Probabilistic modeling for infrastructure lightning

Sofiane HADDAD

OpenTURNS Users Day #7

IMACS

Table of contents

Table of Contents

Study context

- Research project UMEPS (Uncertainty Management of Electromagnetic Protection on Systems)
  - Quantify & analyze uncertainty sources
  - Develop model reduction techniques
  - Demonstrate uncertainy approach in indirect lightning effects, EMC..
  - Promote an electromagnetic protection methodology using probabilistic approach
- Use case specified by Airbus Defence & Space
- Context of protection against lightning: protection of an internal equipment

Industrial context

Critical infrastructure protection against lightning:

External protection Protect the structure.

Facilitate the flow of electrical current to the ground minimizing the impedance of the path used by lightning

- One or more wire conductors stretched above protected facilities
- Downconductors
- Ground network

Internal protection Protect equipments.

Prevent from possible over voltage

- Surge protector
- Ground network
- Shield cables

Study case

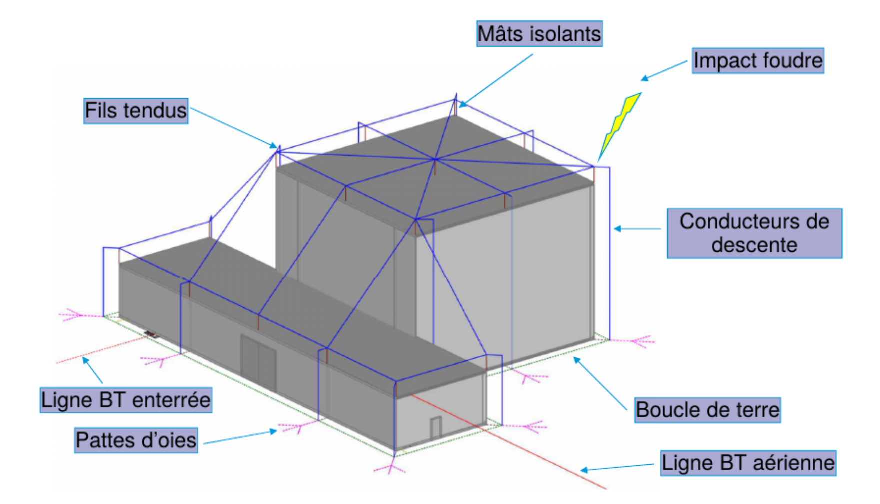


Figure: Presentation of study case

Context & tools

Numerical context

- *3D* Electromagnetic solver in frequency domain
- *BEM* method with many degrees of freedoms: huge computation time
- Solution: replace 3D problem by 1D

Compression techniques

local input parameters Sources, impedances, junctions between structural elements may change from a computation to another

Classical formulation We have to solve C $N \times N$ linear systems with a computational complexity $O(CN^3)$

Context & tools

Numerical context

- *3D* Electromagnetic solver in frequency domain
- *BEM* method with many degrees of freedoms: huge computation time
- Solution: replace 3D problem by 1D

Compression techniques

PORTS The ports of the system are defined in place of these variable local parameters, *p* ports in the model (p « N)

Context & tools

Numerical context

- *3D* Electromagnetic solver in frequency domain
- *BEM* method with many degrees of freedoms: huge computation time
- Solution: replace 3D problem by 1D

Compression techniques

Multiport approach Z_k is a rank p perturbation of matrix Z_0. Using a Schur complement, it is possible to reduce without any loss of accuracy the solution of these C $N \times N$ linear systems to:

- Computation of the admittance matrix Y ($p \times p$) $\longrightarrow$ solve 1 single linear system of size $N \times N$ with p right hand sides - complexity in $\approx N^3$
- C linear systems of size $p \times p$ - complexity $\approx Cp^3$

