Probabilistic tools in OpenEarth

Kees den Heijer

Delft University of Technology, Deltares

July 1, 2014

Outline

TU Delft

Deltares
Enabling Delta Life

Stochastic variable

Structure array

fieldname	description
`Name`	Unique name for each stochastic variable
`Distr`	Functionhandle of distribution function (e.g. `@norm_inv`)
`Params`	Parameters in cell-array as input for the corresponding distribution function

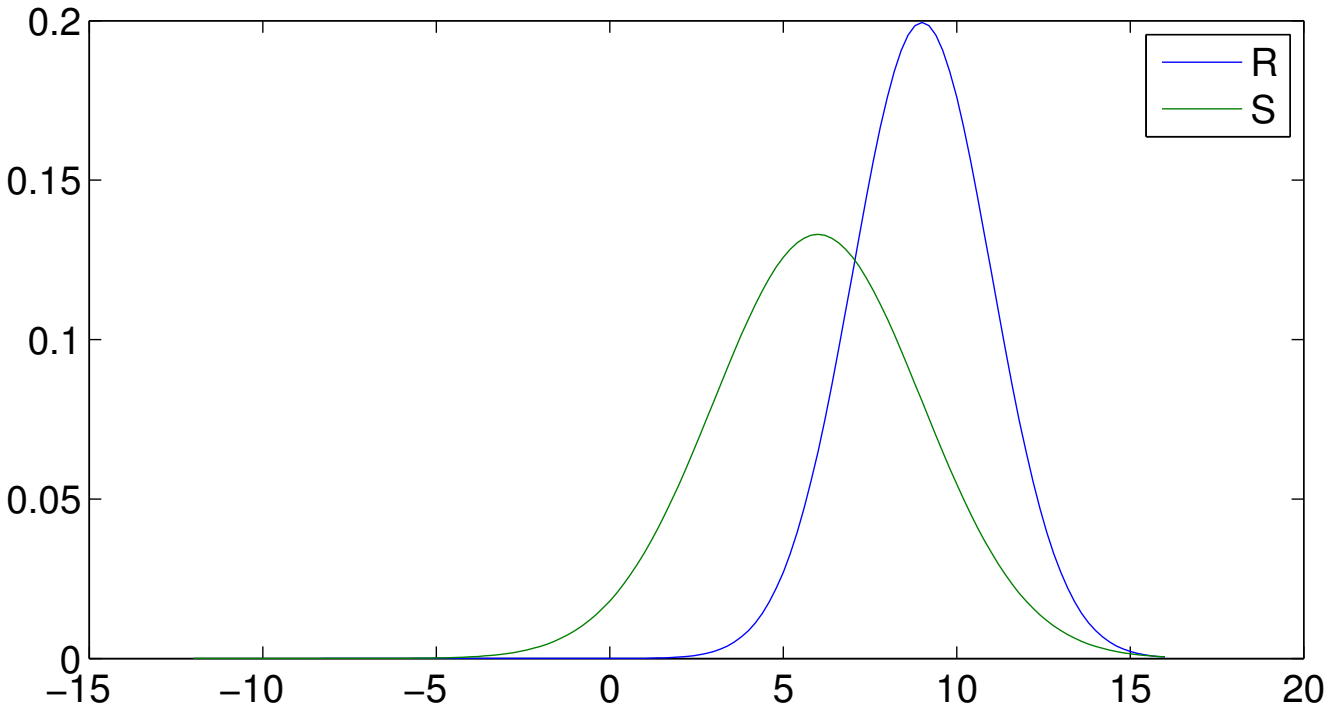
Stochastic variable

```
%% create stochastic variables
stochast = struct();

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

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

Plot stochastic variables



Probability distributions

Distribution	Function handle	Parameters
binomial	`@bino_inv`	`p_success`
χ^2	`@chi2_inv`	`v`
conditional weibull	`@conditionalWeibull`	`omega, rho, alpha, sigma, lambda`
deterministic	`@deterministic`	`x`
extreme value	`@ev_inv`	`mu, sigma, pcov, alpha`
exponential	`@exp_inv`	`lambda, epsilon`
gamma	`@gam_inv`	`a, b, pcov, alpha`
gumbel	`@gumbel_inv`	`mu, sigma`
logistic	`@logistic_inv`	`a, b`
lognormal	`@logn_inv`	`mu, sigma`
normal	`@norm_inv`	`mu, sigma`
rayleigh	`@rayl_inv`	`b`
triangular	`@trian_inv`	`a, b, c`
uniform	`@unif_inv`	`a, b`

Limit state function

- This should be created as a separate function
- Input arguments:
  `varargin` : *propertyname-propertyvalue* pairs specifying the variable names and the corresponding vectors with samples
- Output argument:
  `z` : z-values corresponding to the x-values from the samples

Limit state function

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

%% read options

OPT = struct(                          ...
    'R',   0,                          ...     % resistance value
    'S',   0                           ...     % sollicitation value
);

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

%% compute z-values

z = OPT.R - OPT.S;
```

