L’optimisation mathématique appliquée à la sûreté incendie

Couplage CFAST-Optimiseur Python pour la réalisation d’études de sensibilités « intelligentes » et d’abaques de prédimensionnement

35èmes journées du Groupe du RésoFeux – 10 Juillet 2025

MACQUERON Corentin – Expert Modélisation thermique & Mécanique des fluides
WYGAS Benoît – Data Scientist / ML Engineer

Le besoin

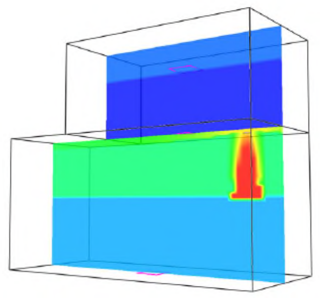
Le métier Modélisation réalise de nombreuses études de type « calcul » pour estimer la température de cibles de sûreté (structures métalliques, panneaux Lexan, etc.) en cas de feu d'équipements divers (armoires électriques, rétentions de liquides, etc.)

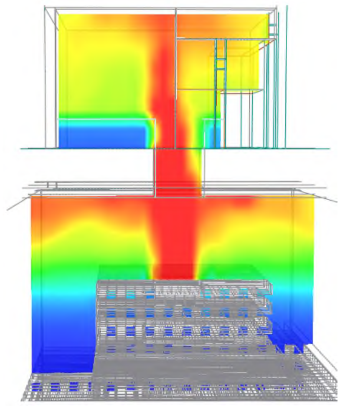
Ces études font intervenir de très nombreux paramètres « entachés » d'incertitude : fraction radiative, taux de suites, émissivités, débits de ventilation, taux de fuite, etc.

Ces paramètres peuvent avoir un impact très significatif et peuvent interagir entre eux. Il n'est pas trivial de déterminer le scénario le plus « pénalisant »

Il est donc nécessaire de réaliser des études de sensibilités

Nous utilisons principalement les logiciels CFAST et FDS du NIST, mais également un code « maison » appelé UltraFAST





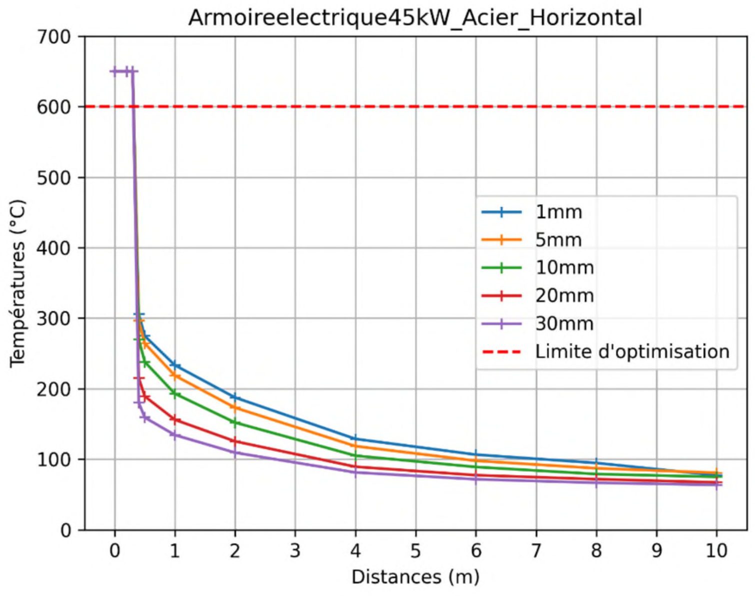
Le besoin

Le métier Sûreté sollicite le métier Modélisation pour des études de sûreté incendie avec des questions qui reviennent souvent, du type :

« Ma structure acier est à 3 mètres d'une armoire électrique en feu, ça passe ou pas? »

Il y a donc un besoin de créer des abaques pour les cas les plus fréquents pour assister et « autonomiser » le métier Sûreté en mode « prédim »

Ces abaques se « réduisent » à une température maximale en fonction d'une distance au feu. Ils doivent donc être enveloppes de l'intégralité du « champ des possibles » et requièrent de ce fait des études de sensibilités massives également



Exemple d'abaques

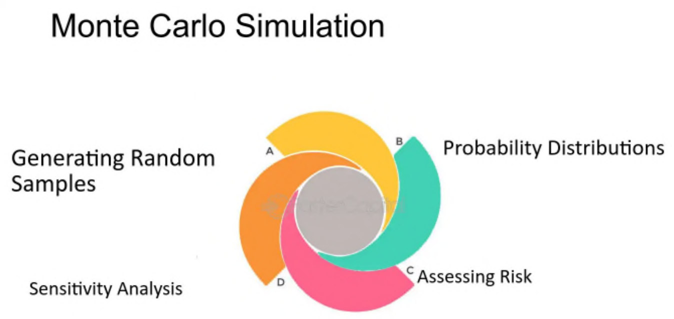
Le besoin

Ces deux besoins (études de sensibilité massives et abaques pénalisants) nécessitent de maîtriser des méthodes d’exploration fiables et robustes dans des espaces mathématiques de « grandes » dimensions

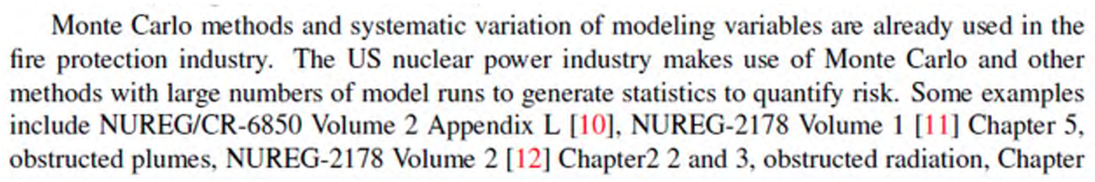
Les outils disponibles (CData du NIST par exemple) se basent usuellement sur des approches de type Monte Carlo telles que la recherche aléatoire

Cette approche n’est malheureusement pas « frugale » et peut donc nécessiter beaucoup de temps computationnel, et son caractère enveloppe est très incertain en grandes dimensions

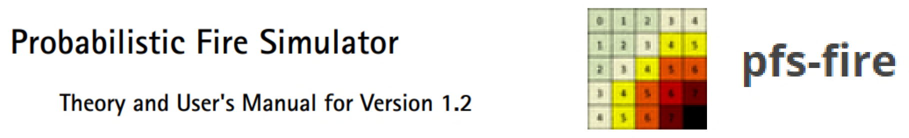
CData s’est par ailleurs révélé ne pas permettre de faire varier certains paramètres importants (fraction radiative par exemple)