Goals

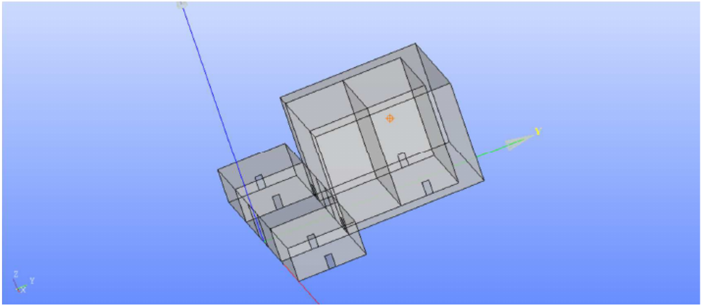


FIGURE: Position of equipment

The variable of interest is the magnetic field $\max_t H_z(t, x_c, y_c, z_c)$, measured at location (x_c, y_c, z_c)

Probabilistic goals

Variable of interest: $\max_{t} H_z(t, x_c, y_c, z_c)$

Quantities of interest

Probability of exceedance:

$$\mathbb{P}(H_z > critical)$$

Control the output variability:

$$\varphi(H_z), \phi(H_z)$$

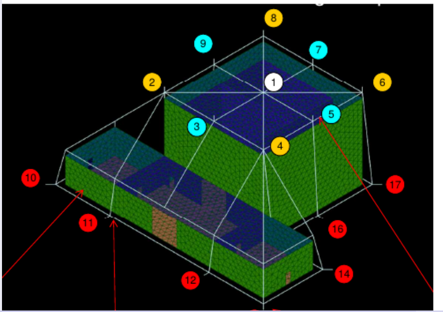
Design and compare with measurements:

$$\phi^{-1}(\alpha)$$

Table of Contents

STRUCTURE

- Point of attachment: how to dimension/position protections?
  $\Longrightarrow$ To fix protection positions is eq. to impose the lightning path
- Ground network junctions
  $\Longrightarrow$ Its efficiency relies on its impedance
- All variables are discrete
- Correlation?



Injection: choose 1 path depending on the attachment port

- Ports of lightning attachment: 9
  $\Longrightarrow$ Repartition : Ports #2, 4, 6, 8 on the corners, ports #3, 5, 7, 9 on the middle & port #1 on the center

Attachment lightning ports

Injection: choose 1 path depending on the attachment port

- Ports of lightning attachment: 9
  $\Longrightarrow$ Repartition : Ports #2, 4, 6, 8 on the corners, ports #3, 5, 7, 9 on the middle & port #1 on the center
- Port attached $= 1V$ injection (or $1A$, depending on the kind of injection), 0 for others

Attachment lightning ports

Injection: choose 1 path depending on the attachment port

- Ports of lightning attachment: 9
  $\Longrightarrow$ Repartition : Ports #2, 4, 6, 8 on the corners, ports #3, 5, 7, 9 on the middle & port #1 on the center
- Port attached = $1V$ injection (or $1A$, depending on the kind of injection), 0 for others
- Needs modeling of attachment
  $\Longrightarrow$ Several strategies: arbitrary (with some physical expectation), Zoning computations, expert...

Our model:

If we denote p the probability of attachment of the center, then:

- $2p$ is the probability of attachment for each middle port
- $4p$ is the probability of attachment for each corner port

Attachment lightning ports

Injection: choose 1 path depending on the attachment port

- Ports of lightning attachment: 9
  $\Longrightarrow$ Repartition : Ports #2, 4, 6, 8 on the corners, ports #3, 5, 7, 9 on the middle & port #1 on the center
- Port attached = $1V$ injection (or $1A$, depending on the kind of injection), 0 for others
- Needs modeling of attachment
  $\Longrightarrow$ Several strategies: arbitrary (with some physical expectation), Zoning computations, expert...

Our model:

If we denote p the probability of attachment of the center, then:

- $2p$ is the probability of attachment for each middle port
- $4p$ is the probability of attachment for each corner port

Thus $p = 4\%$

OpenTURNS modeling of attachment scenario

Simulate scenario of injection

Random vector of interest: $x \in \mathbb{R}^9$, x of type $[0, ..., 1, ..., 0]$
$\Longrightarrow$ Possibility to reduce dimension to 1 by selecting only the **port index** of injection using UserDefined

```python
import openturns as ot
def create_injection_model():
    p = 0.04
    nrports = 9
    ports = [[e] for e in range(1,nrports+1)]
    weights = [p * e for e in [1] + 4 *[4,2]]
    distribution = ot.UserDefined(ports,
        weights)
    return distribution
```

