Uncertainty treatment in dispersion modelling of accidental releases

OpenTURNS Users Day #8. June 12, 2015

Felipe Aguirre Martinez
Dispersion modeling in the presence of uncertainty

Risk assessment methodology for urgent situations

First results

Second study – Some lessons learned so far

Conclusions
First study: École Centrale de Lyon

Vincent Dubourg, Patrick Armand, David Poulet, Florian Vendel, Sébastien Argence, Thierry Yalamas, Fabien Brocheton and Perrine Volta
**First study: École Central de Lyon**

- **High fidelity** atmospheric dispersion modelling…

  ![Maps](image1.png)

  ![Maps](image2.png)

  … increasingly depends on our knowledge of the exact environmental conditions.

  Such conditions are unknown to some extent, especially in the case of accidental releases.

We propose *a risk assessment framework* that accounts for such uncertainty in the form of *probability distributions.*
First study: École Central de Lyon

Dispersion modelling

- The exact source location is supposedly known.
- The release lasts 5:00 minutes
- Meteorological conditions (wind speed, direction, etc. ...) are uncertain (imprecise).
- A Lagrangian model (SLAM) is used for simulating the dispersion of the pollutant (assuming a light gas behaviour).
- A pre-computed CFD database enables the calculation of the perturbed wind field in the constructed area in the vicinity of the source for a large variety of incident winds (using multi-linear interpolation).
First study: École Central de Lyon

Dispersion modelling
First study : École Central de Lyon

Uncertainty modelling

- The lack of knowledge about some parameters describing the release conditions is modelled as a probability distribution.
- These variables are assumed independent.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Probability distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wind speed</td>
<td>Gaussian with mean 2 m.s(^{-1}) and standard deviation 0.17 m.s(^{-1})</td>
</tr>
<tr>
<td>Wind direction</td>
<td>Truncated Gaussian with mean 225(^\circ) and standard deviation 22.15(^\circ), over [215(^\circ); 234(^\circ)]</td>
</tr>
<tr>
<td>Cloud</td>
<td>Truncated Gaussian with mean 6 octas and standard deviation 1 octa, over [1 octa; 9 octas]</td>
</tr>
<tr>
<td>Temperature</td>
<td>Uniform over [14(^\circ)C; 16(^\circ)C]</td>
</tr>
<tr>
<td>Emitted quantity</td>
<td>Uniform over [70 kg.s(^{-1}); 130 kg.s(^{-1})]</td>
</tr>
<tr>
<td>Source height</td>
<td>Uniform over [1.75 m; 2.25 m]</td>
</tr>
</tbody>
</table>
Dispersion & uncertainty modelling

Quantity of interest for risk assessment

- We consider the *cumulated dose causing irreversible effects on human health* according to INERIS recommendations for *phosphine*:

\[
D(X, p, t) = \int_0^t C_{PH_3}(X, p, \tau)^n d\tau
\]

where:

- \( X \) denotes the random vector of *uncertain release conditions*
- \( p \) and \( t \) are the *position* and *exposure time* respectively
- \( C_{PH_3} \) is the *instant phosphine concentration* calculated by SLAM
- \( n = 0.53 \) according to INERIS

- The subject is assumed *not to move* during exposure.

- The *risk analysis* consists in estimating:

\[
p = \text{Prob}[D(X, p, t) > D_0]
\]

where \( D_0 = 20.10 \) according to INERIS.
Sommaire

- Dispersion modeling in the presence of uncertainty
- Risk assessment methodology for urgent situations
- First results
- Second study – Some lessons learned so far
- Conclusions
Risk assessment methodology

Brute-force approach

- The spatio-temporal field of exceedance probabilities can be estimated using Monte Carlo sampling:

\[
P(p, t) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(D(X^{(i)}, p, t) > D_0)
\]

- This estimator converges as the number of samples (the number of SLAM runs) increases.

- A minimum of 10,000 samples is required in order to achieve a reasonable coefficient of variation of 32% on a probability of \(10^{-3}\).

- With 10 minutes per simulation, this would take two months!