Volume 5: CFAST Fire Data Generator (CData)


Monte Carlo methods and systematic variation of modeling variables are already used in the fire protection industry. The US nuclear power industry makes use of Monte Carlo and other methods with large numbers of model runs to generate statistics to quantify risk. Some examples include NUREG/CR-6850 Volume 2 Appendix L [10], NUREG-2178 Volume 1 [11] Chapter 5, obstructed plumes, NUREG-2178 Volume 2 [12] Chapter2 2 and 3, obstructed radiation, Chapter

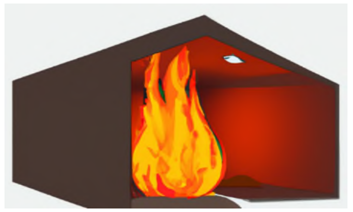
NIST Technical Note 1889 CFAST Fire Data Generator (CData)


Probabilistic Fire Simulator

Theory and User's Manual for Version 1.2

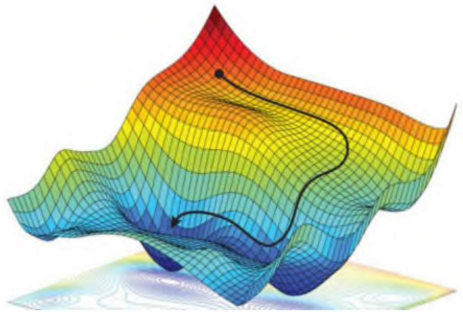
pfs-fire

Le besoin



Nous avons donc développé des approches de couplage d’algorithmes d’optimisation en Python avec des solveurs incendie comme CFAST et avons développé UltraFAST, un code incendie « maison » plus robuste et beaucoup plus rapide que CFAST en cas de sous-ventilation

Cette approche est usuellement beaucoup plus rapide en temps de calcul, et beaucoup plus efficace vis-à-vis du caractère enveloppe du résultat, les algorithmes utilisés étant en effet guidés « intelligemment » plutôt que « lâchés au hasard »



Il reste néanmoins bien évidemment impossible de démontrer mathématiquement que la configuration la plus pénalisante a réellement été trouvée, et quand bien même elle l’aurait été, les modèles numériques-physiques utilisés restent imparfaits

L’absolu n’existe donc pas, mais ce n’est pas une raison pour ne pas essayer de « maximiser » le scénario. Cette approche peut venir non pas en *remplacement* mais en *complément* d’une étude de type Monte Carlo

Cette philosophie a été partagée avec l’ASNR qui encourage l’approche

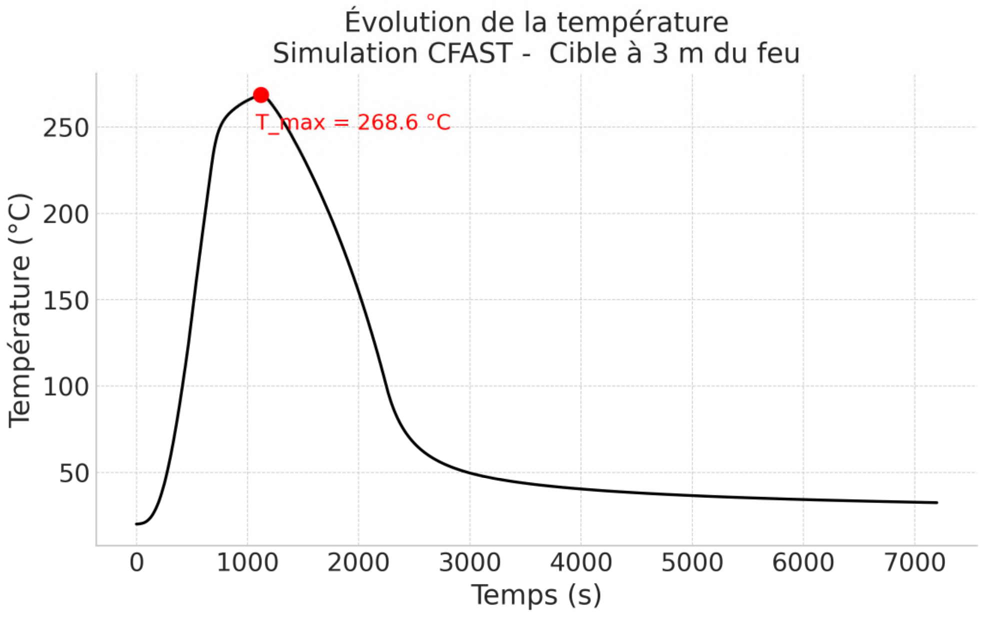
Méthodologie

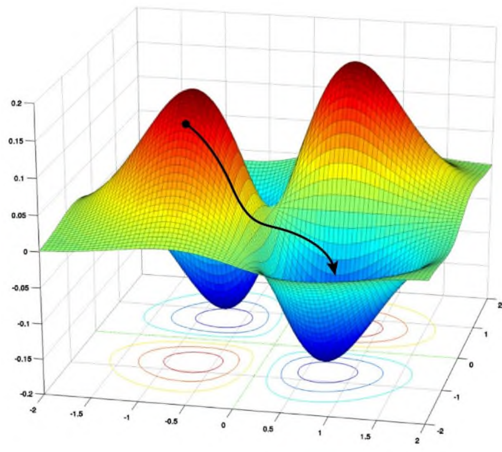
Les paramètres « certains » sont fixés (puissance du feu par exemple), puis on fait varier les nombreux autres paramètres d'entrée non « maîtrisés » (une trentaine)

On cherche à trouver les cas qui maximisent la température de la cible de sûreté

Exemple de paramètres :

```
"exterior_pressure": (101125.0, 101325.0),
"interior_humidity": (0.0, 100.0),
"lower_oxygen_limit": (0.10, 0.15),
"mat_emissivity": (0.7, 1.0),
"mat_thermal_conductivity": (0.57,3.0),
…
```



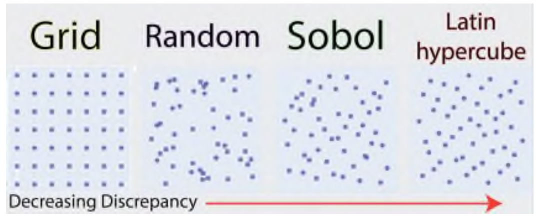


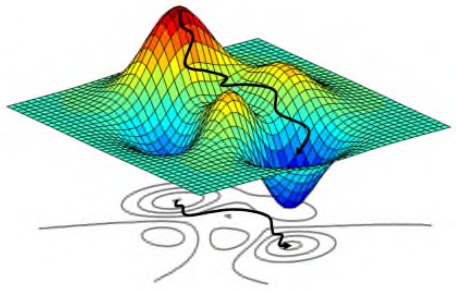
Outils

Pour les algorithmes d'optimisation, nous utilisons la librairie **SciPy** avec le module **scipy.optimize** et la librairie **pycma**