Grounding spikes

In parallel to the injection scenario, there are 13 grounding spikes (ports in the model, with numbers from 10 to 22):

$\Longrightarrow$ They are connected in parallel $\longrightarrow Z_{nominal}$

Grounding spikes

In parallel to the injection scenario, there are 13 grounding spikes (ports in the model, with numbers from 10 to 22):
$\Longrightarrow$ They are connected in parallel $\longrightarrow Z_{nominal}$
Problem: we may face spike failures (due for example to bad connections):

- Failure on a spike is modeled by an infinite impedance $Z = \infty$
- For numerical purposes, $Z_\infty = 10000\ \Omega$
- We could expect 0 spike failure, 1, 2...or 13

Grounding spikes

In parallel to the injection scenario, there are 13 grounding spikes (ports in the model, with numbers from 10 to 22):

$\Longrightarrow$ They are connected in parallel $\longrightarrow Z_{nominal}$

Problem: we may face spike failures (due for example to bad connections):

- Failure on a spike is modeled by an infinite impedance $Z = \infty$
- For numerical purposes, $Z_\infty = 10000\ \Omega$
- We could expect 0 spike failure, 1, 2...or 13

Question: How to select failure ports?

Grounding spikes

In parallel to the injection scenario, there are 13 grounding spikes (ports in the model, with numbers from 10 to 22):

$\Longrightarrow$ They are connected in parallel $\longrightarrow Z_{nominal}$

Problem: we may face spike failures (due for example to bad connections):

- Failure on a spike is modeled by an infinite impedance $Z = \infty$
- For numerical purposes, $Z_{\infty} = 10000\ \Omega$
- We could expect 0 spike failure, 1, 2...or 13

Question: How to select failure ports?

The selection problem is split into 2 parts:

1. Select the number of failures
2. Select failure ports

Select the number of failures

In general, we observe 0, 1, 2 failures. Observing more than 2 failures is considered as a rare event.

Also, the probability of failure decreases with spikes number.

Select the number of failures

In general, we observe 0, 1, 2 failures. Observing more than 2 failures is considered as a rare event.
Also, the probability of failure decreases with spikes number.
$\Longrightarrow$ A Poisson distribution seems accurate to model the number of failure ports:

$$\mathbb{P}(\mathbb{X} = k) = \frac{\lambda^k e^{-\lambda}}{k!}, k \in \mathbb{N}$$

1. $\mathbb{X}$ is a random variable = number of failure ports
2. λ is a numerical parameter, to be set
3. As support is not finite, use of a truncated Poisson distribution ($support = [0, 13]$)

Select the number of failures

In general, we observe 0, 1, 2 failures. Observing more than 2 failures is considered as a rare event.
Also, the probability of failure decreases with spikes number.
$\Longrightarrow$ A Poisson distribution seems accurate to model the number of failure ports:

$$\mathbb{P}(\mathbb{X} = k) = \frac{\lambda^k e^{-\lambda}}{k!}, k \in \mathbb{N}$$

1. $\mathbb{X}$ is a random variable = number of failure ports
2. λ is a numerical parameter, to be set
3. As support is not finite, use of a truncated Poisson distribution ($support = [0, 13]$)

Choice for λ:

1. Probability decreases with k increasing: λ in $]0, 1]$
2. For $k = 3$, the probability should be *negligible*: $\lambda = 0.6$ seems accurate

Select the failure ports

Remark: there are 2 cases:

- Number of failure $= 0$: no choice to perform (impedance is $Z_{nominal}$)!
- Number of failure > 0: select port numbers from 10 to 22

Select the failure ports

Remark: there are 2 cases:

- Number of failure $= 0$: no choice to perform (impedance is $Z_{nominal}$)!
- Number of failure > 0: select port numbers from 10 to 22

Solution: *sampling without replacement*:

1. Consider the list $[1, 2, .., 13]$
2. We are interested in k elements, $k \leq 13$
3. Select randomly an element $\Longrightarrow$ index of failure port
4. We consider the new list without the element previously selected
5. Iterate steps 3,4 $(k-1)$ times

Result $\Longrightarrow$ list of size k

Select the failure ports

Remark: there are 2 cases:

