, public formulamodulertoovertopping::calculatewavelength (real(wp), intent(in) *Tm_10*, real(wp), intent(out) *L0*)

calculateWaveLength: calculate the wave length

Parameters

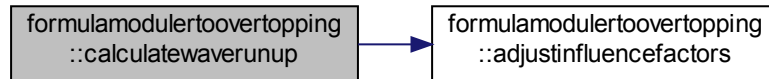
| | | |
|-----|--------------|--------------------------|
| in | <i>tm_10</i> | spectral wave period (s) |
| out | <i>l0</i> | wave length (m) |

Definition at line 181 of file formulaModuleRTOovertopping.f90.

| | | |
|---------|---------------------|---------------------------------------|
| 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 34 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.8 subroutine, public formulamodulertoovtopping::calculatewavesteepness (real(wp), intent(in) *Hm0*, real(wp), intent(in) *Tm_10*, real(wp), intent(out) *s0*, logical, intent(out) *succes*, character(len=*), intent(out) *errorMessage*)

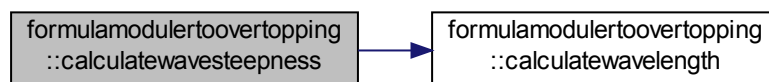
calculateWaveSteepness: calculate the wave steepness

Parameters

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

Definition at line 202 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

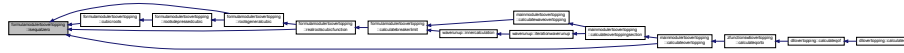


Parameters

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

Definition at line 700 of file formulaModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.3.1.12 subroutine, public formulamodulertovertopping::realrootscubicfunction (real(wp), intent(in) *a*, real(wp), intent(in) *b*, real(wp), intent(in) *c*, real(wp), intent(in) *d*, integer, intent(out) *N*, real(wp), dimension(3), intent(out) *x*, logical, intent(out) *succes*, character(len=*), intent(out) *errorMessage*)

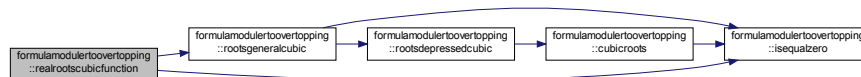
realRootsCubicFunction: calculate the roots of a cubic function

Parameters

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

Definition at line 480 of file formulaModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.13 subroutine, public formulamodulertovertopping::rootsdepressedcubic (real(wp), intent(in) *p*, real(wp), intent(in) *q*, double complex, dimension(3), intent(out) *z*)

rootsDepressedCubic: calculate the roots of a depressed cubic function

Parameters

| | | |
|----|----------|-------------------------------|
| in | <i>p</i> | coefficient p depressed cubic |
|----|----------|-------------------------------|

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)
allocateVectorsGeometry: allocate the geometry vectors
- subroutine, public [deallocategemetry](#) (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)
copyGeometry: copy a geometry structure
- subroutine, public [isequalgeometry](#) (geometry1, geometry2, succes, errorMessage)
isEqualGeometry: are two geometries equal
- subroutine, public [mergesquentialberms](#) (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 [writecrosssection](#) (geometry, geometryName)
writeCrossSection: write a cross section

4.4.1 Function/Subroutine Documentation

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

adjustNonHorizontalBerms: adjust non-horizontal berms

Parameters

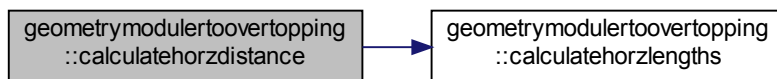
| | | |
|----|-----------------|------------------------------|
| in | <i>geometry</i> | structure with geometry data |
|----|-----------------|------------------------------|

Parameters

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

Definition at line 1023 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

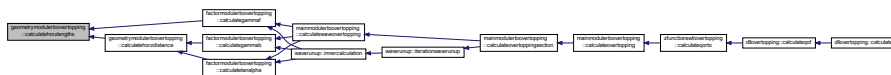
calculateHorzLengths: calculate horizontal lengths

Parameters

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

Definition at line 927 of file geometryModuleRTOoverlapping.f90.

Here is the caller graph for this function:



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

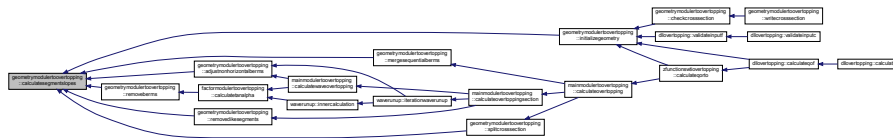
calculateSegmentSlopes: calculate the segment slopes

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 248 of file geometryModuleRTOovertopping.f90.

Here is the caller graph for this function:



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

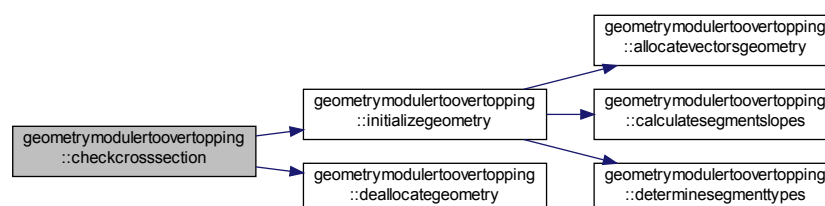
checkCrossSection: check cross section

Parameters

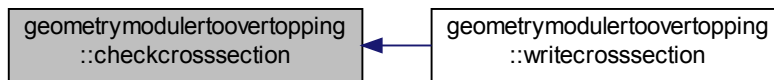
| | | |
|-----|-------------------------|-----------------------|
| in | <i>psi</i> | dike normal (degree) |
| in | <i>ncoordinates</i> | number of coordinates |
| in | <i>xcoordinates</i> | x-coordinates (m) |
| in | <i>ycoordinates</i> | y-coordinates (m+NAP) |
| in | <i>roughnessfactors</i> | roughness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

Definition at line 34 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.7 subroutine, public geometrymodulertooverlapping::copygeometry (type (tpgeometry), intent(in) *geometry*, type (tpgeometry), intent(inout) *geometryCopy*)

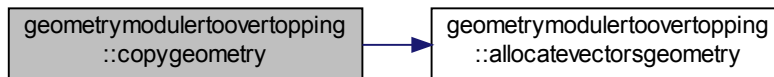
copyGeometry: copy a geometry structure

Parameters

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

Definition at line 330 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

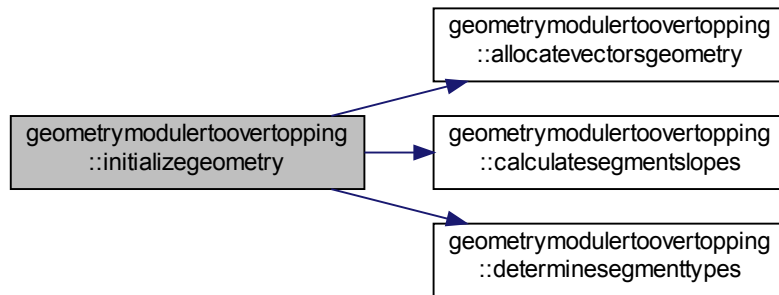
```
deallocateGeometry: deallocate the geometry vectors
```

Parameters

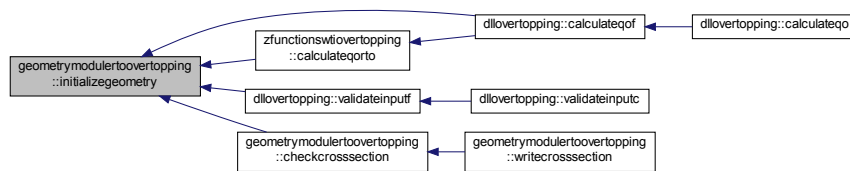
| | | |
|---------|-----------------|------------------------------|
| in, out | <i>geometry</i> | structure with geometry data |
|---------|-----------------|------------------------------|

Definition at line 225 of file geometryModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.11 subroutine, public geometrymodulertovertopping::isequalgeometry (type (tpgeometry), intent(in) *geometry1*, type (tpgeometry), intent(in) *geometry2*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

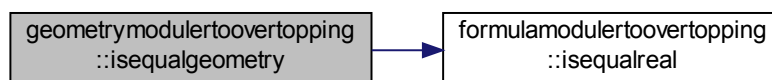
isEqualGeometry: are two geometries equal

Parameters

| | | |
|-----|---------------------|--------------------------------|
| in | <i>geometry1</i> | structure with geometry data 1 |
| in | <i>geometry2</i> | structure with geometry data 2 |
| out | <i>succes</i> | flag for succes |
| out | <i>errormessage</i> | error message |

Definition at line 377 of file geometryModuleRTOOvertopping.f90.