Pour le lancement des PEX aléatoires, nous utilisons la librairie **Numpy** avec la classe **numpy.random.Generator**

Nous utilisons également la classe **scipy.stats.qmc.LatinHypercube** pour générer des points par échantillonnage hypercube latin

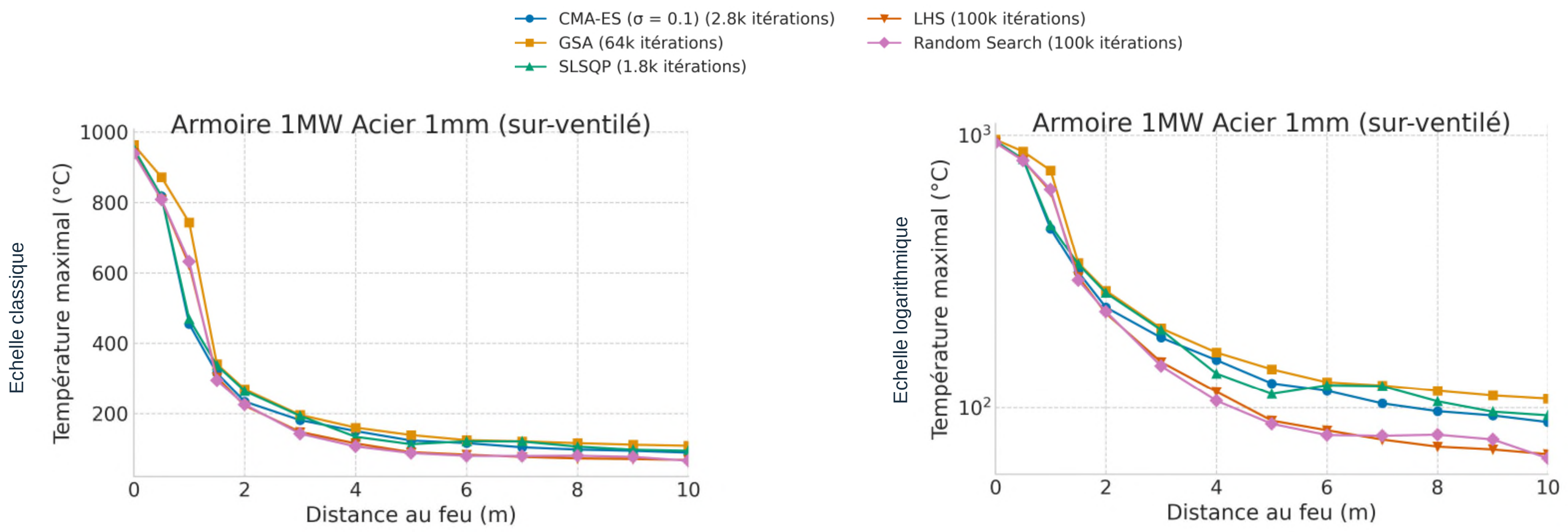




Résultats - CFAST

Les algorithmes d'optimisation permettent de trouver les configurations les plus pénalisantes avec moins d'itérations que les approches de type Monte Carlo.

Usuellement, l'approche « intelligente » requiert **~30 X** moins de temps computationnel (à puissance de calcul égal) et permet de trouver des températures de cible **26.5 %** plus élevées (en termes d'échauffement par rapport à la température initiale)



Algorithmes d'optimisation :

CMA-ES : Convolution Matrix Adaptation Evolution Strategy

GSA : Generalized Simulated Annealing

SLSQP : Sequential Least Squares Programming

Méthode de type Monte Carlo/ Quasi-Monte Carlo

LHS : Latin Hypercube Sampling

Random Search

Comparaison sur-ventilation & sous-ventilation

Sur-ventilé :

```
"comp_width_x": (10, 40),  # longueur du local
"comp_depth_y": (10, 40),  # largeur du local
"comp_height_z": (10, 40),  # hauteur du local
```

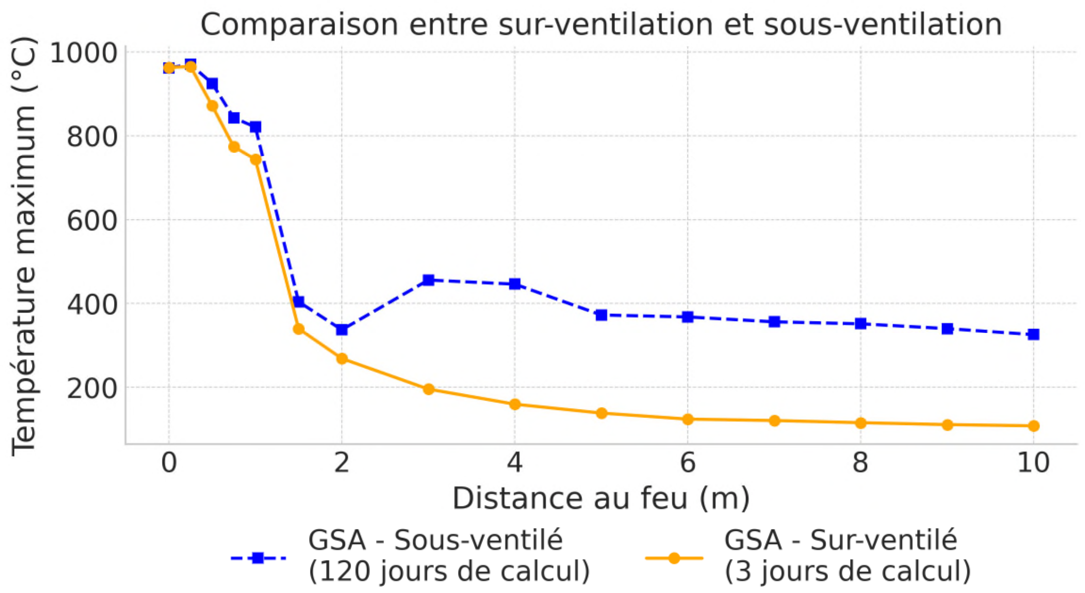
Sous-ventilé :