Run calculation

- general
  - Specify stochast with:
    ```
    'stochast', stochast_variable
    ```
  - Specify z-function with:
    ```
    'x2zFunction', @your_custom_zfunction
    ```
- Monte Carlo
  - Specify number of samples with:
    ```
    'NrSamples', 1e4
    ```
- Additional settings can be parsed as *propertyname-propertyvalue* pairs

Run calculation

```
%% main matter: running the calculation
% run the calculation using Monte Carlo
resultMC = MC(...
    'stochast', stochast,...
    'NrSamples', 3e4,...
    'x2zFunction', @x2z);

% run the calculation using FORM
resultFORM = FORM(...
    'stochast', stochast,...
    'x2zFunction', @x2z);
```

FORM settings

```
% defaults
OPT = struct(...
    'stochast', struct(),... % stochast structure
    'x2zFunction', @x2z,...  % Function to transform x to z
    'x2zVariables', {{}},... % additional variables to use in x2zFunction
    'method', 'matrix',...   % z-function method 'matrix' (default) or 'loop'
    'maxiter', 50,...        % maximum number of iterations
    'DerivativeSides', 1,... % 1 or 2 sided derivatives
    'startU', 0,...          % start value for elements of u-vector
    'du', .3,...             % step size for dz/du / Perturbation Value
    'epsZ', .01,...          % stop criteria for change in z-value
    'maxdZ', 0.1,...         % second stop criterion for change in z-value
    'epsBeta', .01,...       % stop criteria for change in Beta-value
    'Relaxation', .25,...    % Relaxation value
    'dudistfactor', 0,...    % power factor to apply different du to each variable based on the response
    'logconvergence', '' ... % optionally specify file here to log convergence status
    );
```

Monte Carlo settings

```
% defaults
OPT = struct(...
    'stochast',      struct(),    ...     % stochast structure
    'x2zFunction',   @x2z,        ...     % Function to transform x to z
    'x2zVariables', {{}},         ...     % additional variables to use in x2zFunction
    'method',        'matrix',    ...     % z-function method 'matrix' (default) or 'loop'
    'NrSamples',     1e2,         ...     % number of samples
    'IS',            struct(),    ...     % sampling structure
    'P2xFunction',   @P2x,        ...     % function to transform P to x
    'seed',          NaN,         ...     % seed for random generator
    'result',        struct(),    ...     % input existing result structure to re-calculate existing samples
```

TU Delft Deltares Enabling Delta Life

Monte Carlo result

```
>> resultMC

resultMC =

    settings: [1x1 struct]
       Input: [1x2 struct]
      Output: [1x1 struct]
```

Monte Carlo result settings

```
>> resultMC.settings

ans =

     x2zFunction: @zfunction
    x2zVariables: {}
          method: 'matrix'
       NrSamples: 30000
              IS: [1x1 struct]
     P2xFunction: @P2x
            seed: NaN
      ISvariable: ''
               W: 1
              f1: Inf
              f2: 0
```