Here is the call graph for this function:



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

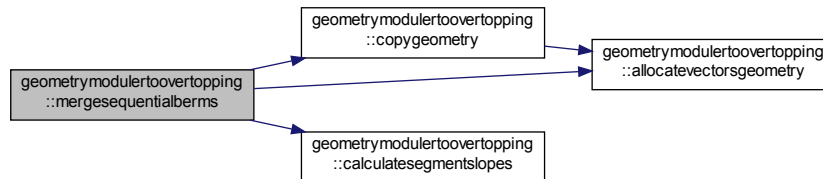
mergeSequentialBerms: merge sequential berms

Parameters

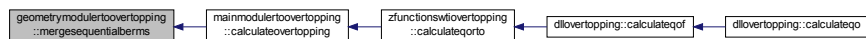
| | | |
|---------|----------------------------|--------------------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in, out | <i>geometrymergedberms</i> | geometry data with merged sequential berms |
| out | <i>succes</i> | flag for succes |
| out | <i>errormessage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

Definition at line 471 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

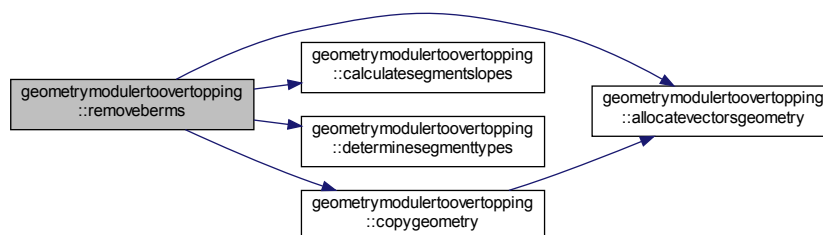
removeBerms: remove berms

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 664 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:

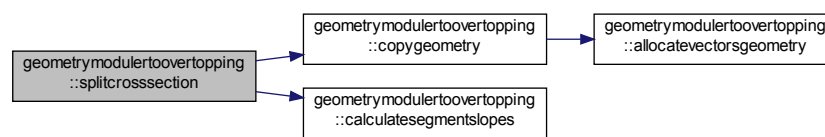


Parameters

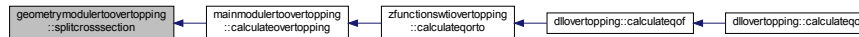
| | | |
|-----|-------------------------|-------------------------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in | <i>l0</i> | wave length (m) |
| out | <i>nwideberms</i> | number of wide berms |
| out | <i>geometrysectionb</i> | geometry data with wide berms to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

Definition at line 825 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.16 subroutine, public geometrymodulertooverlapping::writecrosssection (type (tpgeometry), intent(in) *geometry*, character(len=*), intent(in) *geometryName*)

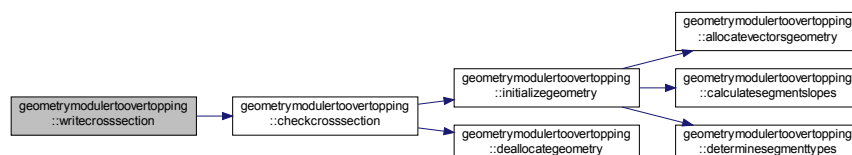
writeCrossSection: write a cross section

Parameters

| | | |
|----|---------------------|------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in | <i>geometryname</i> | description of geometry data |

Definition at line 1066 of file geometryModuleRTOoverlapping.f90.

Here is the call graph for this function:



4.5 mainmodulertooverlapping Module Reference

Functions/Subroutines

- subroutine, public [calculateovertopping](#) (geometry, load, modelFactors, overtopping, succes, errorMessage)

calculateOvertopping: calculate the overtopping

- subroutine, public [calculateovertoppingsection](#) (geometry, h, Hm0, Tm_10, L0, gammaBeta_z, gammaBeta_o, modelFactors, overtopping, succes, errorMessage)

calculateOvertoppingSection: calculate the overtopping for a section

- subroutine, public [calculatewaveovertopping](#) (geometry, h, Hm0, Tm_10, z2, gammaBeta_o, modelFactors, Qo, succes, errorMessage)

calculateWaveOvertopping: calculate wave overtopping

- subroutine [calculateovertoppingnegativefreeboard](#) (load, geometry, overtopping, succes, errorMessage)

calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard

- subroutine, public [interpolateresultssections](#) (geometry, L0, NwideBerms, overtoppingB, overtoppingF, overtopping, succes, errorMessage)

interpolateResultsSections: interpolate results for split cross sections

- subroutine, public [checkinputdata](#) (geometry, load, modelFactors, succes, errorMessage)

checkInputdata: check the input data

- subroutine, public [checkmodelfactors](#) (modelFactors, succes, errorMessage)

checkModelFactors: check the input data

- subroutine, public [convertovertoppinginput](#) (modelFactors, success, errorMessage)

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

4.5.1 Function/Subroutine Documentation

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

calculateOvertopping: calculate the overtopping

Parameters

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

Definition at line 36 of file mainModuleRTOOvertopping.f90.

[illegible]

```

graph LR
    A["mainmodulertoovtopping  
::calculateovertopping"] --> B["zfunctionswtiovertopping  
::calculateqorto"]
    B --> C["dllovertopping::calculateqof"]
    C --> D["dllovertopping::calculateqo"]

```

calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard

| | | |
|---------|---------------------|------------------------------------|
| 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 |

```

4.5.1.3 subroutine, public mainmodulertovertopping::calculateovertoppingsection ( type (tpgeometry), intent(in) geometry,
real(wp), intent(in) h, real(wp), intent(in) Hm0, real(wp), intent(in) Tm_10, real(wp), intent(in) L0, real(wp), intent(inout)
gammaBeta_z, real(wp), intent(inout) gammaBeta_o, type (tpovertoppinginput), intent(in) modelFactors, type
(tpovertopping), intent(out) overtopping, logical, intent(out) succes, character(len=*), intent(out) errorMessage )

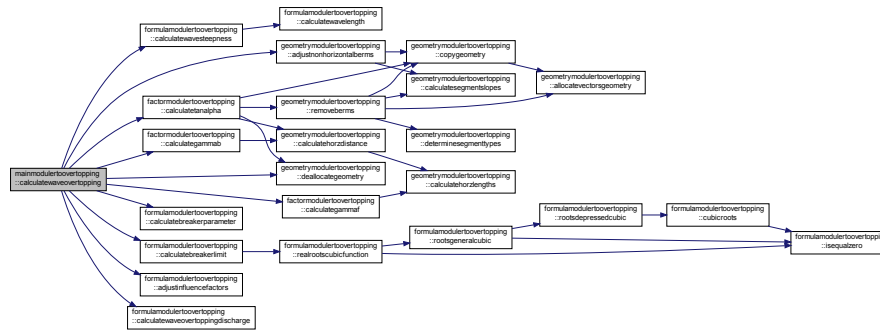
```

Parameters

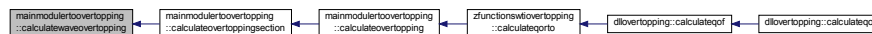
| | | |
|-----|---------------------|-----------------------------------------|
| out | <i>qo</i> | wave overtopping discharge (m3/m per s) |
| out | <i>succes</i> | flag for succes |
| out | <i>errormessage</i> | error message |

Definition at line 391 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.5 subroutine, public mainmodulertoovertopping::checkinputdata (type (tpgeometry), intent(in) *geometry*, type (tpload), intent(in) *load*, type (tpovertoppinginput), intent(in) *modelFactors*, logical, intent(out) *succes*, character(len=*) , intent(out) *errorMessage*)

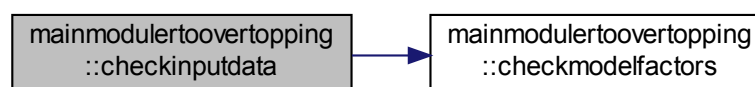
checkInputdata: check the input data

Parameters

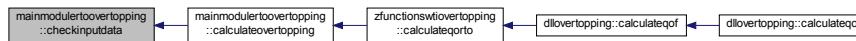
| | | |
|-----|---------------------|--------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in | <i>load</i> | structure with load parameters |
| in | <i>modelfactors</i> | structure with model factors |
| out | <i>succes</i> | flag for succes |
| out | <i>errormessage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

Definition at line 595 of file mainModuleRTOovertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.6 subroutine, public mainmodulertovertopping::checkmodelFactors (type (tpovertoppinginput), intent(in) *modelFactors*, logical, intent(out) *succes*, character(len=*), intent(out) *errorMessage*)

checkModelFactors: check the input data

Parameters

| | | |
|-----|---------------------|------------------------------|
| in | <i>modelFactors</i> | structure with model factors |
| out | <i>succes</i> | flag for succes |
| out | <i>errorMessage</i> | error message |

Definition at line 649 of file mainModuleRTOovertopping.f90.

Here is the caller graph for this function:



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

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

Parameters

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

Definition at line 744 of file mainModuleRTOovertopping.f90.

Here is the caller graph for this function:



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

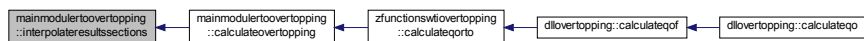
interpolateResultsSections: interpolate results for split cross sections

Parameters

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

Definition at line 514 of file mainModuleRTOovertopping.f90.

Here is the caller graph for this function:



4.6 modulelogging Module Reference

Data Types

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

Variables

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

4.6.1 Variable Documentation

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

copy of argument logging

Definition at line 21 of file ModuleLogging.f90.

4.6.1.2 integer, parameter `modulelogging::maxfilenamlength` = 256

maximum length of filename

Definition at line 13 of file ModuleLogging.f90.

4.7 overtoppinginterface Module Reference

Data Types

- type `overtoppinggeometrytype`

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

Variables

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

4.7.1 Variable Documentation

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

Model factor critical overtopping.

Definition at line 17 of file overtoppingInterface.f90.

