Deltares

Enabling Delta Life

WTI2017 Failure Mechanisms - Piping Kernel

Technical Design

WTI2017 Failure Mechanisms - Piping Kernel

Technical Design

Edwin Spee

WTI2017 Failure Mechanisms - Piping Kernel

Technical Design

Edwin Spee

Version: 1.2

Date: 4 May 2016

Client	Rijkswaterstaat, WVL
Title	VTV - Dikes overtopping Technical Design

Abstract

This document contains the technical design for the dikes overtopping kernel, which forms a part of the WTI 2017 failure mechanism library. The kernel comprises different software components for the calculation of the discharge due to overtopping and the corresponding Z-value.

References

Version	Author	Date	Remarks	Review	Approved by
0.91	Edwin Spee	01-12-2015	see issue OVERS-8	Erik de Goede	
1.0	Edwin Spee	10-12-2015	idem	Erik de Goede	Arthur Baart
1.1	Edwin Spee	11-12-2015		Erik de Goede	Arthur Baart
1.2	Edwin Spee	04-05-2016	Omkeervariant, bilangual		

Project number	1220043.002
Keywords	Dikes overtopping, WTI, Kernel, Technical Design
Number of pages	-
Classification	

Status	Final

Contents

1 Introduction

1.1 Purpose and scope of this document

This document contains the technical design for the dikes overtopping kernel, which forms a part the WTI 2017 failure mechanism library. The kernel comprises different software components for predicting the discharge due to overtopping and the corresponding Z-function.

Note that the kernel is restricted to wave overtopping. Overflow is not a part of this kernel.

The document will not give any background on the context of the WTI project and on the derivation or motivation of the supported physical model. For this purpose the reader is referred to the WTI2017 and to its supporting technical reports and their background reports underneath.

This document will describe how the requirements and functional design are implemented in the kernel.

1.2 Other system documents

The full documentation on the kernel comprises the following documents.

Title	Content	Author(s)	Reviewer(s)
Requirements and functional design	Description of the requirements and functional design.	B. Kuijper M.T. Duits R.G. Kamp J.P. de Waal	J.P. de Waal, P. van Steeg
Technical design	This document	E.J. Spee	E. de Goede
Technical specification	Description of the arguments and usage of different software components, generated from in-line comment with Doxygen	generated	Not applicable
Test plan	Description of the different regression and acceptation rests, including target values.	J.P. de Waal	P. van Steeg
Test report	Actuated results of the test plan.	J.P. de Waal	P. van Steeg

1.3 Assumptions and constraints

CNS 1 As a general constraint, the development process needs to comply with the general process description for WTI software, contained in a separate document.

CNS 2 As a general constraint, the kernel needs to comply with the relevant general requirements and further design rules for the programming, documentation and testing of WTI software. This set of requirements and rules is contained in a separate document. The set includes the constraints CNS 3 to CNS 5, listed hereafter.

CNS 3 As a general WTI software constraint, the failure mechanism library will contain only components for a deterministic analysis to calculate the discharge due to wave overtopping and the corresponding Z-function.

CNS 4 As a general WTI software constraint, all appropriate model constants need to be adaptable outside the kernel, in order to allow for varying values during probabilistic analysis.

CNS 5 The software interface (API) must allow usage from Fortran (HydraRing) and C# (Ringtoets).

2 Technical Design

2.1 General

The dikes overtopping kernel must be usable in RingToets/HydraRing and other Fortran or C#-programs.

To be able to perform a calculation, regardless of the usage, input parameters are required to define the case to be calculated. An overview of all input data is given in sections 2.2 and 2.5.

2.1.1 HydraRing and other Fortran programs

In order to be usable in Hydraring, the kernel provides high level functions in the dll[1] for calculating the discharge and the resulting Z-function, where input arguments are Fortran structs.

2.1.2 RingToets and other C# and .Net programs

In order to be usable in RingToets, the kernel provides high level functions in the dll for calculating the discharge and the resulting Z-function, where input arguments are C# datatypes.

The dll must be wrapped in a C# wrapper class. In the test bench an example wrapper is available that can be used as a blueprint for the actual wrapper in RingToets. For this moment, such a wrapper will not be part of this kernel.

[1] *A dynamic link library (DLL) is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer*

2.2 Description of the required input data for a discharge calculation

To calculate the discharge due to wave overtopping, the user has to provide the load, the geometry, the dike height, the model factors, a level for amount of logging and optionally the name of the log file.

Note that the log file is currently only used when the iterations process in section 3 ends with a higher residue than expected.