Such a large number of SLAM runs is incompatible with the urgency associated to accidental releases scenarii.
Risk assessment methodology

We propose to replace SLAM by a *surrogate model* that is *much faster to evaluate*.

### Elements of surrogate modelling

**DOE**
- Run the model $\mathcal{M}$ on a well-chosen set of input (gathered in an experimental design).
- The purpose is to capture the largest amount of information about the functional relationship between its input $x$ and output $y$.

**fit**
- Choose a family of surrogate models amongst artificial neural networks (ANN), support vector machine (SVM), Gaussian processes (GP), generalized linear models (LM).
- Compute the surrogate model parameters from the dataset $\mathcal{D} = (x^{(i)}, y^{(i)}), i = 1, ..., m$.

**validate**
- Compute summary statistics about the relative error between the original model and its approximation.
- The purpose is to qualify the surrogate model on a bounded domain of the input space.

**predict**
- Use the surrogate model instead of the original model to speed up uncertainty quantification or optimization post-processings.
Risk assessment methodology

Dimension reduction using principal component analysis (PCA)

• We could apply kriging for all $p$ and $t$ over a spatio-temporal grid in order to surrogate the whole output of SLAM.

• But this would be heavy/long for dense grids ($N_x \times N_y \times N_t = 50 \times 50 \times 71$, for the present application)!

• It is proposed to exploit the significant spatio-temporal correlation (coherence) that exists in the output of SLAM for reducing its dimension to a minimal vector of principal components.

• Kriging is then applied to each component of the reduced vector $z$ instead of the original one (the inverse transform is used at predict time).
Sommaire

- Dispersion modeling in the presence of uncertainty
- Risk assessment methodology for urgent situations
- First results
- Second study – Some lessons learned so far
- Conclusions
Monte Carlo experiment used for validation of the surrogate-based approach ($N = 1,000$)

Simulations where distributed on Hyperion (CICT)
- OpenTURNS Python wrapper
- PBS scheduler
- 512 CPUs with 4Go RAM per CPU
- 1000 simulations finished in ~40 minutes
Design of experiments

- Design of experiments used for the surrogate modelling
- K-means clustering
- Subset selection in the previous Monte Carlo experiment, $m = 100$
Résultats sur la grille plane

Validation of the surrogate models
- One surrogate model per time step (PCA over the space)
- Indicators indicate the average over the $10^4$ points on the field.