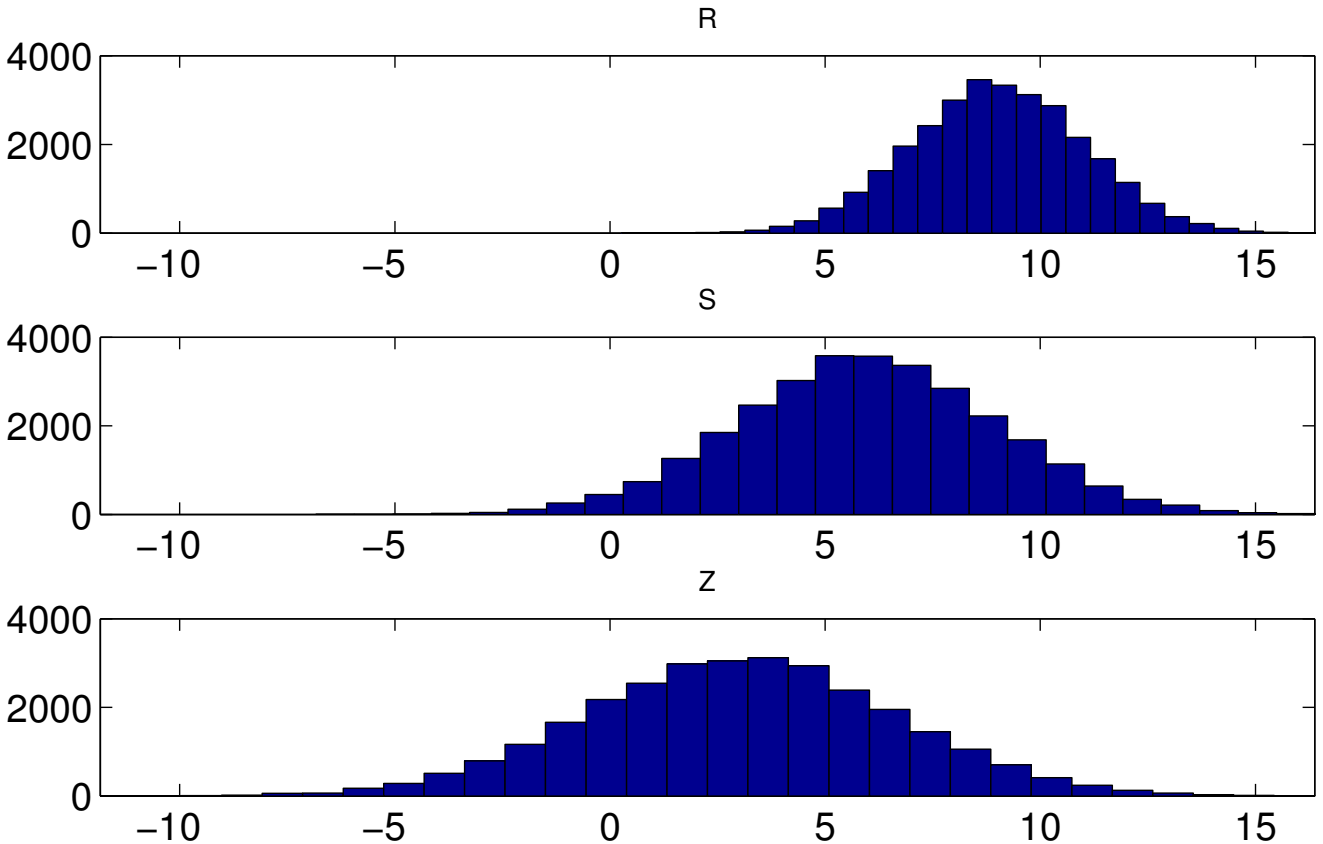
Monte Carlo result output

```
>> resultMC.Output

ans =

       P_f: 0.2033
      Beta: 0.8300
      Calc: 30000
     P_exc: [30000x1 double]
    P_corr: [30000x1 double]
    idFail: [30000x1 logical]
         u: [30000x2 double]
         P: [30000x2 double]
         x: [30000x2 double]
         z: [30000x1 double]
```