4.8 typedefinitionsrtoovertopping Module Reference

Data Types

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

Variables

- real(wp), parameter [xdiff_min](#) = 2.0d-2
minimal value distance between x-coordinates (m)
- real(wp), parameter [margindiff](#) = 1.0d-14
margin for minimal distance (m)
- real(wp), parameter [berm_min](#) = 0.0d0
minimal value gradient berm segment
- real(wp), parameter [berm_max](#) = 1.0d0/15
maximal value gradient berm segment
- real(wp), parameter [slope_min](#) = 1.0d0/8
minimal value gradient slope segment
- real(wp), parameter [slope_max](#) = 1.0d0
maximal value gradient slope segment
- real(wp), parameter [margingrad](#) = 0.0025d0
margin for minimal and maximal gradients
- real(wp), parameter [rfactor_min](#) = 0.5d0
minimal value roughness factor dike segments
- real(wp), parameter [rfactor_max](#) = 1.0d0
maximal value roughness factor dike segments
- real(wp), parameter [mz2_min](#) = 0.0d0

- minimal value model factor of 2% runup height*
- real(wp), parameter `mz2_max` = huge(mz2_max)
- maximal value model factor of 2% runup height*
- real(wp), parameter `frunup1_min` = 0.0d0
- minimal value model factor 1 for wave run-up*
- real(wp), parameter `frunup1_max` = huge(fRunup1_max)
- maximal value model factor 1 for wave run-up*
- real(wp), parameter `frunup2_min` = 0.0d0
- minimal value model factor 2 for wave run-up*
- real(wp), parameter `frunup2_max` = huge(fRunup2_max)
- maximal value model factor 2 for wave run-up*
- real(wp), parameter `frunup3_min` = 0.0d0
- minimal value model factor 3 for wave run-up*
- real(wp), parameter `frunup3_max` = huge(fRunup3_max)
- maximal value model factor 3 for wave run-up*
- real(wp), parameter `fb_min` = 0.0d0
- minimal value model factor for breaking waves*
- real(wp), parameter `fb_max` = huge(fb_max)
- maximal value model factor for breaking waves*
- real(wp), parameter `fn_min` = 0.0d0
- minimal value model factor for non-breaking waves*
- real(wp), parameter `fn_max` = huge(fn_max)
- maximal value model factor for non-breaking waves*
- real(wp), parameter `fs_min` = 0.0d0
- minimal value model factor for shallow waves*
- real(wp), parameter `fs_max` = huge(fS_max)
- maximal value model factor for shallow waves*
- real(wp), parameter `foreshore_min` = 0.3d0
- minimal value reduction factor foreshore*
- real(wp), parameter `foreshore_max` = 1.0d0
- maximal value reduction factor foreshore*
- integer, parameter `z2_iter_max1` = 49
- maximal number of iterations for calculation z2 part 1*
- integer, parameter `z2_iter_max2` = 70
- maximal number of iterations for calculation z2 part 1 & 2*
- real(wp), parameter `z2_margin` = 0.001d0
- margin for convergence criterium calculation z2*

4.8.1 Variable Documentation

4.8.1.1 real(wp), parameter typedefinitionsrtovertopping::berm_max = 1.0d0/15

maximal value gradient berm segment

Definition at line 67 of file typeDefinitionsRTOovertopping.f90.

4.8.1.2 real(wp), parameter typedefinitionsrtovertopping::berm_min = 0.0d0

minimal value gradient berm segment

Definition at line 66 of file typeDefinitionsRTOovertopping.f90.

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

maximal value model factor for breaking waves

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

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

minimal value model factor for breaking waves

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

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

maximal value model factor for non-breaking waves

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

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

minimal value model factor for non-breaking waves

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

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

maximal value reduction factor foreshore

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

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

minimal value reduction factor foreshore

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

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

maximal value model factor 1 for wave run-up

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

4.8.1.10 `real(wp), parameter typeDefinitionsRTOovertopping::frunup1_min = 0.0d0`

minimal value model factor 1 for wave run-up

Definition at line 75 of file `typeDefinitionsRTOovertopping.f90`.

4.8.1.11 `real(wp), parameter typeDefinitionsRTOovertopping::frunup2_max = huge(fRunup2_max)`

maximal value model factor 2 for wave run-up

Definition at line 78 of file `typeDefinitionsRTOovertopping.f90`.

4.8.1.12 `real(wp), parameter typedefinitionsrtooverlapping::frunup2_min = 0.0d0`

minimal value model factor 2 for wave run-up

Definition at line 77 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.13 `real(wp), parameter typedefinitionsrtooverlapping::frunup3_max = huge(fRunup3_max)`

maximal value model factor 3 for wave run-up

Definition at line 80 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.14 `real(wp), parameter typedefinitionsrtooverlapping::frunup3_min = 0.0d0`

minimal value model factor 3 for wave run-up

Definition at line 79 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.15 `real(wp), parameter typedefinitionsrtooverlapping::fs_max = huge(fS_max)`

maximal value model factor for shallow waves

Definition at line 86 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.16 `real(wp), parameter typedefinitionsrtooverlapping::fs_min = 0.0d0`

minimal value model factor for shallow waves

Definition at line 85 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.17 `real(wp), parameter typedefinitionsrtooverlapping::margindiff = 1.0d-14`

margin for minimal distance (m)

Definition at line 65 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.18 `real(wp), parameter typedefinitionsrtooverlapping::margingrad = 0.0025d0`

margin for minimal and maximal gradients

Definition at line 70 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.19 `real(wp), parameter typedefinitionsrtooverlapping::mz2_max = huge(mz2_max)`

maximal value model factor of 2% runup height

Definition at line 74 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.20 `real(wp), parameter typedefinitionsrtooverlapping::mz2_min = 0.0d0`

minimal value model factor of 2% runup height

Definition at line 73 of file typeDefinitionsRTOoverlapping.f90.

4.8.1.21 `real(wp), parameter typeDefinitionsRTOovertopping::rfactor_max = 1.0d0`

maximal value roughness factor dike segments

Definition at line 72 of file `typeDefinitionsRTOovertopping.f90`.

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

minimal value roughness factor dike segments

Definition at line 71 of file `typeDefinitionsRTOovertopping.f90`.

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

maximal value gradient slope segment

Definition at line 69 of file `typeDefinitionsRTOovertopping.f90`.

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

minimal value gradient slope segment

Definition at line 68 of file `typeDefinitionsRTOovertopping.f90`.

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

minimal value distance between x-coordinates (m)

Definition at line 64 of file `typeDefinitionsRTOovertopping.f90`.

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

maximal number of iterations for calculation z2 part 1

Definition at line 89 of file `typeDefinitionsRTOovertopping.f90`.

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

maximal number of iterations for calculation z2 part 1 & 2

Definition at line 90 of file `typeDefinitionsRTOovertopping.f90`.

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

margin for convergence criterium calculation z2

Definition at line 91 of file `typeDefinitionsRTOovertopping.f90`.

4.9 waverunup Module Reference

Functions/Subroutines

- subroutine, public [iterationwaverunup](#) (geometry, h, Hm0, Tm_10, gammaBeta_z, modelFactors, z2, succes, errorMessage)

iterationWaveRunup: iteration for the wave runup

- real(kind=wp) function [innercalculation](#) (geometry, h, Hm0, gammaBeta_z, modelFactors, z2, s0, geometry↔ FlatBerms, succes, errorMessage)
- real(kind=wp) function [determinestartingvalue](#) (i, relaxationFactor, z2_start, z2_end, Hm0)
- integer function [findsmallestresidu](#) (z2_start, z2_end, n)
- subroutine [convergedwithresidu](#) (z2_start, z2_end)

4.9.1 Function/Subroutine Documentation

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

Definition at line 317 of file waveRunup.f90.

Here is the call graph for this function:



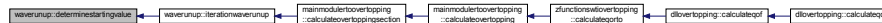
Here is the caller graph for this function:



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

Definition at line 266 of file waveRunup.f90.

Here is the caller graph for this function:



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

Definition at line 288 of file waveRunup.f90.

Here is the caller graph for this function:



4.9.1.4 `real(kind=wp) function waverunup::innercalculation (type (tpgeometry), intent(in) geometry, real(wp), intent(in) h,
real(wp), intent(in) Hm0, real(wp), intent(inout) gammaBeta_z, type (tpovertoppinginput), intent(in) modelFactors,
real(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]`

Parameters

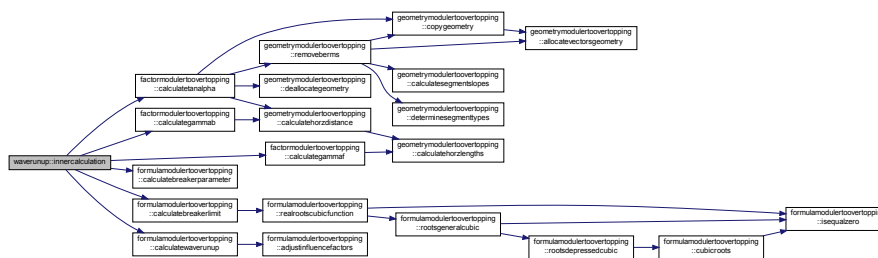
| | | |
|---------|---------------------------|----------------------------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in | <i>h</i> | local water level (m+NAP) |
| in | <i>hm0</i> | significant wave height (m) |
| in, out | <i>gammabeta_z</i> | influence factor 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 162 of file waveRunup.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

iterationWaveRunup: iteration for the wave runup

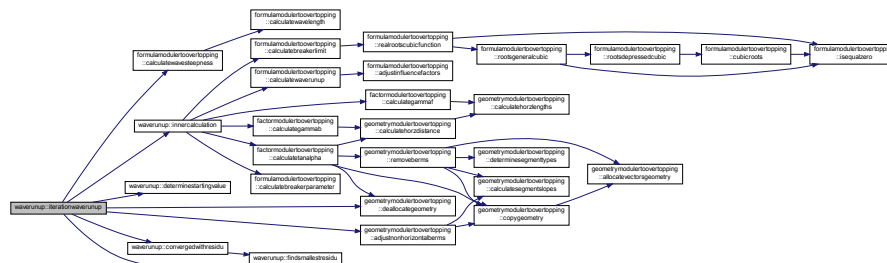
Parameters

| | | |
|---------|--------------------|----------------------------------------------|
| in | <i>geometry</i> | structure with geometry data |
| in | <i>h</i> | local water level (m+NAP) |
| in | <i>hm0</i> | significant wave height (m) |
| in | <i>tm_10</i> | spectral wave period (s) |
| in, out | <i>gammabeta_z</i> | influence factor angle wave attack 2% run-up |

| | | |
|-----|---------------------|------------------------------|
| in | <i>modelfactors</i> | structure with model factors |
| out | <i>z2</i> | 2% wave run-up (m) |
| out | <i>succes</i> | flag for succes |
| out | <i>errormessage</i> | error message |

Definition at line 33 of file waveRunup.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



4.10 zfunctionswtiovertopping Module Reference

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

Functions/Subroutines

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

4.10.1 Detailed Description

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

4.10.2 Function/Subroutine Documentation

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

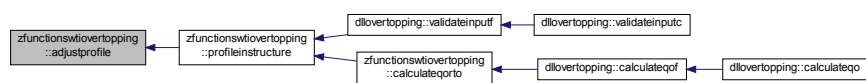
Subroutine adjust the profile due to a desired dike height.