<table>
<thead>
<tr>
<th>Pas de temps (d = 2500)</th>
<th>$r &lt;&lt; d$</th>
<th>$Q^2$</th>
<th>$R^2$ (test)</th>
<th>Pas de temps (d = 2500)</th>
<th>$r &lt;&lt; d$</th>
<th>$Q^2$</th>
<th>$R^2$ (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>79</td>
<td>0.92</td>
<td>0.90</td>
<td>37</td>
<td>93</td>
<td>0.68</td>
<td>0.56</td>
</tr>
<tr>
<td>2</td>
<td>79</td>
<td>0.92</td>
<td>0.90</td>
<td>38</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
</tr>
<tr>
<td>3</td>
<td>79</td>
<td>0.92</td>
<td>0.90</td>
<td>39</td>
<td>93</td>
<td>0.69</td>
<td>0.60</td>
</tr>
<tr>
<td>4</td>
<td>88</td>
<td>0.81</td>
<td>0.76</td>
<td>40</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
</tr>
<tr>
<td>5</td>
<td>88</td>
<td>0.82</td>
<td>0.76</td>
<td>41</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
</tr>
<tr>
<td>6</td>
<td>89</td>
<td>0.82</td>
<td>0.77</td>
<td>42</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
</tr>
<tr>
<td>7</td>
<td>89</td>
<td>0.72</td>
<td>0.57</td>
<td>43</td>
<td>93</td>
<td>0.67</td>
<td>0.58</td>
</tr>
<tr>
<td>8</td>
<td>90</td>
<td>0.73</td>
<td>0.64</td>
<td>44</td>
<td>93</td>
<td>0.67</td>
<td>0.58</td>
</tr>
<tr>
<td>9</td>
<td>91</td>
<td>0.74</td>
<td>0.65</td>
<td>45</td>
<td>93</td>
<td>0.67</td>
<td>0.58</td>
</tr>
<tr>
<td>10</td>
<td>91</td>
<td>0.65</td>
<td>0.57</td>
<td>46</td>
<td>93</td>
<td>0.66</td>
<td>0.57</td>
</tr>
<tr>
<td>11</td>
<td>91</td>
<td>0.67</td>
<td>0.59</td>
<td>47</td>
<td>93</td>
<td>0.65</td>
<td>0.57</td>
</tr>
<tr>
<td>12</td>
<td>92</td>
<td>0.68</td>
<td>0.60</td>
<td>48</td>
<td>93</td>
<td>0.65</td>
<td>0.56</td>
</tr>
<tr>
<td>13</td>
<td>92</td>
<td>0.64</td>
<td>0.64</td>
<td>49</td>
<td>93</td>
<td>0.64</td>
<td>0.54</td>
</tr>
<tr>
<td>14</td>
<td>92</td>
<td>0.66</td>
<td>0.60</td>
<td>50</td>
<td>93</td>
<td>0.63</td>
<td>0.55</td>
</tr>
<tr>
<td>15</td>
<td>92</td>
<td>0.67</td>
<td>0.61</td>
<td>51</td>
<td>93</td>
<td>0.62</td>
<td>0.55</td>
</tr>
<tr>
<td>16</td>
<td>93</td>
<td>0.65</td>
<td>0.58</td>
<td>52</td>
<td>93</td>
<td>0.61</td>
<td>0.54</td>
</tr>
<tr>
<td>17</td>
<td>93</td>
<td>0.67</td>
<td>0.60</td>
<td>53</td>
<td>93</td>
<td>0.60</td>
<td>0.54</td>
</tr>
<tr>
<td>18</td>
<td>93</td>
<td>0.68</td>
<td>0.61</td>
<td>54</td>
<td>93</td>
<td>0.59</td>
<td>0.53</td>
</tr>
<tr>
<td>19</td>
<td>93</td>
<td>0.67</td>
<td>0.55</td>
<td>55</td>
<td>93</td>
<td>0.58</td>
<td>0.53</td>
</tr>
<tr>
<td>20</td>
<td>93</td>
<td>0.68</td>
<td>0.60</td>
<td>56</td>
<td>93</td>
<td>0.58</td>
<td>0.52</td>
</tr>
<tr>
<td>21</td>
<td>93</td>
<td>0.68</td>
<td>0.61</td>
<td>57</td>
<td>93</td>
<td>0.57</td>
<td>0.52</td>
</tr>
<tr>
<td>22</td>
<td>93</td>
<td>0.67</td>
<td>0.51</td>
<td>58</td>
<td>93</td>
<td>0.56</td>
<td>0.50</td>
</tr>
<tr>
<td>23</td>
<td>93</td>
<td>0.68</td>
<td>0.60</td>
<td>59</td>
<td>93</td>
<td>0.55</td>
<td>0.50</td>
</tr>
<tr>
<td>24</td>
<td>93</td>
<td>0.69</td>
<td>0.60</td>
<td>60</td>
<td>93</td>
<td>0.54</td>
<td>0.49</td>
</tr>
<tr>
<td>25</td>
<td>93</td>
<td>0.67</td>
<td>0.57</td>
<td>61</td>
<td>93</td>
<td>0.53</td>
<td>0.49</td>
</tr>
<tr>
<td>26</td>
<td>93</td>
<td>0.68</td>
<td>0.60</td>
<td>62</td>
<td>93</td>
<td>0.53</td>
<td>0.49</td>
</tr>
<tr>
<td>27</td>
<td>93</td>
<td>0.68</td>
<td>0.60</td>
<td>63</td>
<td>92</td>
<td>0.52</td>
<td>0.48</td>
</tr>
<tr>
<td>28</td>
<td>93</td>
<td>0.68</td>
<td>0.55</td>
<td>64</td>
<td>92</td>
<td>0.51</td>
<td>0.48</td>
</tr>
<tr>
<td>29</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
<td>65</td>
<td>92</td>
<td>0.50</td>
<td>0.47</td>
</tr>
<tr>
<td>30</td>
<td>93</td>
<td>0.69</td>
<td>0.60</td>
<td>66</td>
<td>92</td>
<td>0.49</td>
<td>0.47</td>
</tr>
<tr>
<td>31</td>
<td>94</td>
<td>0.68</td>
<td>0.57</td>
<td>67</td>
<td>92</td>
<td>0.48</td>
<td>0.46</td>
</tr>
<tr>
<td>32</td>
<td>94</td>
<td>0.68</td>
<td>0.59</td>
<td>68</td>
<td>92</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>33</td>
<td>94</td>
<td>0.68</td>
<td>0.59</td>
<td>69</td>
<td>92</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>34</td>
<td>94</td>
<td>0.68</td>
<td>0.58</td>
<td>70</td>
<td>92</td>
<td>0.46</td>
<td>0.44</td>
</tr>
<tr>
<td>35</td>
<td>93</td>
<td>0.68</td>
<td>0.59</td>
<td>71</td>
<td>92</td>
<td>0.45</td>
<td>0.44</td>
</tr>
<tr>
<td>36</td>
<td>93</td>
<td>0.68</td>
<td>0.60</td>
<td>72</td>
<td>92</td>
<td>0.45</td>
<td>0.44</td>
</tr>
</tbody>
</table>
Probability of exceeding the threshold dose of irreversible effects