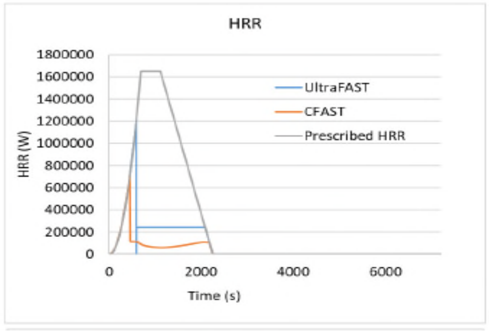
```
"comp_width_x": (3, 40),  # longueur du local
"comp_depth_y": (3, 40),  # largeur du local
"comp_height_z": (3, 40),  # hauteur du local
```

Les configurations les plus pénalisantes se trouvent (souvent) dans des cas de sous-ventilation

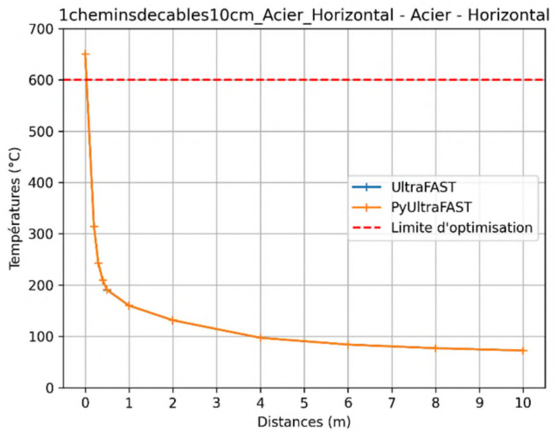
Problème : en sous-ventilation, le temps de calcul CFAST peut devenir très important (des semaines…)

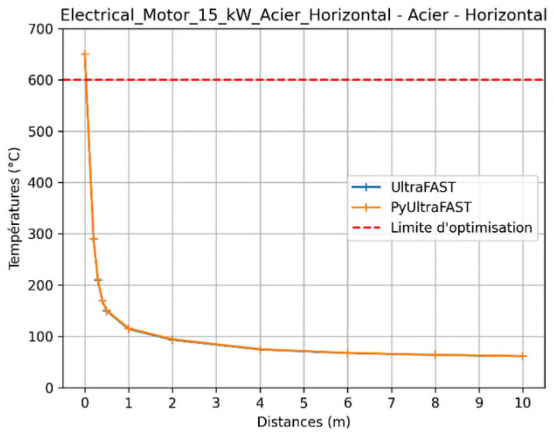
Nous utilisons donc un autre solveur incendie développé en interne à Orano : UltraFAST (tableur Excel à l'origine, converti en Python)





Conversion d’UltraFAST en Python (pyUltraFAST)





Exemple de KTESTs de « vérification » en utilisant SLSQP (déterministe) UltraFAST vs pyUltraFAST

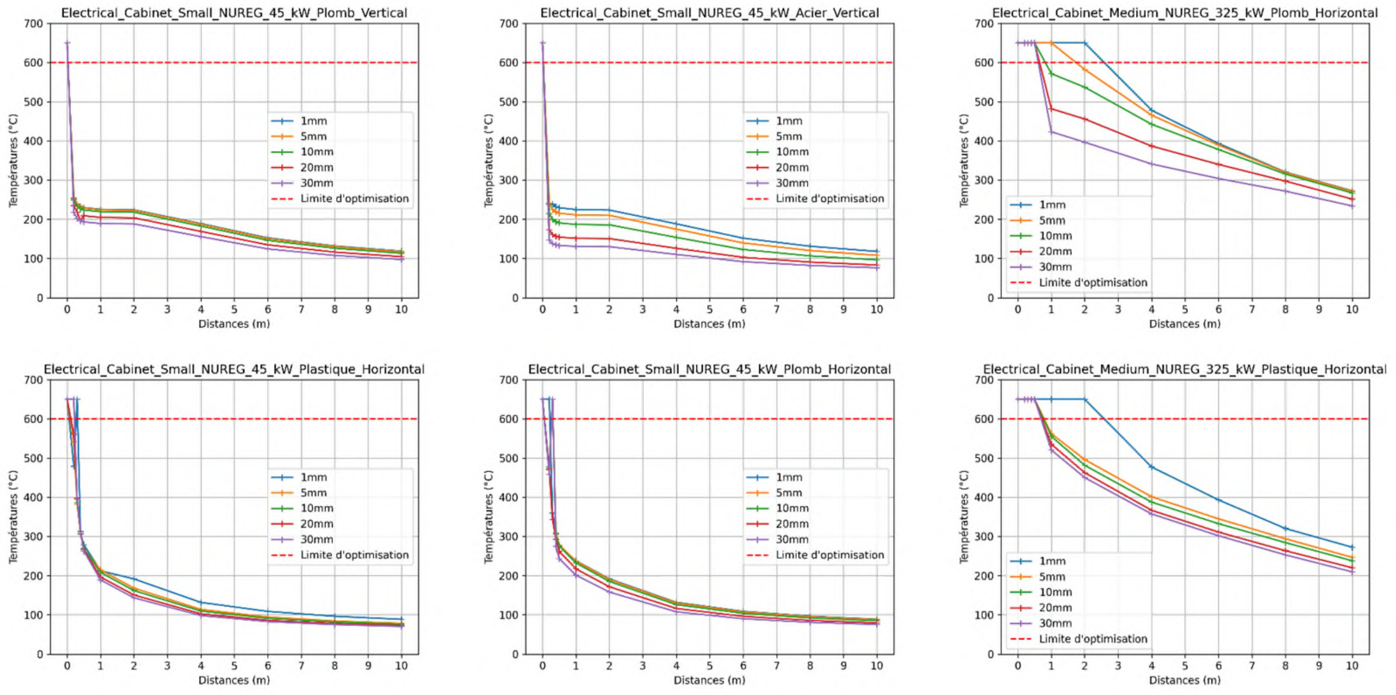
Avantages de la version Python :

- Parallélisation native sur serveurs HPC avec dask
- Algorithmes plus complexes utilisables

Résultats - pyUltraFAST

Ces abaques permettent de répondre rapidement aux questionnements du métier Sûreté, sans devoir lancer de nouvelles études à chaque fois.

Au total, près de **900 configurations** seront générées pour couvrir l’ensemble des cas demandé par le métier.



