CT4310 release notes

Kees den Heijer

January 11, 2012*

1 Introduction

This document gives some remarks about the OpenEarth release for the assignment of the CT4310 course (CT4310 Bed, Bank and Shoreline Protection).
https://repos.deltares.nl/repos/OpenEarthTools/tags/CT4310_release/CT4310_release.zip

2 2009-12-09 7:47:11 +0100 (Wed 9 Dec 2009)

Initial version, as explained in the lecture of 9 December. The Powerpoint presentation can be found on Blackboard (`CT4310_probabilistic_OET_introduction.ppt`).

3 2009-12-17 18:27:35 +0100 (Thu, 17 Dec 2009)

3.1 Input arguments of Z-function changed (!)

The structure of the Z-function has been changed in order to make it simpler to use. In the example with the *van der Meer formula* as discussed during the lecture, the Z-function call looked like:

```
function z = prob_vdMeer_example_x2z(x, varnames, Resistance, varargin)
```

In that old situation, the first part of the function was standard for any case, looking like:

```
%% retrieve calculation values
for i = 1:size(x,2)
    samples.(varnames{i}) = x(:,i);
end
```

In the new situation, the Z-function call looks like:

```
function z = prob_vdMeer_example_x2z(samples, Resistance, varargin)
```

*$Date: 2012-01-11 09:15:46 +0100 (Wed, 11 Jan 2012)$

Now, instead of input variables `x` and `varnames`, directly the structure `samples` is passed. The `fieldnames` of the structure `samples` are corresponding to the variable names as specified in `stochast.Name`. Each of the fields of `samples` contain a column vector with a length that corresponds to the number of samples. This means that the old loop to create the `samples` structure is no longer needed.

To summarise, in the *van der Meer* example, the old code was:

```
%% Z-function
function z = prob_vdMeer_example_x2z(x, varnames, Resistance, varargin)

%% retrieve calculation values
for i = 1:size(x,2)
    samples.(varnames{i}) = x(:,i);
end

%%
g = 9.81;                                   %[m/s2]
for i = 1:size(x,1)
    Delta = (samples.RhoS(i) - samples.RhoW(i)) / samples.RhoW(i);    % [-] relative density
    Ksi = samples.TanAlfa(i)/sqrt(samples.Steep(i));      % [-] Iribarren number
    z(i,:) = samples.Cpl(i)*samples.P(i)^0.18*(samples.S(i)/sqrt(samples.N(i)))^0.2*Ksi^(-0.
end
```

The new code is:

```
function z = prob_vdMeer_example_x2z(samples, Resistance, varargin)

%%
g = 9.81;                                   %[m/s2]
for i = 1:length(samples.RhoS)
    Delta = (samples.RhoS(i) - samples.RhoW(i)) / samples.RhoW(i);    % [-] relative density
    Ksi = samples.TanAlfa(i)/sqrt(samples.Steep(i));      % [-] Iribarren number
    z(i,:) = samples.Cpl(i)*samples.P(i)^0.18*(samples.S(i)/sqrt(samples.N(i)))^0.2*Ksi^(-0.
end

%{
% alternatively, the z can be calculated as matrix operation (so, no loop
% needed) as follows:
Delta = (samples.RhoS - samples.RhoW) ./ samples.RhoW;    % [-] relative density
Ksi = samples.TanAlfa ./ sqrt(samples.Steep);      % [-] Iribarren number
z = samples.Cpl .* samples.P .^0.18.*(samples.S./sqrt(samples.N)).^0.2.*Ksi.^(-0.5)-samples.
%}
```

3.2 Binominal distribution added

A function for the binominal distribution has been added. In the `stochast` structure, in the field `stochast.Distr`, it can be specified as `@bino_inv`. The

only parameter, which has to be specified in `stochast.Params` is `p_success`. The output `x` is a `logical` array, so, elements are either `true` (`1`) or `false` (`0`).

3.3 Triangular distribution added

A function for the triangular distribution has been added. In the `stochast` structure, in the field `stochast.Distr`, it can be specified as `@trian_inv`. The parameters `a` (lower limit), `b` (upper limit) and `c` (mode) have to be specified in `stochast.Params`.

4 2009-12-18 8:38:14 +0100 (Fri, 18 Dec 2009)

In this new release, the general tutorial (`prob_calculation_tutorial.m`) has been changed to be consistent with the new structure of the Z-function as explained in subsection 3.1.

5 2009-12-21 17:09:08 +0100 (Mon, 21 Dec 2009)

In this new release, a correction is made in the help block of the `trian_inv` function (thanks to Rory van Doorn). The distribution parameters should be:

```
%   a = lower limit of x
%   b = upper limit of x
%   c = mode of x
```

6 2011-01-03 14:19:44 +0100 (Mon, 03 Jan 2011)

The function call to the z-function has been changed in this release. For proper use, an extra field `propertyName` has to be added to the `stochast` variable. By setting the `propertyName` for each stochastic variable to `true`, the samples are parsed to the z-function as propertyname-propertyvalue pairs. In the z-function below, these propertyname-propertyvalue pairs are converted to a `samples` structure, which is easy to use in calculating the z-values.

```
function z = prob_vdMeer_example_x2z(varargin)

%% create samples-structure based on input arguments
samples = struct(varargin{:});

%% calculate z-values
% pre-allocate z
z = nan(size(samples.RhoS));
% loop through all samples and derive z-values
for i = 1:length(samples.RhoS)
```

```
    Delta = (samples.RhoS(i) - samples.RhoW(i)) / samples.RhoW(i);    % [-] relative density
    Ksi = samples.TanAlfa(i)/sqrt(samples.Steep(i));      % [-] Iribarren number
    z(i,:) = samples.Cpl(i)*samples.P(i)^0.18*(samples.S(i)/sqrt(samples.N(i)))^0.2*Ksi^(-0.
end
```

7 2012-01-11 09:03:31 +0100 (Wed, 11 Jan 2012)

The basic functionalities and input procedure has not changed. A more specific error message has been added in case complex $\frac{dz}{du}$ values occur. In addition, for advanced techniques such as Importance Sampling, which are not part of the CIE4310 exercise, the input procedure has changed.