Case 1: Basic probabilistic example

This case deals with the very basic probabilistic case of estimating the probability that the load (or solicitation) S exceeds the strength (or resistance) R. Both R and S are assumed to be normally distributed (Figure 1). The calculation will be carried out both with the Crude Monte Carlo and the FORM method.

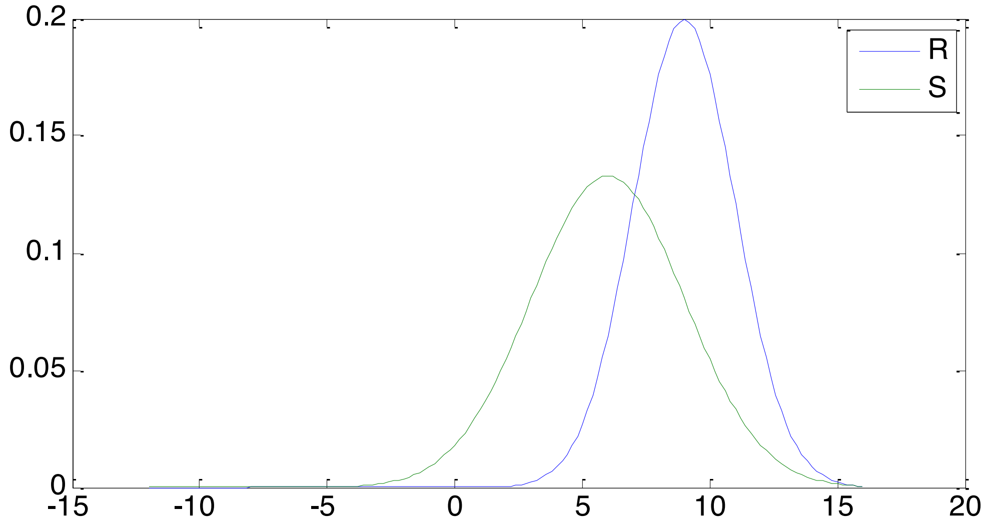

Figure 1 Distributions of the stochastic variables

The unfavorable situation or failure is described by:

$$Z = R - S < 0$$

Step-by-step description

The required code to run this example is already available (case1.m). The steps taken to come to this code are described below:

1. Define the stochastic variables. In this example we use two normal distributions with means and standard deviations as specified in the table:

Variable name	*Distribution*	*Mean*	*Standard deviation*
R	Normal	9	2
S	Normal	6	4

2. Create an M-file (case1.m) to define the necessary stochastic variable structure. It has the following layout:

```
stochast =

1x2 struct array with fields:
    Name
    Distr
    Params
```

And can be created as follows:

```
stochast = struct();

% resistance
stochast(1).Name   = 'R';
stochast(1).Distr  = @norm_inv;
stochast(1).Params = {9 2};

% solicitation
stochast(2).Name   = 'S';
stochast(2).Distr  = @norm_inv;
stochast(2).Params = {6 3};
```

3. We will use the default limit state function available in the OpenEarth toolbox. Normally, you will need a more advanced function and possibly create your own, which will be explained in the next few cases. The default limit state function has the following layout:

```
function z = x2z(varargin)
%X2Z  Basic x2z function

%% read options
OPT = struct(                        ...
    'R',    0,                       ...     % resistance value
    'S',    0                        ...     % solicitation value
);

OPT = setproperty(OPT, varargin{:});

%% compute z-values
z = OPT.R - OPT.S;
```

The variable *OPT* contains the random samples of the stochastic variables you defined in the previous step. So, it will not contain the entire stochast structure just created, but only a set of numbers (random samples in case of MC and iteration steps in case of FORM). You can access the sample for the R, for example, as follows:

```
OPT.R
```

Be aware that it is possible that this variable will be a vector containing several samples at once. Your code should be able to process these vectors item-by-item. Therefore, use dot-operators (.* and ./ etc.) in your formulations, if applicable.

4. Add a call to the FORM routine in case1.m specifying both the newly created stochastic variable structure and the z-function handle. For example:

```
F = FORM( ...
    'x2zFunction',  @x2z,      ...
    'stochast',     stochast);
```

5. Do the same using the MC routine. Choose a sensible number of samples. For example:

```
M = MC( ...
    'x2zFunction',  @x2z,      ...
    'NrSamples',    1e4,       ...
    'stochast',     stochast);
```

6. Execute your script and visualize your results using the *plotFORMresult*, *plotMCResult* and *hist* functions. For example:

```
plotFORMresult(F);
plotMCResult(M);
```

Additional questions

You should be able be to run case1.m. After running this script, several figures pop up on your screen, and among others the variables M and F appear in your workspace. Explore these variables and pay special attention to the *Output* field to answer the following questions:

1. Has the FORM computation converged? What is the probability of failure according to the FORM and the Monte Carlo method?

2. What is the value of beta? What would be de probability of failure in case beta was 2?

3. In the scatter plot of the Monte Carlo result, a divide between the failure and non-failure area is visible. This is the limit state function Z, which should be a 1:1 line, since R = S. Is it?

4. Modify the standard deviation of stochastic variable R to 3. Run again the script case1.m. What is the difference between the two runs? What causes the difference?