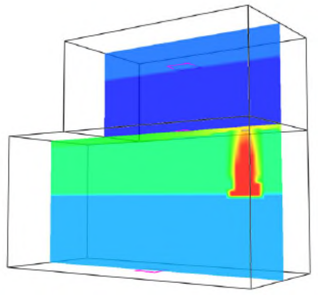
Conclusion

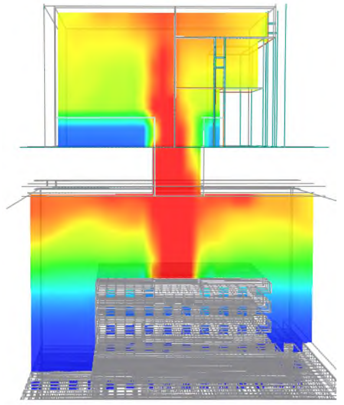
Orano a développé des outils permettant de réaliser des études de sensibilités « intelligentes » par algorithmes d'optimisation mathématique, permettant de trouver **plus rapidement** des configurations **plus sécuritaires** qu'avec les méthodes Monte Carlo aléatoires « classiques » pour ses études d'ingénierie incendie

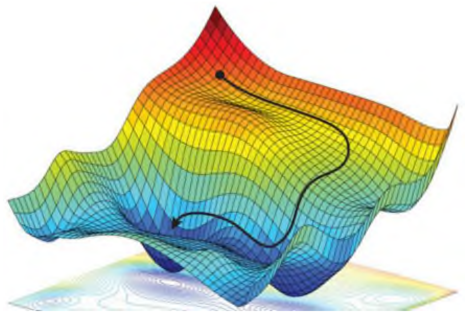
Les approches « intelligente » et « aléatoire » ne sont pas forcément exclusives mais potentiellement complémentaires pour renforcer la sûreté

Cette approche a été partagée avec l'ASNR qui encourage la démarche

Un « papier » sur le sujet a été rédigé et est disponible







Questions?

Worst case optimization applied to fire safety engineering with optimization algorithm

Benoît WYGAS[1] Corentin MACQUERON[2]

Orano Projets

{benoit.wygas, corentin.macqueron}@orano.group

Abstract

Investigating fire hazards is an essential aspect of nuclear safety. The fire risk modeling software, CFAST, is commonly used to conduct these studies. Traditional approaches often involve stochastic methods such as Monte Carlo analysis in conjunction with CFAST (CData) for fire safety research and design. This article introduces a novel approach that integrates optimization algorithms within CFAST to identify penalizing configurations that maximize target's surface temperature. Our primary focus is on Nelder-Mead and SLSQP algorithms. Results indicate that optimization methods can deliver comparable or superior performance to stochastic methods in certain cases, especially in our last scenario where a significant relative heating ratio was observed between the stochastic and optimization methods. Therefore, optimization methods can serve as a valuable supplement to stochastic methods in fire safety research. Using both method is crucial in the context of nuclear engineering.

Keywords: Fire Safety, Nuclear plant, Optimization, Stochastic process, CFAST, CData, Latin hypercube sampling

1 Introduction

In the context of nuclear engineering, fire hazards represent a significant risk. They are likely to trigger incidents or critical situations and may also lead to the dispersion of harmful substances into the environment. Aspen Plus, a chemical process simulator, has a built-in optimization module based on sequential quadratic programming [2]. Moreover COMSOL has a programming structure based in MATLAB that allow to use genetic algorithm for multi-objective optimization [11]. Because of the presence of flammable materials and various heat sources in nuclear plants, fire safety is given special attention. Nuclear regulations impose that safety studies for nuclear facilities must use modeling software specifically qualified and certified for this purpose.

At Orano, the two primary fire risk modeling software that are qualified are FDS (Fire Dynamics Simulator) [6] and CFAST [9] (Consolidated Fire and Smoke Transport). This study will focus on CFAST software, comparing various methods to identify the most penalized cases in a given scenario. Identifying these cases is vital for designing nuclear plants, as they must comply to the safety margins and sizing criteria set forth by regulatory standards. Here, we present a novel optimization method for identifying the worst-case scenario within a large configuration space. In this comparative study, we found that using optimization techniques, instead of standard stochastic methods, can yield comparable or superior results.

2 Methods

The methods that have been proposed to find penalizing cases can be split in two main categories, stochastic processes techniques and optimization algorithm. The methods that are based on stochastic processes like Monte Carlo involves running lots of simulation. The second approach is a ”smarter” way to obtain penalizing cases by using optimization algorithm that will minimize a objective function that is wrapping the CFAST model.

2.1 Stochastic methods

Stochastic processes inherently require a substantial number of inputs or trials; in general, the results tends to be more accurate as the number of trials increase. In our context, nothing ensures that we reached the most penalizing cases when using such a random sampling. To find the maximal surface temperature, our methodology needs to include a search strategy that thoroughly investigates the possible outcomes.

The CFAST Fire Data Generator (CData) is as software that has been developed to help users do Monte Carlo analyses. Initially CData is made to obtain data from common devices and sensors, such as heat and smoke detectors, within any chosen building environment and run Monte Carlo Analysis. CData comprises four key components: a basic input handler, a distribution function module, a sampling module, and a simulation engine.

Our internal tool, Fire Design of Experiments (FireDoE), has a strong resemblance to CData because it allow us to conduct multiple CFAST experiments with specified ranges for input parameters. The initial 1,000 runs of FireDoE are dedicated to a basic sensitivity analysis, while the subsequent thousands last enable us to conduct Latin Hypercube Sampling. Between these two phases is a phase of random uniform sampling, similar to the capability provided by CData.

Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) is a statistical method used to generate a sample of plausible collections of parameter values from a multidimensional distribution. The technique is an extension of stratified sampling which ensures that each interval of the distribution of the input variables is equally represented. In LHS, the range of each input variable is divided into N non-overlapping intervals; from each interval, a value is sampled at random. The sampled values for each variable form an N-dimensional matrix, where each row corresponds to one stratified sample and each column to a variable. This matrix is then randomly permuted to produce a sample that is representative of the overall distribution [7].

One of the main advantages of LHS is its ability to efficiently sample from high-dimensional spaces, ensuring that each sample is as informative as possible, which is particularly beneficial when the output is significantly affected by only a few input variables. This method provides a more comprehensive and robust representation of the input parameter space compared to simple random sampling, especially when dealing with complex models where certain inputs may have a disproportionate impact on the output.

Another tool developed for computation of the distributions of fire model like CFAST, named Probabilistic Simulation of Fire Scenarios (PFS), [5] is also available. However, it has not been updated to be compatible with the latest version of CFAST.

2.2 Optimization algorithms

Optimization algorithms constitute a fundamental aspect of computational science, aimed at identifying the most effective solutions within a defined set of constraints or bounds. These algorithms are pivotal in systematically exploring and determining optimal solutions in various problem spaces, where the objectives and constraints can vary significantly. Their application spans numerous scientific and engineering disciplines, ranging from operational research and logistics to machine learning and systems engineering. The essence of these algorithms lies in their methodical approach to problem-solving, whether it involves minimizing costs, maximizing efficiency, or finding the best possible configuration among a set of options.