Parameters

| | | |
|-----|-------------------------|---------------------------------------------------|
| in | <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 109 of file zFunctionsWTIOvertopping.f90.

Here is the caller graph for this function:



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

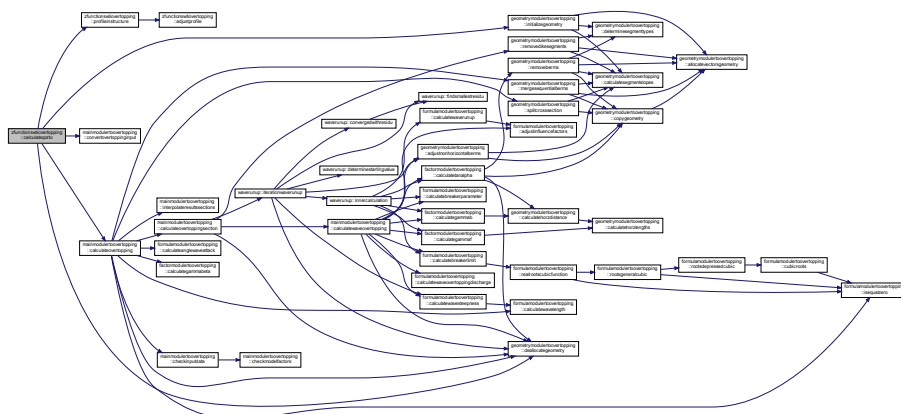
Subroutine to calculate the overtopping discharge with the RTO-overtopping dll.

Parameters

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

Definition at line 32 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

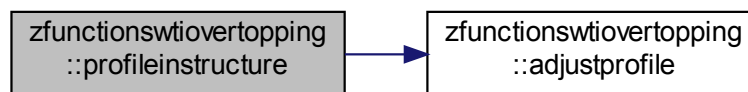
Subroutine to fill the profile in a structure and call the adjustment function of the profile due to a desired dike height.

Parameters

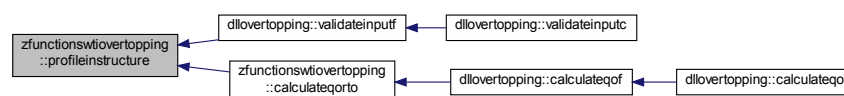
| | | |
|-----|-------------------------|---------------------------------------------------|
| in | <i>nrcoordinates</i> | number of coordinates of the profile |
| in | <i>xcoordinates</i> | vector with x-coordinates of the profile |
| in | <i>ycoordinates</i> | vector with y-coordinates of the profile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

Definition at line 84 of file zFunctionsWTIOvertopping.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

Parameters

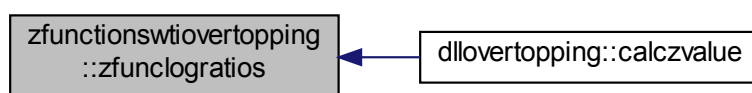
| | | |
|-----|---------------------|---------------------------------------------|
| in | <i>qo</i> | computed overtopping discharge |
| in | <i>qc</i> | Critical overtopping discharge |
| in | <i>mgo</i> | Model factor computed overtopping discharge |
| in | <i>mqc</i> | Model factor Critical overtopping 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 198 of file zFunctionsWTIOvertopping.f90.

Here is the caller graph for this function:

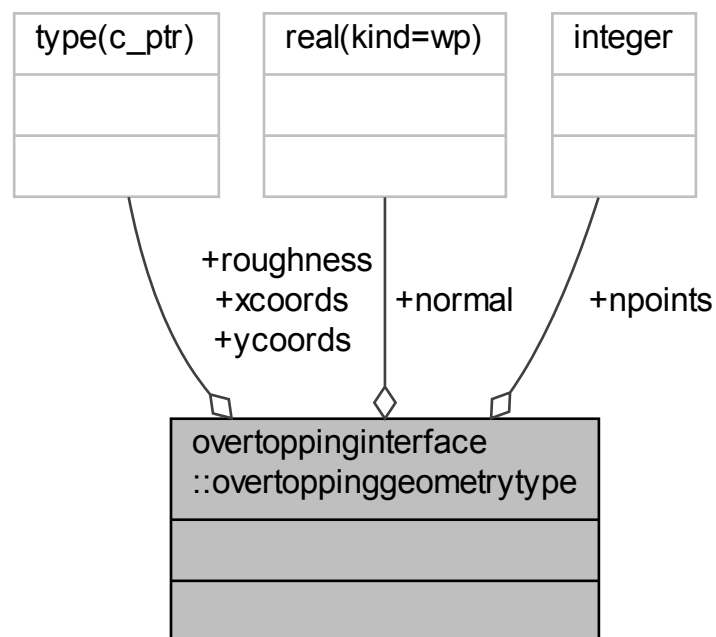


Chapter 5

Data Type Documentation

5.1 overtoppinginterface::overtoppinggeometrytype Type Reference

Collaboration diagram for overtoppinginterface::overtoppinggeometrytype:



Public Attributes

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

5.1.1 Detailed Description

Definition at line 25 of file overtoppingInterface.f90.

5.1.2 Member Data Documentation

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

Definition at line 26 of file overtoppingInterface.f90.

5.1.2.2 `integer overtoppinginterface::overtoppinggeometrytype::npoints`

Definition at line 27 of file overtoppingInterface.f90.

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

Definition at line 30 of file overtoppingInterface.f90.

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

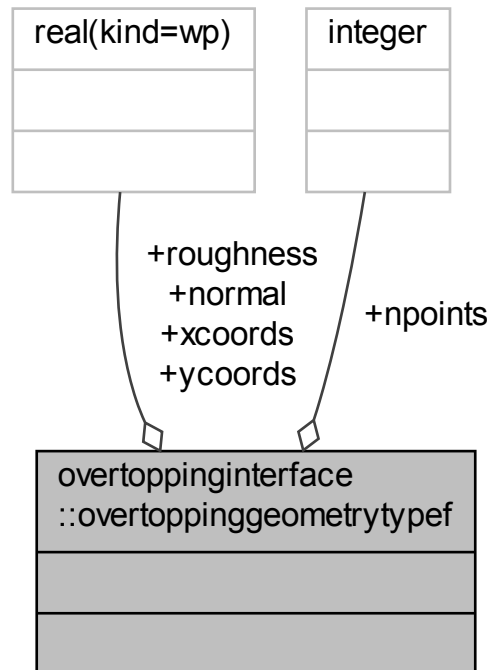
Definition at line 28 of file overtoppingInterface.f90.

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

Definition at line 29 of file overtoppingInterface.f90.

5.2 overtoppinginterface::overtoppinggeometrytypef Type Reference

Collaboration diagram for overtoppinginterface::overtoppinggeometrytypef:



Public Attributes

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

5.2.1 Detailed Description

Definition at line 33 of file overtoppingInterface.f90.

5.2.2 Member Data Documentation

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

Definition at line 34 of file overtoppingInterface.f90.

5.2.2.2 integer overtoppinginterface::overtoppinggeometrytypef::npoints

Definition at line 35 of file overtoppingInterface.f90.

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

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

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

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

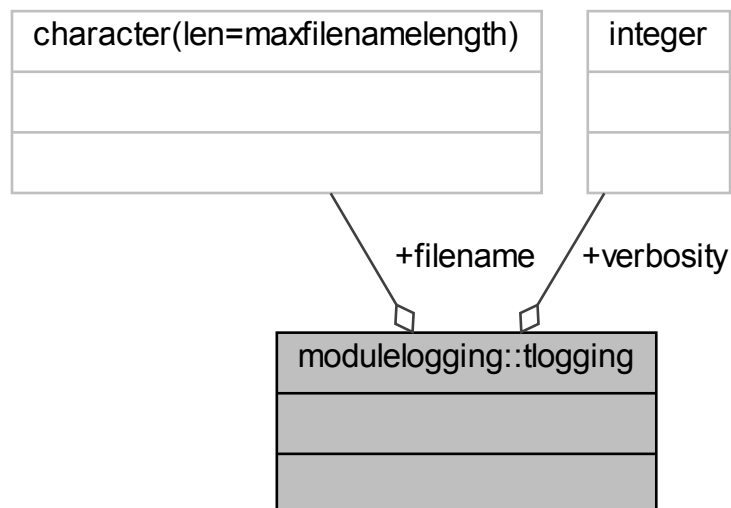
5.2.2.5 `real(kind=wp), dimension(:), pointer overtoppinginterface::overtoppinggeometrytypef::ycoords`

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

5.3 `modulelogging::tlogging` Type Reference

TLogging: structure for steering the logging.