- The surrogate-based approach accounts for the uncertainty in the kriging predictor (Gaussian):

\[
\hat{P}(p, t) = \frac{1}{N} \sum_{i=1}^{N} 1 - \Phi \left( \frac{D_0 - \mu_y(x^{(i)}, p, t)}{\sigma_y(x^{(i)}, p, t)} \right)
\]
Results

Risk map

- The probability of exceeding the threshold dose of irreversible effects is:
  - less than 2.5 % in the green zone;
  - between 2.5 % and 97.5 % in the orange zone;
  - larger than 97.5 % in the red zone.

Brute-force approach

\( (N = 1,000) \)

- The probability of exceeding the threshold dose of irreversible effects is:
  - less than 2.5 % in the green zone;
  - between 2.5 % and 97.5 % in the orange zone;
  - larger than 97.5 % in the red zone.

Surrogate-based approach

\( (N = 10,000) \)
Results

Risk map (with different emitted quantity distributions)

• An arbitrarily large emitted quantity distribution was first used for reaching the threshold of irreversible effects in the far field.
• A smaller emitted quantity distribution eventually augments the spread of the uncertain (orange) zone.
Sommaire

- First study : École Central de Lyon
- Risk assessment methodology for urgent situations
- First results
- Second study – Some lessons learned so far
- Conclusions
Second study on a larger scale

Objective: Apply the same methodology on a city scale

Specifications:

- Computational chain with more than one code to run (~200 different calls)
- Runtime: ~90 minutes over 64 cores...
- Spatio-temporal grid with:
  - 60 time steps
  - Spatial grid of:
    - One tile 360 x 440 for a simplified test model
    - 63 tiles of size 430 x 430 each for the real model
- Number of output variables: 60 x 63 x 430 x 430 ~ = 70 millions!
  - Test model: 60 text files of 7MB each
  - Real model: 1360 text files of 7MB each
- Distribution of simulations on TGCC through submission scripts

The big challenge here lays on the complexity of the model.
Parsing big text files

- Each simulation produces:
  - Test model: 60 text files of 7MB each
  - Real model: 1360 text files of 7MB each

- Pandas parses files much faster than any other solution

```
In [4]: %timeit np.atleast_2d(np.loadtxt(conc_file, skiprows=1, usecols=[4]))
1 loops, best of 3: 981 ms per loop

In [5]: %timeit np.atleast_2d(pd.read_csv(conc_file, sep='\s+', usecols=['C[ppmV]'])).T
10 loops, best of 3: 59.1 ms per loop
```

- 16 times faster!
- But it introduces a dependency.... 😞
- Distributing file parsing

```
from sklearn.externals.joblib import Parallel, delayed
Results = Parallel(n_jobs=30)(delayed(func)(thing) for thing in iterator)
```
Handling and logging errors

- When submitting jobs through submission scripts, you loose track of the execution!