More formally, optimization involves the search for the minimum (or maximum) of a specific quantity, called cost or objective function. Optimization problems can be write in the following general form:

$$(P) \quad \inf_{x \in K} J(x).$$

Where J is a function mapping from $\mathbb{R}^N$ to $\mathbb{R}$ and x a vector containing N real variables of the following form $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$. Generally, the variables $x_1, \ldots, x_N$ satisfy constraints represented by a subset $K \subseteq \mathbb{R}^N$. A solution to problem (P) exists if we can find a set of variables $x_0 \in K$ such that

$$\forall x \in K \quad J(x_0) \leq J(x).$$

In such cases x_0 is called a minimizer (or point of minimum) of J over K, and $J(x_0)$ is considered the minimum value of J over K.

For the purposes of this paper, our focus will be on optimization methods such as the Sequential Least Squares Programming Method and the Nelder-Mead algorithm.

Sequential Least Squares Programming Method

This technique is used on mathematical problems for which constraints and the objective function are two fold continuously differentiable. In SLSQP, each iteration involves solving two subsidiary problems: a linear program (LP) and equality-constrained quadratic program (EQP) here, LP used to determine an active set and EQP used to compute the total step. It is able to process constraints on the problem, so for the exhibition of the algorithm, study a general nonlinear programming:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

Subject to

$$\begin{aligned} g_j(x) &= 0, \quad j = 1, \ldots, k, \\ g_j(x) &\geq 0, \quad j = k+1, \ldots, m, \\ x_l &\leq x \leq x_u. \end{aligned}$$

Where m represents the number of equality and inequality constraints and k is the number of equality constraints. This constrained optimization problem may consistently be reformulated using the Lagrangian method and the Lagrange function $L(x, \lambda)$:

$$
\begin{aligned}
L(x, \lambda) &= f(x) - \sum_{j=1}^{m} \lambda_j g_j(x) \\
\min_{x,\lambda} L(x, \lambda) &\quad \text{such that} \quad x_l \leq x \leq x_u.
\end{aligned}
\tag{2}
$$

In eq. (2) function $f(x)$ is the objective function which left out for the minimization problems, where it is constant.

Nelder-Mead Algorithm

The Nelder-Mead algorithm, introduced by J. A. Nelder and R. Mead, is a well-known method for optimizing multidimensional functions without constraints. It operates through a simplex, a geometric figure analogous to a triangle in higher dimensions, which adapts and moves through iterative steps—reflection, expansion, contraction, and shrinkage—seeking the function's minimum value [8]. The strength of this method lies in its simplicity and the fact that it does not require derivative calculations, making it ideal for functions where derivatives are unavailable or hard to determine. Despite being less advanced than other optimization methods, its straightforwardness keeps it in active use for various applied mathematics and engineering problems.

Regarding the selection of these algorithms, they have been chosen for their documented efficacy. While there is a wide array of optimization methods available, our intention is not to provide an exhaustive review of every available option. Instead, our focus is to show how these particular methods, when applied in conjunction with CFAST, can effectively identify penalized scenarios in a manner similar to stochastic approaches.

2.3 Experimental Design

In each experiment, the objective is to identify the most penalized cases, specifically those where the target surface temperature (TRGSURT) is maximized. 11 targets will be strategically positioned on different location within the compartment.

The experiment design includes both fixed parameters (detailed in the Appendix under "Fixed Parameters") and variable parameters (listed in the Appendix under "Variable Parameters") which will variate within predefined ranges. The experiments are divided into two scenarios based on compartment size: the first scenario involves a larger compartment of 1800 m^3, while the second scenario focus on a smaller compartment of 120 m^3. For each scenario, we employ both stochastic methods and a variety of deterministic optimization algorithms to conduct the experiments. The outcomes will be systematically compiled into a table for comparative analysis.

The initial experiments will be conducted using a simplified model of one of the larger compartments (1800 m^3). This simplification was necessary because CData, the software we use for running multiple CFAST simulations, imposes certain limitations on our experiments. For instance, CData does not support variations in the fire Froude number.

2.3.1 Fire characterization

The Heat Release Rate (HRR), measured in kW, represents the speed at which heat is generated by combustion. It stands as the most crucial yet challenging parameter to predict in assessing fire consequences. This is because HRR fundamentally influences the estimation of conditions within fire-induced phenomena, including plumes, ceiling jets, smoke layers, and flame radiation[1].

In general, the HRR is calculated using flammability properties of the fuel. This approach is easily applied in the case of liquid and some plastic combustibles. These flammability parameters are the heat of combustion (ΔH_c, kJ/kg), and the specific burning rate ($\dot{m}''$, kg/sec-m^2). Using these two parameters, along with the burning area, the HRR by the fire is estimated as:

$$\dot{Q}_f = \Delta H_c \cdot \dot{m}'' \cdot A \tag{3}$$

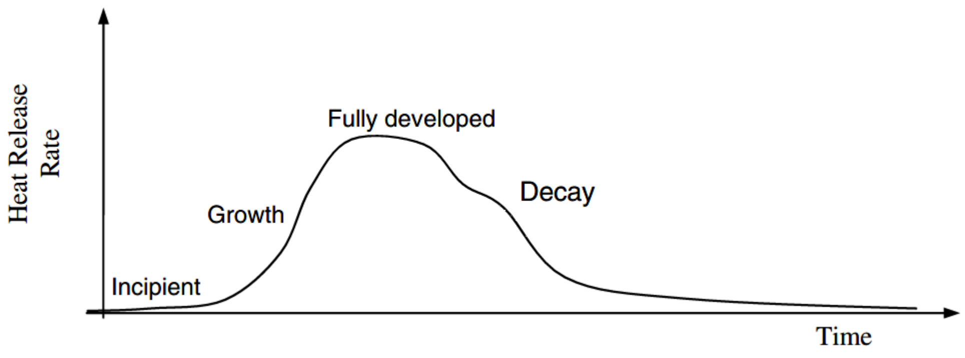


Figure 1: Heat Release Rate Profile

These four stages are not observed in all fires and do not need to be included in every analysis, as long as the modeled fire conditions capture the fire hazards posed by the ignition source to the target. The following list provides a few general notes on developing HRR profiles.

