Case 3b: Working with extreme scenarios

Case 2 described the set-up of a simple probabilistic computation using Hunt's formula. Now it is time to add some real-world trouble to this case. As has been explained, Monte Carlo is a robust method, but will become very time consuming in case of extremely low probabilities of occurrence.

We use the probabilistic computation from Case 2 as starting point. In Case 2 we computed the probability of overtopping of a dike (Figure 1). We will alter the characteristics of the dike and boundary conditions to obtain a situation where overtopping is very unlikely to occur.

First, we will alter the stochastic variables. Next, we will compute the probability of overtopping using both the FORM and Crude Monte Carlo method. Finally, we will visualize the results.

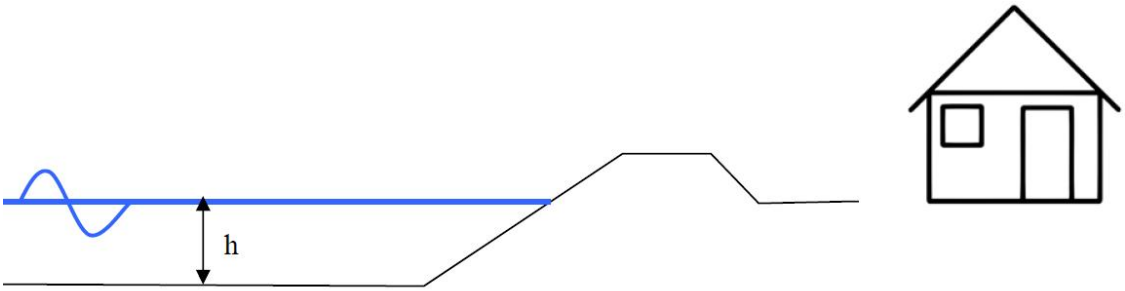

Figure 1 Situation sketch

Step-by-step description

1. No additional stochastic variables are introduced in this case, but we will choose the water level to be known and give it a value of NAP+0m. Copy the M-file from Case 2 and alter the code to do so.

   Choosing a relatively low value for the water level will decrease the probability of failure. Decreasing the number of stochastic variables will make visualization easier as well.

2. You may alter other (stochastic) variables to create an almost indestructible dike. For example:

Variable name	*Value*
h_{crest}	>> NAP+5m
tan(α)	<< 1/3

3. Copy the M-file with the limit state function description from Case 2.

4. Execute your script and visualize your results using the *plotFORMresult* and *plotMCResult* functions.

Additional questions
If you succeeded in running the probabilistic computations above, you can ask yourself the following questions in order to explore the results. The results are stored in the *Output* field of the result variable.

1. Has the FORM computation converged? How many iterations were needed? And how many limit state function realizations?

2. How many limit state function realizations were needed in the Monte Carlo simulation? How many of these realizations are in the failure domain? Is that enough?

3. What is the difference in failure probability computed by FORM and Monte Carlo? Is this a large difference? What causes the difference?

4. Are you satisfied with the FORM results? And the Monte Carlo results? If yes, you have been aloof. Start over and be extreme!

Troubleshoot
If you managed to create a real extreme scenario, you noticed that the Crude Monte Carlo method may fail to draw enough points in the failure domain to compute a (reliable) probability of failure. FORM may even crash, since it can reach the limits of the variable precision leading to infinite values (for example, try to reduce the standard deviations of the wave parameters).

We will leave FORM for what it is at this moment and concentrate on the Monte Carlo method. There are two possibilities available in the OpenEarth probabilistic toolbox to solve the problem concerning extremely small probabilities:

- Importance Sampling
- Directional Sampling

Directional Sampling is an alternative for Crude Monte Carlo and works quite the same. The same input variables (stochastic variable and z-function) can be used. However, the current implementation in the OpenEarth toolbox is still experimental. Therefore we will use Importance Sampling for now, which is an extension of the Crude Monte Carlo method as we used before.

Importance Sampling shifts the area where most samples are drawn and corrects for the error made in the computed probabilities. We will replace the earlier defined probability density functions with a uniform distribution between set limits. This will be explained step-by-step in the following:

1. Extend your M-file so it creates an Importance Sampling structure for the wave height. It has the following layout:

```
is =

      Name: 'H'
    Method: @prob_is_uniform
    Params: {0 6}
```

Where *H* is the name of the stochastic variable you want to use Importance Sampling on and *0* and *6* are the lower and upper limits between which samples will be drawn.

2. Add an additional call to the MC routine and extend it with the Importance Sampling input. So, feed the variable *is* to the parameter *IS* in the MC routine. For example:

```
M = MC( ...
    'x2zFunction',  @x2z,       ...
    'NrSamples',    3e4,        ...
    'IS',           is,         ...
    'stochast',     stochast);
```

3. Execute your code and visualize your results using the *plotMCResult* function.

4. Now add Importance Sampling to the wave period as well and repeat the computation and visualize the results. Use 0 and 8 as limits for the uniform distribution.

Additional questions (continued)
If you succeeded in running the probabilistic computations with extreme probabilities of failure, you can ask yourself the following questions in order to explore the results.

1. How many limit state function realizations were needed in the Monte Carlo simulation? How many of these realizations are in the failure domain? Is that enough?

2. What is the difference in probability computed with and without Importance Sampling? Is this a large difference? What causes the difference?

3. Can you visualize the convergence of the Monte Carlo computation in terms of probability?

   Hint: the probability of failure in a Monte Carlo simulation is determined by the number of failure points, but the number of failure points is artificially increased using Importance Sampling. Normally, all failure points have an equal value of one. In case of Importance Sampling the failure points are

valued individually. These individual values are stored in the field *P_corr* of the output.