The exact definition of all input structs can be found in the technical documentation. A description is given here:

parameter	*symbol*	*type*	*description*
load	h	double	local water level (m + NAP)
	H_{m0}	double	significant wave height (m)
	$T_{m-1,0}$	double	spectral wave period (s)
	φ	double	wave direction (degrees in range 0 … 360)
geometry	ψ	double	orientation of the dike normal (degrees in range 0 … 360)
	nPoints	integer	number of coordinates
	$\underline{x}$	double[]	xCoords (array of size nPoints, m)
	$\underline{y}$	double[]	yCoords (array of size nPoints, m + NAP)
	$\underline{r}$	double[]	roughness (array of size nPoints - 1)
dikeHeigth	h_{crest}	double	dike height (m + NAP)
model factors	f_n	double	model factor for non-breaking waves
	f_b	double	model factor for breaking waves
	m_{z2}	double	model factor describing the uncertainty of 2% run up height
	$f_{shallow}$	double	model factor for shallow waves
	m_{q0}	double	model factor computed overtopping discharge
	m_c	double	model factor critical overtopping discharge
	R_{user}	double	relaxation factor iteration procedure wave run-up
	f_{red}	double	reduction factor foreshore
logging	verbosity	integer	one of verboseNone (-1), verboseBasic (0), verboseDetailed (1), verboseDebugging (2)
	filename	char[255]	filename of the log file

The struct with model factors is used for both the discharge calculation and the Z-function.

The wave height H_{m0} must be >= 0. Small waves (H_{m0} <= 10^{-7} *m*) are neglected and lead to zero discharge.

2.3 Description of the output data for a discharge calculation

The calculation of the discharge leads to three types of output:

1. Overtopping output struct
2. Success flag, 0 for success, otherwise failure
3. Error text (only relevant is not successful)

The overtopping output struct consists of two fields: z_2 and Q_o, respectively 2% wave run-up (m) and the wave overtopping discharge (m3/m per s).

2.4 Possible error messages

The following input is not allowed and will lead to an error message:
1. Local water level is above the crest level (is overflow; not a part of this kernel)
2. Wave height is less than zero
3. Wave period is less than zero
4. Wave direction is not between 0 and 360 degrees
5. The model factors must be in a reasonable range and at least >= 0
6. The geometry is not valid

See section 2.7 for the possibilities of validation.

Two other error messages may occur:
1. 2% wave run up cannot be calculated, see section 3
2. Out of memory

2.5 Description of the required input data for the resulting Z-function

The computation of the Z-value[2] needs three types of input:

1. q_c : the critical overtopping discharge rate
2. m_c and m_{q0} , found in the struct with the model factors
3. q_0 : the calculated discharge

The struct with the model factors is the same struct as used in the calculation of the discharge, but only the model factors for the computed and critical overtopping are relevant (resp. m_{q0} and m_c).

[2] *Z-function: limit state function. Z < 0 means failure*

2.6 Description of the output data for the resulting Z-function

The calculation of the Z-function corresponding to the overtopping discharge, leads to three types of output:

1. Z: the computed Z-value
2. Success flag, 0 for success, otherwise failure
3. Error text (only relevant is not successful)

The Z-function will not be successful only if one of the model factors or the critical overtopping is 0 or negative.

The Z-function is given by:

$$Z = \log(m_c * q_c) - \log(\max(m_{q0} * q_o, \varepsilon))$$

Where:

m_c	Model factor critical overtopping discharge
q_c	Critical overtopping discharge
m_{q0}	Model factor computed overtopping discharge
q_0	Computed overtopping discharge
ε	Small number ($2*10^{-306}$), to avoid log(0)

2.7 Validation

As shown in section 2.4 there are many cases in which the kernel ends up with an error message. To prevent error messages during the computation, a validation routine is provided in the dll.

The validation only checks the profile and the range of the model factors.
To check the profile, also the dike height must be given.

Validation is not possible when using the old definition for the model factors for wave run up.

2.8 Version number

The dll has a function to get the version number of the kernel.
The version number can also be found in the file properties of the dll.

2.9 SetLanguage

Set the language for error messages and validation messages. Only ‘NL’ and ‘UK’ are implemented. The default language is Dutch.

2.10 GetLanguage

Get the current language for error messages and validation messages.

2.11 Omkeervariant

The inverse function: get dike height for given geometry, load, critical discharge and model factors.