Collaboration diagram for `modulelogging::tlogging`:



Public Attributes

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

5.3.1 Detailed Description

TLogging: structure for steering the logging.

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

5.3.2 Member Data Documentation

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

filename of logging

Definition at line 18 of file ModuleLogging.f90.

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

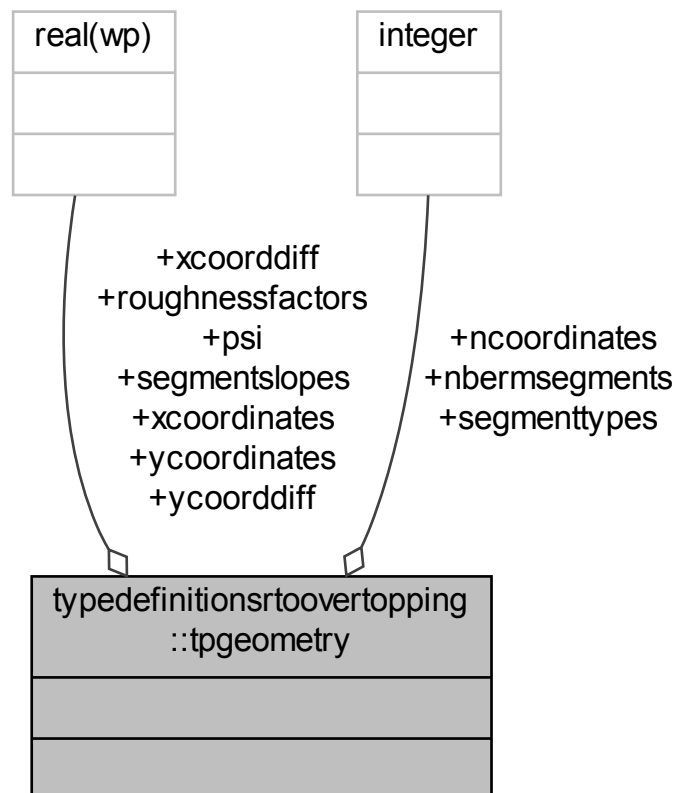
level of verbosity: one of verboseNone, verboseBasic, verboseDetailed, verboseDebugging

Definition at line 17 of file ModuleLogging.f90.

5.4 typedefinitionsrtoovertopping::tpgeometry Type Reference

tpGeometry: structure with geometry data

Collaboration diagram for typedefinitionsrtoovertopping::tpgeometry:



Public Attributes

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

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

5.4.1 Detailed Description

tpGeometry: structure with geometry data

Definition at line 18 of file typeDefinitionsRTOovertopping.f90.

5.4.2 Member Data Documentation

5.4.2.1 integer typedefinitionsrtooverlapping::tpgeometry::nbermsegments

number of berm segments

Definition at line 28 of file typeDefinitionsRTOovertopping.f90.

5.4.2.2 integer typedefinitionsrtooverlapping::tpgeometry::ncoordinates

number of coordinates cross section

Definition at line 20 of file typeDefinitionsRTOovertopping.f90.

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

dike normal (degrees)

Definition at line 19 of file typeDefinitionsRTOovertopping.f90.

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

vector with roughness factors cross section

Definition at line 23 of file typeDefinitionsRTOovertopping.f90.

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

vector with slopes dike segments

Definition at line 26 of file typeDefinitionsRTOover topping.f90.

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

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

Definition at line 27 of file typeDefinitionsRTOover topping.f90.

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

vector with differences in x-coordinates (m)

Definition at line 24 of file typeDefinitionsRTOover topping.f90.

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

vector with x-coordinates cross section (m)

Definition at line 21 of file typeDefinitionsRTOover topping.f90.

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

vector with differences in y-coordinates (m)

Definition at line 25 of file typeDefinitionsRTOover topping.f90.

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

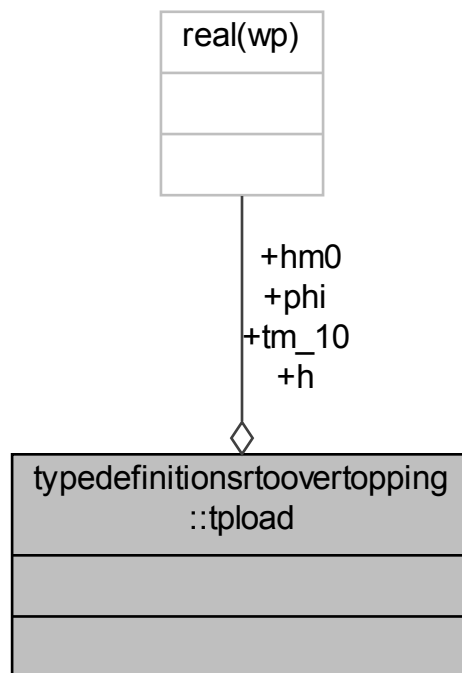
vector with y-coordinates cross section (m+NAP)

Definition at line 22 of file typeDefinitionsRTOover topping.f90.

5.5 typedefinitionsrtoover topping::tpload Type Reference

tpLoad: structure with load parameters

Collaboration diagram for `typedefinitionsrtooveropping::tpload`:



Public Attributes

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

5.5.1 Detailed Description

`tpLoad`: structure with load parameters

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

5.5.2 Member Data Documentation

5.5.2.1 `real(wp)` `typedefinitionsrtooveropping::tpload::h`

local water level (m+NAP)

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

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

significant wave height (m)

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

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

wave direction (degrees)

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

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

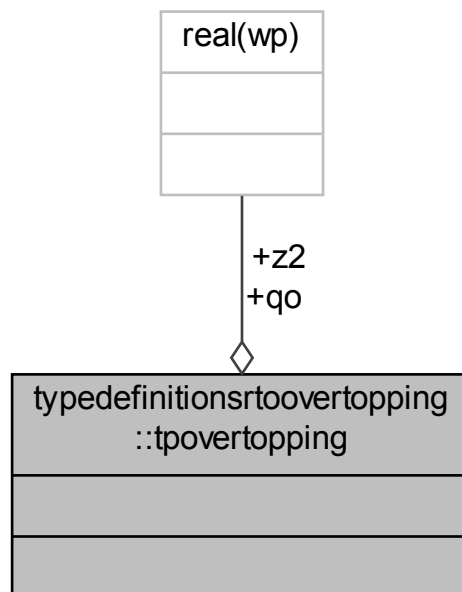
spectral wave period (s)

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

5.6 `typedefinitionsrtoovertopping::tpovertopping` Type Reference

`tpOvertopping`: structure with overtopping results

Collaboration diagram for `typedefinitionsrtoovertopping::tpovertopping`:



Public Attributes

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

wave overtopping discharge (m³/m per s)

5.6.1 Detailed Description

tpOvertopping: structure with overtopping results

Definition at line 56 of file typeDefinitionsRTOovertopping.f90.

5.6.2 Member Data Documentation

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

wave overtopping discharge (m³/m per s)

Definition at line 58 of file typeDefinitionsRTOovertopping.f90.

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

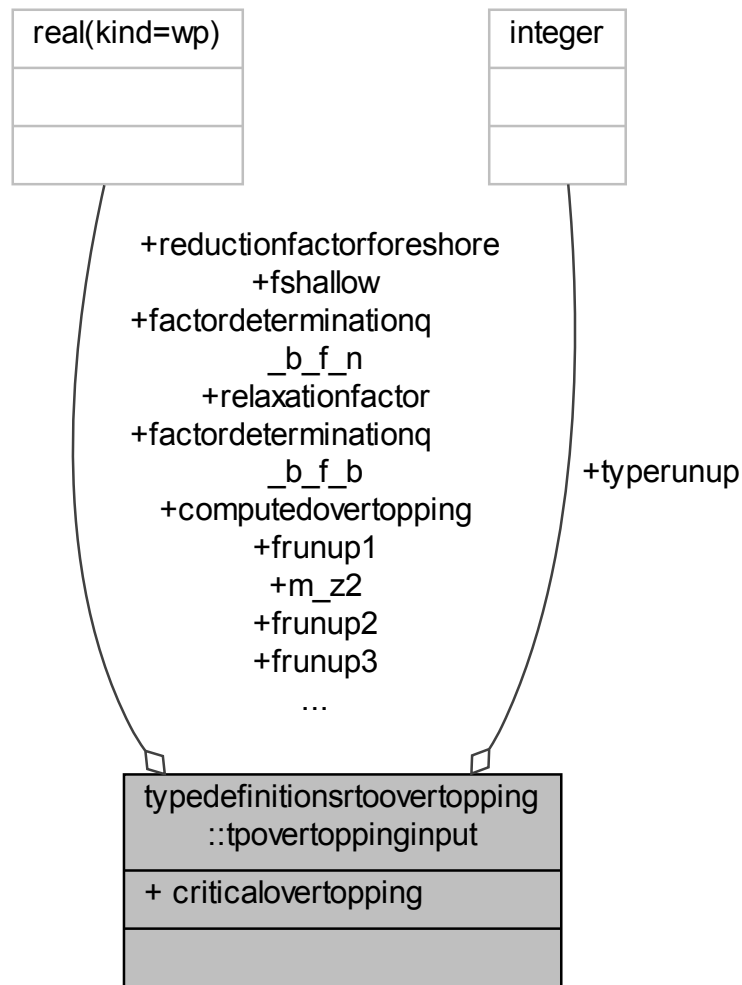
2% wave run-up (m)

Definition at line 57 of file typeDefinitionsRTOovertopping.f90.

5.7 `typedefinitionsrtooveropping::tpoveroppinginput` Type Reference

OvertoppingModelFactors: C-structure with model factors.