1. The incipient stage is not usually included due to its uncertainty in duration and that it is not expected to generate thermal conditions that threaten the integrity of other targets in the room.
2. A fire may run out of fuel before reaching its peak HRR (steady burning stage). If the amount of combustible is known, the analyst may choose to model the growth stage until all the fuel is consumed.
3. Oxygen availability can affect the fire after it starts to grow. A fire may decay due to low oxygen concentrations. In contrast, an additional supply of oxygen to an already low oxygen concentration room may cause the fire to increase in intensity.
4. HRR profiles can be affected by combustible layouts in the room. Therefore, it is not always easy to develop or find a profile for a specific situation. If a constant HRR profile is selected, the constant value should be the peak fire intensity.
5. Temperature and heat flux results associated with the decay stage of the fire will, in general, suggest less hazardous conditions than the growth of the fully developed stage. Once the fire starts to decay, temperatures in the room will decrease to ambient conditions. As a result, and depending on the objectives of the simulation, modeling the decay stage of the fire usually does not provide critical information in support of risk decisions.

In our experiments, we aim to characterize electrical cabinet fires based on guidelines from NUREG (Electrical Cabinet Large NUREG 1000 kW)[1]. The recommended HRR profiles for these cases have been established by EPRI/NRC-RES.

	Time to Peak (minutes)	**Steady Burning (minutes)**	**Time to Decay (minutes)**
Average	11.4	7.1	19

Table 1: HRR Profiles for Electical Cabinet Fires

A t^2 function can be use for representing the growing phase of the fire. The t^2 function is expressed in the form:

$$\dot{Q}(t) = \min\left\{\dot{Q}_{\text{peak}}, \dot{Q}_{\text{peak}}\left(\frac{t}{\tau}\right)^2\right\} \text{ (kW)}, \tag{4}$$

where τ is the time to reach the peak HRR, $\dot{Q}_{\text{peak}}$ is the peak HRR, and t is the current time.

We use a linear function for representing the decay stage phase of the fire. While a t^2 function could also replicate the degrowth phase, opting for a linear approach allows us to adopt a more conservative representation.

$$\dot{Q} = \dot{Q}_{peak}(\frac{t}{\tau})^2 \tag{5}$$

Our cabinet fires are modeled as a mixture of Polyvinylchloride (PVC), Polymethylmethacrylate (PMMA) and Polyethylene (PE). These polymers have the following composition [10] :

Polymer	**Composition**
Polyvinylchloride, PVC	$CH_{1.5}Cl_{0.50}$
Polymethylmethacrylate, PMMA	$CH_{1.6}Cl_{0.40}$
Polyethylene, PE	CH_2

Table 2: Composition of polymer used for cabinet fire

To ensure the mass fraction of each polymer in the mixture (Polyvinylchloride [PVC], Polymethylmethacrylate [PMMA], and Polyethylene [PE]) sums to 1, we use random draws between 0 and 1 for each polymer, adjusting to achieve a total sum of 1. The chemical composition of the fire is then determined by the weighted average of these fractions. In CFAST simulations, fuels are modeled as hydrocarbons composed of carbon, hydrogen, oxygen, nitrogen, and chlorine. For the purpose of characterizing electrical cabinet fires, which are considered mixtures of PVC, PMMA, and PE, we use the following equation as input parameters for CFAST:

$$\begin{gathered} Hydrogen = 1.5PVC + 1.6PMMA + 1.9PE \\ Oxygen = 0.4PMMA \\ Chlorine = 0.5PVC + 0.13PE \end{gathered}$$

2.3.2 CFAST workflow with optimizer

The workflow for coupling CFAST with an optimizer involves several key steps. Initially, we define the bounds and the initial vector for our input parameters. Next, feature scaling is applied to normalize these parameters between 0 and 1. This normalization is essential to prevent bias in the optimizer's performance. After scaling, all parameters are fed into the optimizer. Within our objective function, we first revert the scaling of parameters to their original scale and then calculate specific physics-related parameters. Specifically, this phase includes determining the leak area A_{leak} based on the pressure difference ΔP, the input flow rate Qv_{in} and the output flow rate Qv_{out}. This approach ensures the optimizer operates efficiently along the fire model.

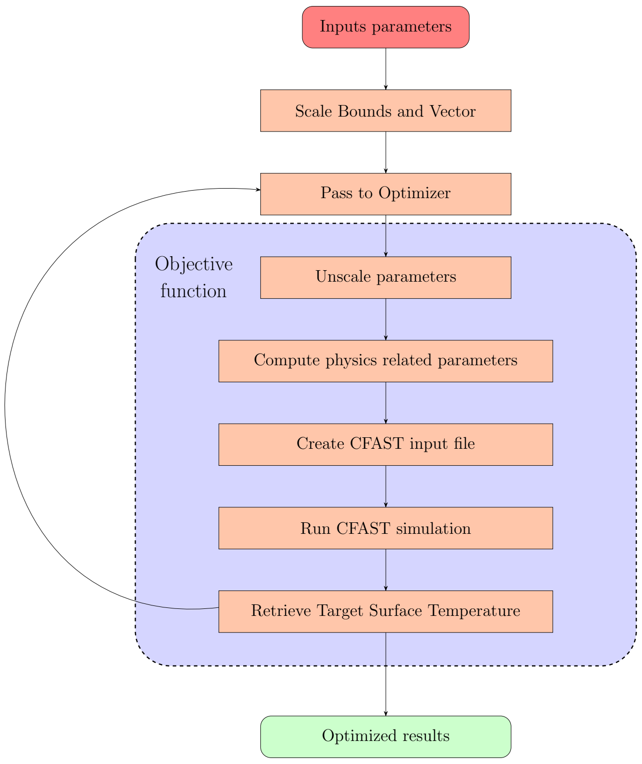


Figure 2: CFAST workflow along optimizer

From Bernoulli's principle :

$$\Delta P = \frac{1}{2}\rho\xi v^2$$
$$\text{with } \rho = 1.205\ kg \cdot m^{-3} \text{ density of dry air at } 293,15K$$
$$\xi = 2.025$$
$$v = \frac{Qv_{out} - Qv_{in}}{A_{leak}}$$

$$A_{leak} = \sqrt{\frac{-\rho\xi(Qv_{out} - Qv_{in})^2}{2\Delta P}} \quad \text{with} \quad \Delta P \in [-200, -20] \tag{6}$$

We also compute the fire area using the Froude Number (Zukowski) formula.

$$\dot{Q}^* = \frac{\dot{Q}}{\rho_\infty c_p T_\infty \sqrt{gD}D^2} \quad \text{and} \quad A_{fire} = \frac{\pi D^2}{4}$$
$$\text{with } \rho = 1.205\ kg \cdot m^{-3} \text{ density of dry air at } 293,15K$$
$$c_p = 1.007\ kJ \cdot K^{-1} \cdot kg^{-1} \text{ heat capacity of dry air at } 293,15K$$
$$T_\infty = 293.15\ K \text{ and } g = 9.81\ m \cdot s^{-2}$$

$$A_{fire} = \max\left\{\frac{\pi}{4}\left(\frac{\dot{Q}}{\dot{Q}^*\rho_\infty c_p T_\infty\sqrt{g}}\right)^{\frac{4}{5}}, 0.1\right\} \tag{7}$$

Within the "Compute physics-related parameters" step, we undertake more complex operations, including recalculating an approximation of the true Heat Release Rate (HRR) that is computed during the simulation. This implies that before launching a simulation of CFAST, HRR is recalculated. Since fire area is linked to HRR $\dot{Q}$ from eq. (7), fire area is also changed before launching the CFAST simulation. The approximation made on HRR is based on oxygen limit which is set at 0.15.