- Number of failure $= 0$: no choice to perform (impedance is $Z_{nominal}$)!
- Number of failure > 0: select port numbers from 10 to 22

Solution: *sampling without replacement*:

1. Consider the list $[1, 2, .., 13]$
2. We are interested in k elements, $k \leq 13$
3. Select randomly an element $\Longrightarrow$ index of failure port
4. We consider the new list without the element previously selected
5. Iterate steps 3,4 $(k - 1)$ times

Result $\Longrightarrow$ list of size k

OpenTURNS tool for this sampling: `KPermutationsDistribution`

Usage: KPermutationsDistribution(k,n) (n here is 14, because we could select 13)

Algorithmic details for sampling

1. Sample the scenario of injection $\Longrightarrow$ port index from 1 to 9
   For that index, source is 1 V, 0 for others
2. Sample the number of failures k
3. If $k = 0$, then impedance is $Z_{nominal}$ for all ports
4. If $k > 0$, sample the failure ports indexes (from 13 to 22)
   Fix Z_∞ for these ports, $Z_{nominal}$ for the others

ALGORITHMIC DETAILS FOR SAMPLING

1. Sample the scenario of injection $\Longrightarrow$ port index from 1 to 9
   For that index, source is 1 V, 0 for others
2. Sample the number of failures k
3. If $k = 0$, then impedance is $Z_{nominal}$ for all ports
4. If $k > 0$, sample the failure ports indexes (from 13 to 22)
   Fix Z_∞ for these ports, $Z_{nominal}$ for the others

PythonRandomVector OR *PythonDistribution*?

- Numpy/python capabilities are useful
- Both are consumed by OpenTURNS algorithms!
- Interest is only sampling
- `PythonDistribution` requires `computeCDF` $\rightarrow$ *PythonRandomVector*

Remark: in this case, we could **explicitely** write all potential events:

- We can define all points/weights for failure ports: 2^n
- Combined with injection case, we get here 73728 cases ($n = 13$)

Random vector class 1/2

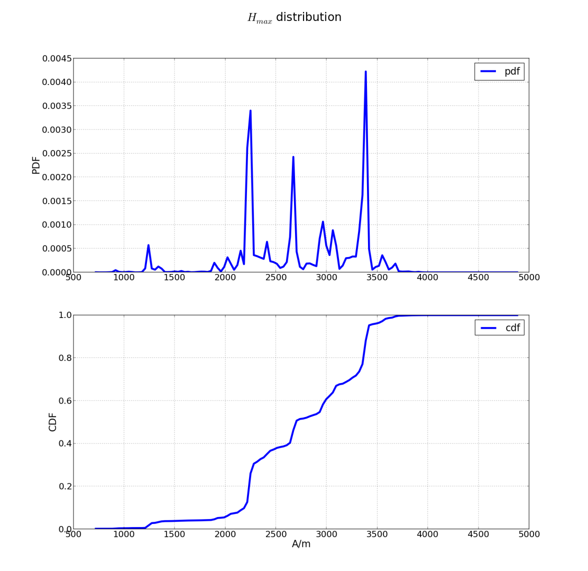
```
class MultiportRV(ot.PythonRandomVector):

  def __init__(self, lambda_=0.6,
     z_nominal=100, z_infty=10000):
    # Dimension ==> 14
    ot.PythonRandomVector.__init__(self,14)
    self.n_fail = self.getDimension() - 1
    self.z_nominal_ = z_nominal
    self.z_infty_ = z_infty
    # Scenario of injection
    self.injection_ = create_injection_model()
    # Z (spike) modelization ==> Poisson
       (Truncated) for number of failure spikes
    poisson = ot.Poisson(lambda_)
    self.failure_dist_ =
       ot.TruncatedDistribution(poisson, 0.0,
       13.0)
```