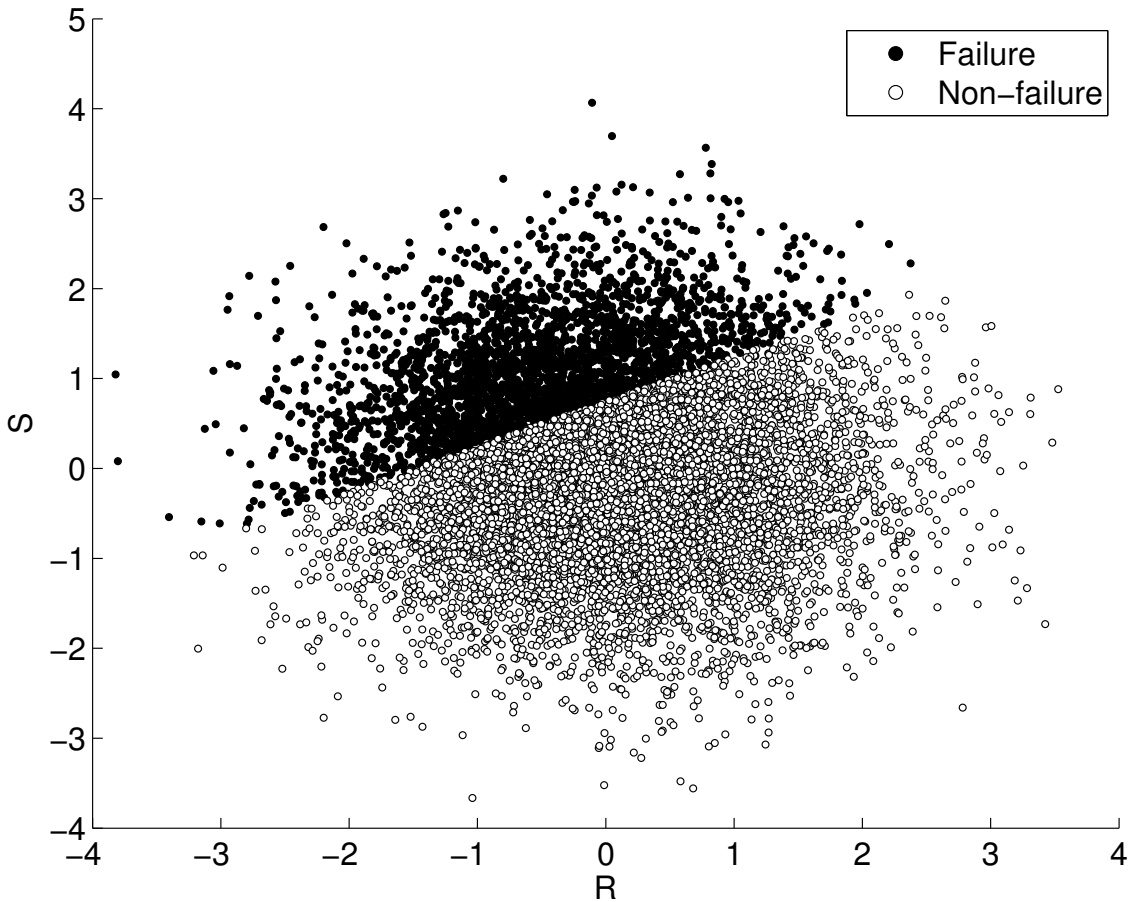
Histograms Monte Carlo



TU Delft

Deltares
Enabling Delta Life

Scatter Monte Carlo



TU Delft

Deltares
Enabling Delta Life

FORM result

```
>> resultFORM

resultFORM =

    settings: [1x1 struct]
       Input: [1x2 struct]
      Output: [1x1 struct]
```

FORM result settings

```
>> resultFORM.settings

ans =

           stochast: [1x2 struct]
         x2zFunction: @zfunction
        x2zVariables: {}
              method: 'matrix'
             maxiter: 50
    DerivativeSides: 1
             startU: 0
                 du: 0.3000
               epsZ: 0.0100
              maxdZ: 0.1000
            epsBeta: 0.0100
         Relaxation: 0.2500
       dudistfactor: 0
      logconvergence: ''
```