- Protect your wrapper with a try/except structure!

```python
class Wrapper(ot.OpenTURNSPythonFunction):
    def _exec(self, X):
        try:
            # Do stuff
            except Exception, e:
                logger.error(e, exc_info=True)
                raise e
        return Y
```

- Or use a decorator 😊!
  - More details at the end if need be
Handling output

- It is impossible to keep a numerical sample per simulation due to memory limits!
  - Primarily due to the fact that each run is an independent job submission

- Dump results to disk at the end for later post treatment.

- But one run represents ~300 mb per text files → ~1.8Gb....
  - Use gzip for compression and pickle.dump(array, file, protocol=2) for speed

```python
def dump_array(array, filename):
    with gzip.open(filename, 'wb') as fh:
        pickle.dump(array, fh, protocol=2)
```

- HDF5 and netCDF to be tested!

© Phimeca Engineering
Using argparse

- Usually we run our simulations from an Ipython interpreter...
- But on clusters you often need to go through submission scripts!
- Create a command line interface of the wrapper using argparse!

```python
• python wrapper.py -X 170 3 0.05
• python wrapper.py -MonteCarlo -N 1000

if __name__ == '__main__':
    import argparse
    parser = argparse.ArgumentParser(description="Python wrapper example." )
    parser.add_argument('-X', nargs=3, metavar=('X1', 'X2', 'X3'),
                        help='Vector on which the model will be evaluated')

    args = parser.parse_args()
    X = ot.NumericalPoint([float(x) for x in args.X])
    Y = model(X)
    dump_array(X, 'InputSample.pkl')
    dump_array(Y, 'OutputSample.pkl')
```
Conclusion

- **Probabilistic modelling** is used to describe uncertain release conditions.

- **Risk** is assessed as the probability of exceeding a critical dose.

- **Surrogate modelling** enables a drastic speed-up in the production of risk maps:
  - provided the CFD database is already computed (for industrial sites at risk);
  - 20 minutes per SLAM run in the DOE ($\times 100$ runs, but $\times \frac{1}{N_{CPUs}}$ using HPC);
  - about 12 seconds per time step for fitting the kriging predictors;
  - about 25 seconds per time step to predict the 10,000 configurations required for the final probability estimation.

- **Kriging** is a convenient surrogate for incorporating the uncertainty about the surrogate model in the final risk maps.

- Risk can be represented as time-varying maps of dose exceedance probabilities.
The killer wrapper!

- It is able to run on different environments:
  - Workstation
  - Office made heterogenous clusters (e.g., IPython parallel with SSH),
  - HPC through submission scripts (e.g., TGCC, Hyperion ou Poincare)
  - Cloud solutions (e.g. Simulagora ou DominoUp)

- It catches and logs errors for easy debugging

- It can either run or simply prepare runs
  - Usefull when using clusters

- You can use it as a script (argsparse module):
  - python wrapper.py -X 170 3 0.05

- It is by default an ot.NumericalMathFunction ! (decorators !)

- It might seem complex, but wrappers are repetitive. A good cookbook might be enough to spread this to the community!
Handling and logging errors

```python
from functools import wraps
def debug(func, logger):
    @wraps(func)
    def wrapper(*args, **kwargs):
        try:
            return func(*args, **kwargs)
        except Exception, e:
            logger.error(e, exc_info=True)
            raise e

    return wrapper

class Wrapper(ot.OpenTURNSPythonFunction):
    @debug(logger)
    def _exec(self, X):
        # Do stuff
        return Y
```

Take a look at David Beazley’s tutorial for PyCon’2013