Random vector class 2/2

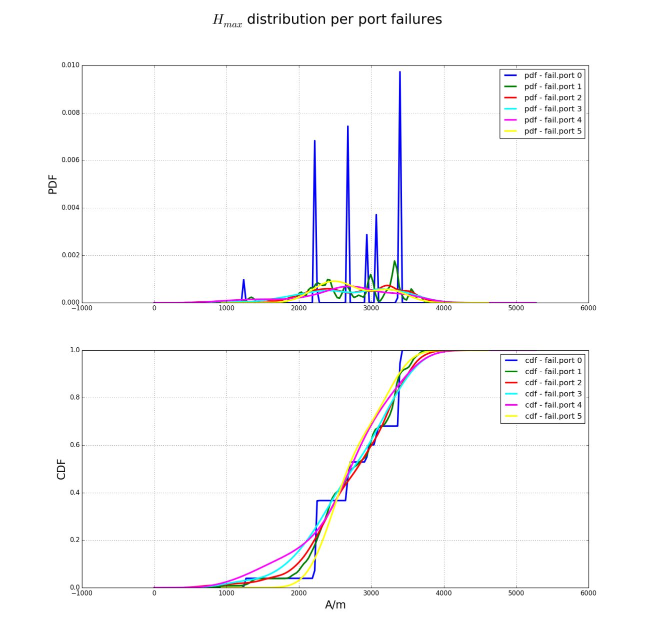
```
def getRealization(self):
  # which port of injection?
  x = list(self.injection_.getRealization())
  # nr of failure spikes
  k = self.failure_dist_.getRealization()[0]
  # Nominal impedances
  y = self.n_fail * [self.z_nominal_]
  # change impedance from z_nominal to z_infty
     if k > 0
  if k > 0:
    dist = ot.KPermutationsDistribution(int(k),
        self.n_fail)
    list_failure_spikes = dist.getRealization()
    for index in list_failure_spikes:
        y[int(index)] = self.z_infty_
  return x + y
#usage
inRv = ot.RandomVector(MultiportRV())
outRv = ot.RandomVector(myWrapper, inRv)
```

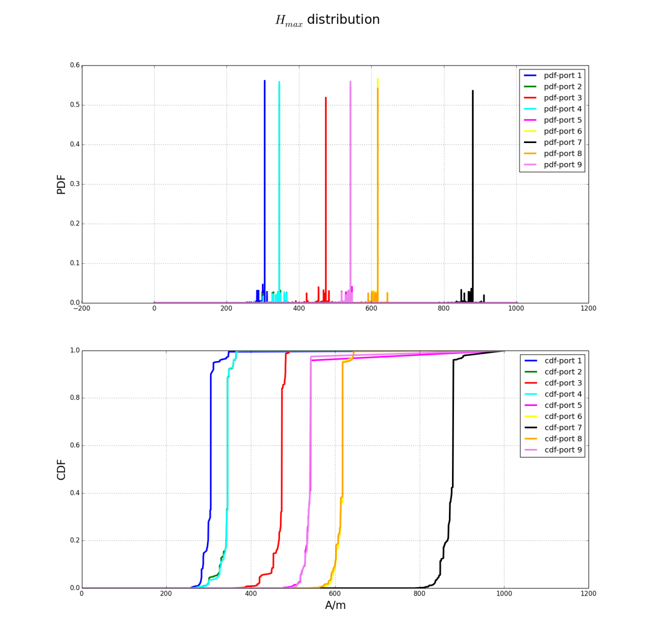
Table of Contents

10000 runs, possible thanks to 1D compression (10 s/run)



Some kind of sensitivity analysis 1/2





H_{max} distribution
PDF
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−200
0
200
400
600
800
1000
1200
pdf-port 1
pdf-port 2
pdf-port 3
pdf-port 4
pdf-port 5
pdf-port 6
pdf-port 7
pdf-port 8
pdf-port 9
CDF
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
A/m
cdf-port 1
cdf-port 2
cdf-port 3
cdf-port 4
cdf-port 5
cdf-port 6
cdf-port 7
cdf-port 8
cdf-port 9

Table of Contents

Conclusion

From the study part:

- Operability of the methodology
- Benefits of the compression method (10s/run, compared to 1h/run)
- Drawback: missing statistical data

From the OpenTURNS methodology:

- Easy implementation of the wrapper
- Fundamental ingredients: TruncatedDistribution, UserDefined, KPermutationsDistribution
- Use of OpenTURNS capabilities through the development of `RandomVector` in `python`
- However could not use `QuadraticCumul`, `ImportanceSampling`, `FORM`, `SORM`