Collaboration diagram for typedefinitionsrtovertopping::tpovertoppinginput:



Public Attributes

- `real(kind=wp)` [factordeterminationq_b_f_n](#)
model factor for non-breaking waves
- `real(kind=wp)` [factordeterminationq_b_f_b](#)
model factor for breaking waves
- `real(kind=wp)` [m_z2](#)
model factor describing the uncertainty of 2% runup height
- `real(kind=wp)` [frunup1](#)
model factor 1 for wave run-up (for backwards compatability)
- `real(kind=wp)` [frunup2](#)
model factor 2 for wave run-up (idem)
- `real(kind=wp)` [frunup3](#)
model factor 3 for wave run-up (idem)
- `real(kind=wp)` [fshallow](#)

- model factor for shallow waves*
- `real(kind=wp) computedovertopping`
model factor computed overtopping
- `real(kind=wp) criticalovertopping`
model factor critical overtopping
- `integer typerunup`
0: fRunup1, 2 and 3 are given; 1: m_z2 is given
- `real(kind=wp) relaxationfactor`
relaxation factor iteration procedure wave runup
- `real(kind=wp) reductionfactorforeshore = 0.5_wp`
reduction factor foreshore

5.7.1 Detailed Description

OvertoppingModelFactors: C-structure with model factors.

Definition at line 40 of file typeDefinitionsRTOovertopping.f90.

5.7.2 Member Data Documentation

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

model factor computed overtopping

Definition at line 48 of file typeDefinitionsRTOovertopping.f90.

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

model factor critical overtopping

Definition at line 49 of file typeDefinitionsRTOovertopping.f90.

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

model factor for breaking waves

Definition at line 42 of file typeDefinitionsRTOovertopping.f90.

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

model factor for non-breaking waves

Definition at line 41 of file typeDefinitionsRTOovertopping.f90.

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

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

Definition at line 44 of file typeDefinitionsRTOovertopping.f90.

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

model factor 2 for wave run-up (idem)

Definition at line 45 of file typeDefinitionsRTOovertopping.f90.

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

model factor 3 for wave run-up (idem)

Definition at line 46 of file typeDefinitionsRTOovertopping.f90.

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

model factor for shallow waves

Definition at line 47 of file typeDefinitionsRTOovertopping.f90.

5.7.2.9 real(kind=wp) typedefinitionsrtoovertopping::tpovertoppinginput::m_z2

model factor describing the uncertainty of 2% runup height

Definition at line 43 of file typeDefinitionsRTOovertopping.f90.

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

reduction factor foreshore

Definition at line 52 of file typeDefinitionsRTOovertopping.f90.

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

relaxation factor iteration procedure wave runup

Definition at line 51 of file typeDefinitionsRTOovertopping.f90.

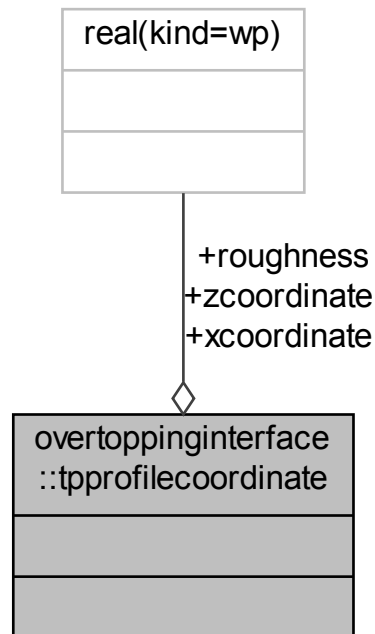
5.7.2.12 integer typedefinitionsrtoovertopping::tpovertoppinginput::typerunup

0: fRunup1, 2 and 3 are given; 1: m_z2 is given

Definition at line 50 of file typeDefinitionsRTOovertopping.f90.

5.8 overtoppinginterface::tpprofilecoordinate Type Reference

Collaboration diagram for overtoppinginterface::tpprofilecoordinate:



Public Attributes

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

5.8.1 Detailed Description

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

5.8.2 Member Data Documentation

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

Roughness of the area between two points.

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

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

X-coordinate foreland profile.

Definition at line 20 of file overtoppingInterface.f90.

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

Z-coordinate foreland profile.

Definition at line 21 of file overtoppingInterface.f90.

Chapter 6

File Documentation

6.1 dllOvertopping.f90 File Reference

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

Modules

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

Functions/Subroutines

- subroutine, public [dllovertopping::calculateqo](#) (load, geometryInput, dikeHeight, modelFactors, overtopping, success, errorText, verbosity, logFile)
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping Wrapper for calculateQoF: convert C-like input structures to Fortran input structures.
- subroutine, public [dllovertopping::calculateqof](#) (load, geometryF, dikeHeight, modelFactors, overtopping, success, errorText, logging)
Subroutine that calculates the discharge needed for the Z-function DikesOvertopping.
- subroutine, public [dllovertopping::calczvalue](#) (criticalOvertoppingRate, modelFactors, Qo, z, success, error↔ Message)
Subroutine that calculates the Z-function DikesOvertopping based on the discharge calculated with calculateQoF.
- subroutine, public [dllovertopping::validateinputc](#) (geometryInput, dikeHeight, modelFactors, success, error↔ Text)
Subroutine that validates the geometry Wrapper for ValidateInputF: convert C-like input structures to Fortran input structures.
- subroutine, public [dllovertopping::validateinputf](#) (geometryF, dikeHeight, modelFactors, success, errorText)
Subroutine that validates the geometry.
- 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:

- zFuncOvertopping

- calculateQo
- calculateQoF
- ValidateInputC
- ValidateInputF
- versionNumber

6.2 factorModuleRTOovertopping.f90 File Reference

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

Modules

- module [factormodulertooveropping](#)

Functions/Subroutines

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

6.2.1 Detailed Description

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

6.3 formulaModuleRTOovertopping.f90 File Reference

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

Modules

- module [formulamodulertooveropping](#)

Functions/Subroutines

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

6.3.1 Detailed Description

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

6.4 geometryModuleRTOovertopping.f90 File Reference

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

Modules

- module [geometrymodulertooverlapping](#)

Functions/Subroutines

- subroutine, public [geometrymodulertooverlapping::checkcrosssection](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, succes, errorMessage)
checkCrossSection: check cross section
- subroutine, public [geometrymodulertooverlapping::initializegeometry](#) (psi, nCoordinates, xCoordinates, yCoordinates, roughnessFactors, geometry, succes, errorMessage)
initializeGeometry: initialize the geometry
- subroutine, public [geometrymodulertooverlapping::allocatevectorsgeometry](#) (nCoordinates, geometry)
allocateVectorsGeometry: allocate the geometry vectors
- subroutine, public [geometrymodulertooverlapping::deallocategemetry](#) (geometry)
deallocateGeometry: deallocate the geometry vectors
- subroutine, public [geometrymodulertooverlapping::calculatesegmentslopes](#) (geometry, succes, errorMessage)
calculateSegmentSlopes: calculate the segment slopes
- subroutine, public [geometrymodulertooverlapping::determinesegmenttypes](#) (geometry)
determineSegmentTypes: determine the segment types
- subroutine, public [geometrymodulertooverlapping::copygeometry](#) (geometry, geometryCopy)
copyGeometry: copy a geometry structure
- subroutine, public [geometrymodulertooverlapping::isequalgeometry](#) (geometry1, geometry2, succes, errorMessage)
isEqualGeometry: are two geometries equal
- subroutine, public [geometrymodulertooverlapping::mergesquentialberms](#) (geometry, geometryMergedBerms, succes, errorMessage)
mergeSequentialBerms: merge sequential berms
- subroutine, public [geometrymodulertooverlapping::adjustnonhorizontalberms](#) (geometry, geometryFlatBerms, succes, errorMessage)
adjustNonHorizontalBerms: adjust non-horizontal berms
- subroutine, public [geometrymodulertooverlapping::removeberms](#) (geometry, geometryNoBerms, succes, errorMessage)
removeBerms: remove berms
- subroutine, public [geometrymodulertooverlapping::removedikesgments](#) (geometry, index, geometryAdjusted, succes, errorMessage)
removeDikeSegments: remove dike segments
- subroutine, public [geometrymodulertooverlapping::splitcrosssection](#) (geometry, L0, NwideBerms, geometrysectionB, geometrysectionF, succes, errorMessage)
splitCrossSection: split a cross section
- subroutine, public [geometrymodulertooverlapping::calculatehorzlengths](#) (geometry, yLower, yUpper, horzLengths, succes, errorMessage)
calculateHorzLengths: calculate horizontal lengths
- subroutine, public [geometrymodulertooverlapping::calculatehorzdistance](#) (geometry, yLower, yUpper, dx, succes, errorMessage)
calculateHorzDistance: calculate horizontal distance
- subroutine, public [geometrymodulertooverlapping::writecrosssection](#) (geometry, geometryName)
writeCrossSection: write a cross section

6.4.1 Detailed Description

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

6.5 mainModuleRTOovertopping.f90 File Reference

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

Modules

- module [mainmodulertooverlapping](#)

Functions/Subroutines

- subroutine, public [mainmodulertooverlapping::calculateovertopping](#) (geometry, load, modelFactors, overtopping, succes, errorMessage)
calculateOvertopping: calculate the overtopping
- subroutine, public [mainmodulertooverlapping::calculateovertoppingsection](#) (geometry, h, Hm0, Tm_10, L0, gammaBeta_z, gammaBeta_o, modelFactors, overtopping, succes, errorMessage)
calculateOvertoppingSection: calculate the overtopping for a section
- subroutine, public [mainmodulertooverlapping::calculatewaveovertopping](#) (geometry, h, Hm0, Tm_10, z2, gammaBeta_o, modelFactors, Qo, succes, errorMessage)
calculateWaveOvertopping: calculate wave overtopping
- subroutine [mainmodulertooverlapping::calculateovertoppingnegativefreeboard](#) (load, geometry, overtopping, succes, errorMessage)
calculateOvertoppingNegativeFreeboard: calculate overtopping in case of negative freeboard
- subroutine, public [mainmodulertooverlapping::interpolateresultssections](#) (geometry, L0, NwideBerms, overtoppingB, overtoppingF, overtopping, succes, errorMessage)
interpolateResultsSections: interpolate results for split cross sections
- subroutine, public [mainmodulertooverlapping::checkinputdata](#) (geometry, load, modelFactors, succes, errorMessage)
checkInputdata: check the input data
- subroutine, public [mainmodulertooverlapping::checkmodelfactors](#) (modelFactors, succes, errorMessage)
checkModelFactors: check the input data
- subroutine, public [mainmodulertooverlapping::converttooverlappinginput](#) (modelFactors, success, errorMessage)
convertOvertoppingInput: convert the model factors from C-like to Fortran

6.5.1 Detailed Description

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

6.6 ModuleLogging.f90 File Reference

Module for steering the extra logging.