3 Calculation of $z_{2\%}$, the 2% wave run-up.

As given in the functional design, the calculation of the 2% wave run up consists of several steps:

Iterate until 2% wave run-up reaches equilibrium:
a) Estimate 2% wave run-up: provide starting value for $z_{2\%}$
b) Calculate representative slope angle tan α .
c) Calculate $z_{2\%,smooth}$, neglecting the effect of berms and roughness (γ_b=1, γ_f=1).
d) Calculate influence factor roughness on slope: γ_f.
e) Calculate $z_{2\%,rough}$, neglecting the effect of berms (assume γ_b = 1).
f) Calculate influence factor berms: γ_b
g) Calculate new influence factor roughness on slope: γ_f.
h) If applicable, adjust the influence factors
i) Calculate 2% wave run-up: $z_{2\%}$

This iteration process can take many iterations and it is not guaranteed that this will converge within a reasonable margin. In some cases it happens that $z_{2\%}$ will flip flop between two values.

To improve convergence, but to change the results as little as possible, the next procedure will be used:

1. Start with $z_{2\%}$ = 1.5 H_{m0}
2. In iteration 2 … 5 use result of previous iteration
3. In iteration 6 … 25 use relaxation only if the user provides a relaxation factor
4. In iteration 26 … 49 use a relaxation factor of at least 0.5
5. Search with 10 small steps at both sides of the value with the lowest residue of the previous steps to find a new minimal residue.

When using relaxation in iterations 6 … 49, the new $z_{2\%}$ is given by:

$$z_{2\%,new} = R * z_{2\%,new} + (1 - R) * z_{2\%,prev}$$

Where:

R	Relaxation factor
$z_{2\%,new}$	$z_{2\%}$ at the end of the previous iteration
$z_{2\%,prev}$	$z_{2\%}$ at the beginning of the previous iteration

Note that this approach always leads to a value for z_2, but possible with a residue higher than the expected value of 10^{-3} *m*. In that case a warning is written to the log file, if specified.

Note that within each iteration step, steps a … i may give an error. In that case the computation of z_2 fails.

Figure 1 shows a histogram of the number of iterations needed when running the test bench with 32479 evaluations.

About 95% of the cases need less than 10 iterations. Most of them only need 2 iterations. Approximately 1% finishes shortly after switching on relaxation.

Only one case needs step 5.

All test cases finish without errors in the steps a … i.

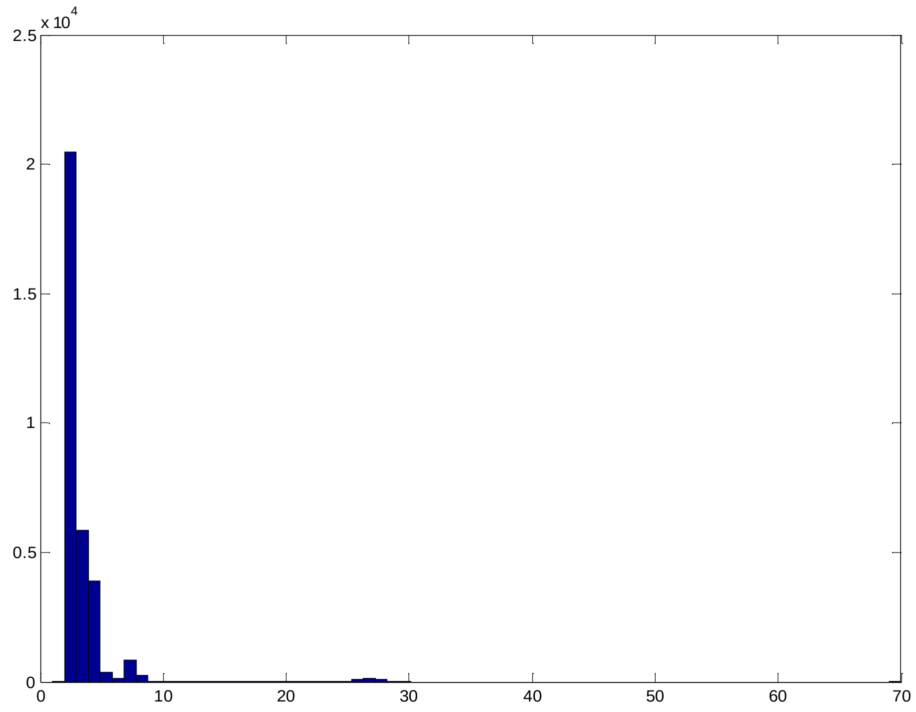

Figure 1: number of iterations needed while running the overtopping test bench