FORM result output

```
>> resultFORM.Output

ans =

        Converged: 1
              P_f: 0.2027
             Beta: 0.8321
            alpha: [0.5547 −0.8321]
             Iter: 18
             dzdu: [18x2 double]
           alphas: [18x2 double]
            Betas: [18x1 double]
      criteriumZ1: [18x1 double]
      criteriumZ2: [18x1 double]
    criteriumBeta: [18x1 double]
             Calc: 55
                u: [55x2 double]
                P: [55x2 double]
                x: [55x2 double]
                z: [55x1 double]
      designpoint: [1x1 struct]
```

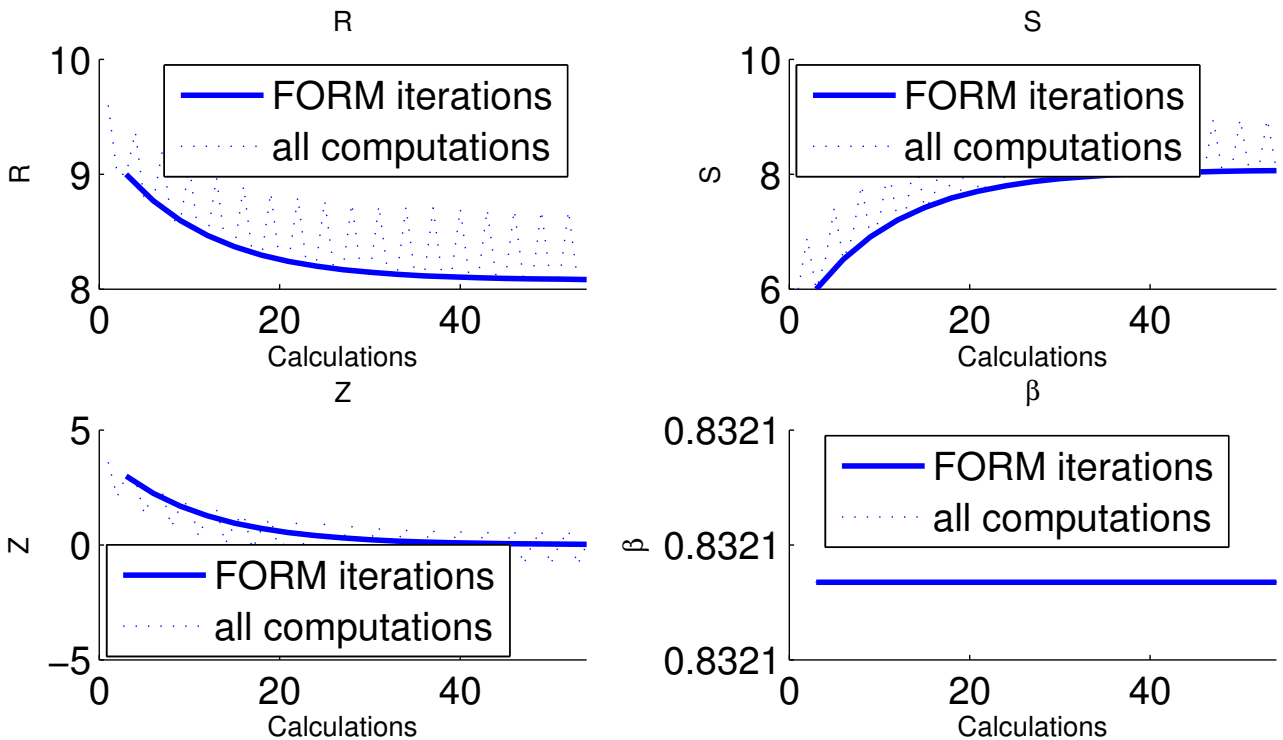
FORM result designpoint

```
>> resultFORM.Output.designpoint

ans =

          R: 8.0769
          S: 8.0769
     finalP: [0.3222 0.7556]
     finalU: [-0.4615 0.6923]
     startU: 0
      distU: 0.8321
        BSS: 0
```

FORM



TU Delft

Deltares
Enabling Delta Life