Data Types

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

Modules

- module [modulelogging](#)

Variables

- integer, parameter `modulelogging::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 `overtoppingInterface.f90` File Reference

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

Data Types

- type `overtoppinginterface::tpprofilecoordinate`
- type `overtoppinginterface::overtoppinggeometrytype`
- type `overtoppinginterface::overtoppinggeometrytypef`

Modules

- module `overtoppinginterface`

Variables

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

6.7.1 Detailed Description

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

6.8 `typeDefinitionsRTOovertopping.f90` File Reference

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

Data Types

- type `typedefinitionsrtooveropping::tpgeometry`
tpGeometry: structure with geometry data
- type `typedefinitionsrtooveropping::tpload`
tpLoad: structure with load parameters
- type `typedefinitionsrtooveropping::tpovertoppinginput`
OvertoppingModelFactors: C-structure with model factors.
- type `typedefinitionsrtooveropping::tpovertopping`
tpOvertopping: structure with overtopping results

Modules

- module [typedefinitionsrtooverlapping](#)

Variables

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

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

6.8.1 Detailed Description

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

6.9 waveRunup.f90 File Reference

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

Modules

- module [waverunup](#)

Functions/Subroutines

- subroutine, public [waverunup::iterationwaverunup](#) (geometry, h, Hm0, Tm_10, gammaBeta_z, modelFactors, z2, succes, errorMessage)
iterationWaveRunup: iteration for the wave runup
- real(kind=wp) function [waverunup::innercalculation](#) (geometry, h, Hm0, gammaBeta_z, modelFactors, z2, s0, geometryFlatBerms, succes, errorMessage)
- real(kind=wp) function [waverunup::determinestartingvalue](#) (i, relaxationFactor, z2_start, z2_end, Hm0)
- integer function [waverunup::findsmallestresidu](#) (z2_start, z2_end, n)
- subroutine [waverunup::convergedwithresidu](#) (z2_start, z2_end)

6.9.1 Detailed Description

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

6.10 zFunctionsWTIOvertopping.f90 File Reference

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

Modules

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

Functions/Subroutines

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

6.10.1 Detailed Description

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

Index

- adjustinfluencefactors
 - formulamodulertoovertopping, [16](#)
- adjustnonhorizontalberms
 - geometrymodulertoovertopping, [23](#)
- adjustprofile
 - zfunctionswtioovertopping, [48](#)
- allocatevectorsgeometry
 - geometrymodulertoovertopping, [24](#)
- berm_max
 - typedefinitionsrtoovertopping, [41](#)
- berm_min
 - typedefinitionsrtoovertopping, [41](#)
- calculateanglewaveattack
 - formulamodulertoovertopping, [16](#)
- calculatebreakerlimit
 - formulamodulertoovertopping, [16](#)
- calculatebreakerparameter
 - formulamodulertoovertopping, [17](#)
- calculategammab
 - factormodulertoovertopping, [12](#)
- calculategammabeta
 - factormodulertoovertopping, [13](#)
- calculategammaf
 - factormodulertoovertopping, [13](#)
- calculatehorzdistance
 - geometrymodulertoovertopping, [24](#)
- calculatehorzlenghts
 - geometrymodulertoovertopping, [25](#)
- calculateovertopping
 - mainmodulertoovertopping, [34](#)
- calculateovertoppingnegativefreeboard
 - mainmodulertoovertopping, [35](#)
- calculateovertoppingsection
 - mainmodulertoovertopping, [35](#)
- calculateqo
 - dllovertopping, [7](#)
- calculateqof
 - dllovertopping, [8](#)
- calculateqorto
 - zfunctionswtioovertopping, [50](#)
- calculatesegmentslopes
 - geometrymodulertoovertopping, [25](#)
- calculatetanalpha
 - factormodulertoovertopping, [14](#)
- calculatewavelength
 - formulamodulertoovertopping, [17](#)
- calculatewaveovertopping
 - mainmodulertoovertopping, [36](#)
- calculatewaveovertoppingdischarge
 - formulamodulertoovertopping, [18](#)
- calculatewaverunup
 - formulamodulertoovertopping, [18](#)
- calculatewavesteepness
 - formulamodulertoovertopping, [19](#)
- calczvalue
 - dllovertopping, [9](#)
- checkcrosssection
 - geometrymodulertoovertopping, [26](#)
- checkinputdata
 - mainmodulertoovertopping, [37](#)
- checkmodelfactors
 - mainmodulertoovertopping, [38](#)
- computedovertopping
 - typedefinitionsrtoovertopping::tpovertoppinginput, [66](#)
- convergedwithresidu
 - waverunup, [45](#)
- convertovertoppinginput
 - mainmodulertoovertopping, [38](#)
- copygeometry
 - geometrymodulertoovertopping, [27](#)
- criticalovertopping
 - typedefinitionsrtoovertopping::tpovertoppinginput, [66](#)
- cubicroots
 - formulamodulertoovertopping, [19](#)
- currentlogging
 - modulelogging, [39](#)
- deallocategeometry
 - geometrymodulertoovertopping, [27](#)
- determinesegmenttypes
 - geometrymodulertoovertopping, [28](#)
- determinestartingvalue
 - waverunup, [45](#)
- dllovertopping.f90, [71](#)
- dllovertopping, [7](#)
 - calculateqo, [7](#)
 - calculateqof, [8](#)
 - calczvalue, [9](#)
 - geometry_c_f, [10](#)
 - validateinputc, [10](#)
 - validateinputf, [11](#)
 - versionnumber, [12](#)
- factorModuleRTOovertopping.f90, [72](#)
- factordeterminationq_b_f_b

- typedefinitionsrtooverlapping::tpoverlappinginput, 66
- factordeterminationq_b_f_n
 - typedefinitionsrtooverlapping::tpoverlappinginput, 66
- factormodulertooverlapping, 12
 - calculategammab, 12
 - calculategammabeta, 13
 - calculategammaf, 13
 - calculatetanalpha, 14
- fb_max
 - typedefinitionsrtooverlapping, 41
- fb_min
 - typedefinitionsrtooverlapping, 42
- filename
 - modulelogging::tlogging, 59
- findsmallestresidu
 - waverunup, 45
- fn_max
 - typedefinitionsrtooverlapping, 42
- fn_min
 - typedefinitionsrtooverlapping, 42
- foreshore_max
 - typedefinitionsrtooverlapping, 42
- foreshore_min
 - typedefinitionsrtooverlapping, 42
- formulaModuleRTOoverlapping.f90, 72
- formulamodulertooverlapping, 15
 - adjustinfluencefactors, 16
 - calculateanglewaveattack, 16
 - calculatebreakerlimit, 16
 - calculatebreakerparameter, 17
 - calculatewavelength, 17
 - calculatewaveoverlappingdischarge, 18
 - calculatewaverunup, 18
 - calculatewavesteepness, 19
 - cubicroots, 19
 - isequalreal, 20
 - isequalzero, 20
 - realrootscubicfunction, 21
 - rootsdepressedcubic, 21
 - rootsgeneralcubic, 22
- frunup1
 - typedefinitionsrtooverlapping::tpoverlappinginput, 66
- frunup1_max
 - typedefinitionsrtooverlapping, 42
- frunup1_min
 - typedefinitionsrtooverlapping, 42
- frunup2
 - typedefinitionsrtooverlapping::tpoverlappinginput, 66
- frunup2_max
 - typedefinitionsrtooverlapping, 42
- frunup2_min
 - typedefinitionsrtooverlapping, 42
- frunup3
 - typedefinitionsrtooverlapping::tpoverlappinginput, 66
- frunup3_max
 - typedefinitionsrtooverlapping, 43
- frunup3_min
 - typedefinitionsrtooverlapping, 43
- fs_max
 - typedefinitionsrtooverlapping, 43
- fs_min
 - typedefinitionsrtooverlapping, 43
- fshallow
 - typedefinitionsrtooverlapping::tpoverlappinginput, 67
- geometry_c_f
 - dloverlapping, 10
- geometryModuleRTOoverlapping.f90, 73
- geometrymodulertooverlapping, 22
 - adjustnonhorizontalberms, 23
 - allocatevectorsgeometry, 24
 - calculatehorzdistance, 24
 - calculatehorzlengths, 25
 - calculatesegmentslopes, 25
 - checkcrosssection, 26
 - copygeometry, 27
 - deallocategeometry, 27
 - determinesegmenttypes, 28
 - initializegeometry, 28
 - isequalgeometry, 29
 - mergesequentialberms, 29
 - removeberms, 31
 - removedikesegments, 32
 - splitcrosssection, 32
 - writecrosssection, 33
- h
 - typedefinitionsrtooverlapping::tpload, 62
- hm0
 - typedefinitionsrtooverlapping::tpload, 62
- initializegeometry
 - geometrymodulertooverlapping, 28
- innercalculation
 - waverunup, 45
- interpolateresultssections
 - mainmodulertooverlapping, 38
- isequalgeometry
 - geometrymodulertooverlapping, 29
- isequalreal
 - formulamodulertooverlapping, 20
- isequalzero
 - formulamodulertooverlapping, 20
- iterationwaverunup
 - waverunup, 47
- m_z2
 - typedefinitionsrtooverlapping::tpoverlappinginput, 67
- mainModuleRTOoverlapping.f90, 75

- mainmodulertoovertopping, [33](#)
 - calculateovertopping, [34](#)
 - calculateovertoppingnegativefreeboard, [35](#)
 - calculateovertoppingsection, [35](#)
 - calculatewaveovertopping, [36](#)
 - checkinputdata, [37](#)
 - checkmodelfactors, [38](#)
 - converttoovertoppinginput, [38](#)
 - interpolateresultssections, [38](#)
- margindiff
 - typedefinitionsrtoovertopping, [43](#)
- margingrad
 - typedefinitionsrtoovertopping, [43](#)
- maxfilenamelength
 - modulelogging, [39](#)
- mergesequentialberms
 - geometrymodulertoovertopping, [29](#)
- ModuleLogging.f90, [75](#)
- modulelogging, [39](#)
 - currentlogging, [39](#)
 - maxfilenamelength, [39](#)
- modulelogging::tlogging, [58](#)
 - filename, [59](#)
 - verbosity, [59](#)
- mz2_max
 - typedefinitionsrtoovertopping, [43](#)
- mz2_min
 - typedefinitionsrtoovertopping, [43](#)
- nbermssegments
 - typedefinitionsrtoovertopping::tpgeometry, [60](#)
- ncoordinates
 - typedefinitionsrtoovertopping::tpgeometry, [60](#)
- normal
 - overtoppinginterface::overtoppinggeometrytype, [56](#)
 - overtoppinginterface::overtoppinggeometrytypef, [57](#)
- npoints
 - overtoppinginterface::overtoppinggeometrytype, [56](#)
 - overtoppinginterface::overtoppinggeometrytypef, [57](#)
- overtoppingInterface.f90, [76](#)
- overtoppinginterface, [39](#)
 - varmodelfactorcriticalovertopping, [40](#)
- overtoppinginterface::overtoppinggeometrytype, [55](#)
 - normal, [56](#)
 - npoints, [56](#)
 - roughness, [56](#)
 - xcoords, [56](#)
 - ycoords, [56](#)
- overtoppinginterface::overtoppinggeometrytypef, [57](#)
 - normal, [57](#)
 - npoints, [57](#)
 - roughness, [57](#)
 - xcoords, [58](#)
 - ycoords, [58](#)
- overtoppinginterface::tpprofilecoordinate, [68](#)
 - roughness, [68](#)
 - xcoordinate, [68](#)
 - zcoordinate, [69](#)
- phi
 - typedefinitionsrtoovertopping::tpload, [63](#)
- profileinstructure
 - zfunctionswtioovertopping, [51](#)
- psi
 - typedefinitionsrtoovertopping::tpgeometry, [60](#)
- qo
 - typedefinitionsrtoovertopping::tpovertopping, [64](#)
- realrootscubicfunction
 - formulamodulertoovertopping, [21](#)
- reductionfactorforeshore
 - typedefinitionsrtoovertopping::tpovertoppinginput, [67](#)
- relaxationfactor
 - typedefinitionsrtoovertopping::tpovertoppinginput, [67](#)
- removeberms
 - geometrymodulertoovertopping, [31](#)
- removedikessegments
 - geometrymodulertoovertopping, [32](#)
- rfactor_max
 - typedefinitionsrtoovertopping, [43](#)
- rfactor_min
 - typedefinitionsrtoovertopping, [44](#)
- rootsdepressedcubic
 - formulamodulertoovertopping, [21](#)
- rootsgeneralcubic
 - formulamodulertoovertopping, [22](#)
- roughness
 - overtoppinginterface::overtoppinggeometrytype, [56](#)
 - overtoppinginterface::overtoppinggeometrytypef, [57](#)
 - overtoppinginterface::tpprofilecoordinate, [68](#)
- roughnessfactors
 - typedefinitionsrtoovertopping::tpgeometry, [60](#)
- segmentslopes
 - typedefinitionsrtoovertopping::tpgeometry, [60](#)
- segmenttypes
 - typedefinitionsrtoovertopping::tpgeometry, [61](#)
- slope_max
 - typedefinitionsrtoovertopping, [44](#)
- slope_min
 - typedefinitionsrtoovertopping, [44](#)
- splitcrosssection
 - geometrymodulertoovertopping, [32](#)
- tm_10
 - typedefinitionsrtoovertopping::tpload, [63](#)
- typeDefinitionsRTOovertopping.f90, [76](#)
- typedefinitionsrtoovertopping, [40](#)

- berm_max, [41](#)
- berm_min, [41](#)
- fb_max, [41](#)
- fb_min, [42](#)
- fn_max, [42](#)
- fn_min, [42](#)
- foreshore_max, [42](#)
- foreshore_min, [42](#)
- frunup1_max, [42](#)
- frunup1_min, [42](#)
- frunup2_max, [42](#)
- frunup2_min, [42](#)
- frunup3_max, [43](#)
- frunup3_min, [43](#)
- fs_max, [43](#)
- fs_min, [43](#)
- margindiff, [43](#)
- margingrad, [43](#)
- mz2_max, [43](#)
- mz2_min, [43](#)
- rfactor_max, [43](#)
- rfactor_min, [44](#)
- slope_max, [44](#)
- slope_min, [44](#)
- xdiff_min, [44](#)
- z2_iter_max1, [44](#)
- z2_iter_max2, [44](#)
- z2_margin, [44](#)
- typedefinitionsrtooverlapping::tpgeometry, [59](#)
 - nbermsegments, [60](#)
 - ncoordinates, [60](#)
 - psi, [60](#)
 - roughnessfactors, [60](#)
 - segmentslopes, [60](#)
 - segmenttypes, [61](#)
 - xcoorddiff, [61](#)
 - xcoordinates, [61](#)
 - ycoorddiff, [61](#)
 - ycoordinates, [61](#)
- typedefinitionsrtooverlapping::tpload, [61](#)
 - h, [62](#)
 - hm0, [62](#)
 - phi, [63](#)
 - tm_10, [63](#)
- typedefinitionsrtooverlapping::tpovertopping, [63](#)
 - qo, [64](#)
 - z2, [64](#)
- typedefinitionsrtooverlapping::tpovertoppinginput, [64](#)
 - computedovertopping, [66](#)
 - criticalovertopping, [66](#)
 - factordeterminationq_b_f_b, [66](#)
 - factordeterminationq_b_f_n, [66](#)
 - frunup1, [66](#)
 - frunup2, [66](#)
 - frunup3, [66](#)
 - fshallow, [67](#)
 - m_z2, [67](#)
 - reductionfactorforeshore, [67](#)
 - relaxationfactor, [67](#)
 - typerunup, [67](#)
- typerunup
 - typedefinitionsrtooverlapping::tpovertoppinginput, [67](#)
- validateinputc
 - dllovertopping, [10](#)
- validateinputf
 - dllovertopping, [11](#)
- varmodelfactorcriticalovertopping
 - overtoppinginterface, [40](#)
- verbosity
 - modulelogging::tlogging, [59](#)
- versionnumber
 - dllovertopping, [12](#)
- waveRunup.f90, [78](#)
- waverunup, [44](#)
 - convergedwithresidu, [45](#)
 - determinestartingvalue, [45](#)
 - findsmallestresidu, [45](#)
 - innercalculation, [45](#)
 - iterationwaverunup, [47](#)
- writecrosssection
 - geometrymodulertooverlapping, [33](#)
- xcoorddiff
 - typedefinitionsrtooverlapping::tpgeometry, [61](#)
- xcoordinate
 - overtoppinginterface::tpprofilecoordinate, [68](#)
- xcoordinates
 - typedefinitionsrtooverlapping::tpgeometry, [61](#)
- xcoords
 - overtoppinginterface::overtoppinggeometrytype, [56](#)
 - overtoppinginterface::overtoppinggeometrytypef, [58](#)
- xdiff_min
 - typedefinitionsrtooverlapping, [44](#)
- ycoorddiff
 - typedefinitionsrtooverlapping::tpgeometry, [61](#)
- ycoordinates
 - typedefinitionsrtooverlapping::tpgeometry, [61](#)
- ycoords
 - overtoppinginterface::overtoppinggeometrytype, [56](#)
 - overtoppinginterface::overtoppinggeometrytypef, [58](#)
- z2
 - typedefinitionsrtooverlapping::tpovertopping, [64](#)
- z2_iter_max1
 - typedefinitionsrtooverlapping, [44](#)
- z2_iter_max2
 - typedefinitionsrtooverlapping, [44](#)
- z2_margin
 - typedefinitionsrtooverlapping, [44](#)

zFunctionsWTIOvertopping.f90, [78](#)
zcoordinate
 overtoppinginterface::tpprofilecoordinate, [69](#)
zfunclogratios
 zfunctionswtiovertopping, [51](#)
zfunctionswtiovertopping, [48](#)
 adjustprofile, [48](#)
 calculateqorto, [50](#)
 profileinstructure, [51](#)
 zfunclogratios, [51](#)