By recalibrating the input parameters, we achieve simulations that are more realistic. Below is a graphic comparison of the expected Heat Release Rate (HRR) and fire area against the default values. This approach enhances the accuracy of our simulations, ensuring that our models are more close of real-world scenarios.

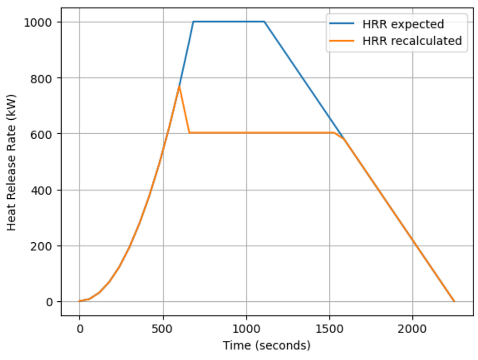


Figure 3: HRR comparaison

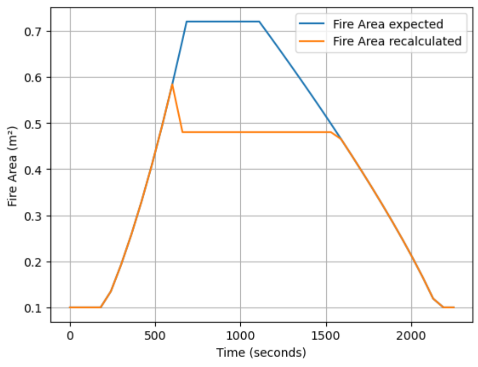


Figure 4: Fire Area comparaison

Our study is still subject to certain limitations arising from these specific assumptions, previously listed: For instance, eq. (7) assumes that the areas considered are equal to or greater than 0.1 m^2 and that the ambient temperature T_∞ is maintained at a constant 293 K throughout the simulation. These conditions may not reflect a real-world scenarios, potentially affecting the applicability of our findings across different settings or scales.

2.3.3 Hardware and software specifications

The simulations are executed on a computer equipped with an Intel Xeon CPU E5-2697 v3 @ 2.60GHz and 256GB of RAM. We used version 7.7.3 of CFAST for our analyses and the version 3.8 of Python, with various libraries. These include Scipy, which provides the scipy.optimize.minimize function for optimization tasks, as well as scikit-learn, pandas, and numpy for efficient data manipulation.

3 Results and discussion

The first case is designed for compatibility with all available methods and to ensure relatively quick computation times; thus, it represents a simplified version of one of our compartments. This simplification is imposed by the limitations of CData, which does not support the complex operations previously detailed. As a result, only 12 parameters will vary in this experiment, chosen for their simplicity and the feasibility of their variation within the constraints of our modeling tools.

3.1 Simplified large compartment (1800 m^3)

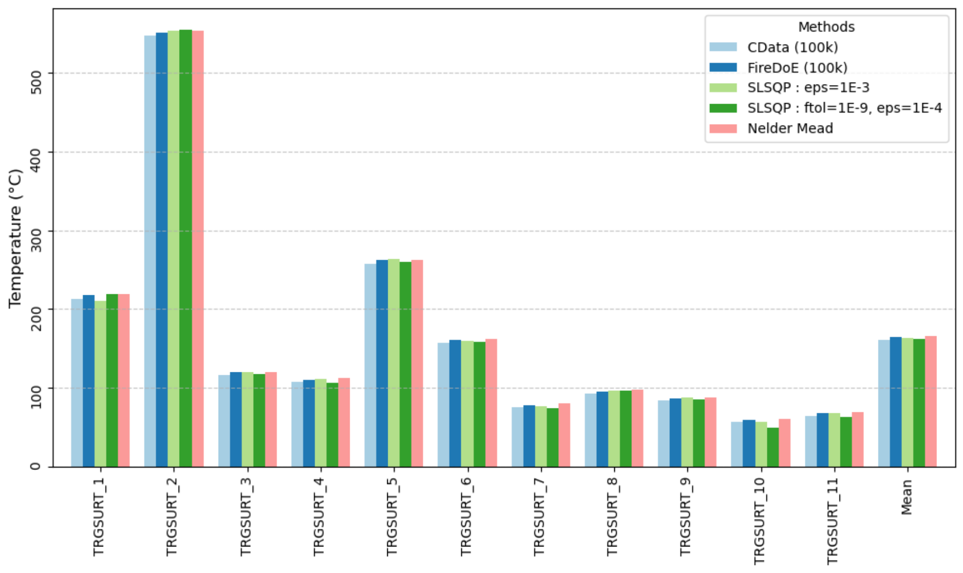


Figure 5: Large compartment (1800 m^3, 12 parameters), Target Surface Temperature Maximum

In our comparison within a straightforward scenario, the differences in results were minor, suggesting comparable performance by both methodologies under basic conditions. Nevertheless, we expect that by raising the number of variable parameters, the distinctions between these methods will be more evident. Besides results founds are dependent of the algorithm itself and their parameters.

Methods	Time (seconds)
CData (100k)	51 660
FireDoE (100k)	29 220
SLSQP: eps=1E-3	1 579
SLSQP: ftol=1E-9, eps=1E-4	4 160
Nelder Mead	19 270

Table 3: Large compartment (1800 m3, 12 parameters), elapsed time for computing

Both CData and FireDoE launch the same amount of simulations. However, CData requires a longer duration to process 100,000 cases due to its consistent operation limited at 50% CPU load. This limitation becomes more pronounced when the RAM usage reaches a quarter of the total capacity, resulting in Fortran errors for some caes. In contrast, FireDoE's, through manual configuration CPU usage was set to not exceed 80%.

In terms of speed results revealed that all of the optimization methods outperformed random search strategies (such as CData and FireDoE). However, the Nelder-Mead algorithm, despite its performance, had relatively longer execution times. Based on the criterion of achieving the optimal balance between value and time efficiency, the Sequential Least Squares Programming (SLSQP) method, with an epsilon value of 1E-3, was selected for the next scenario. This decision is made based on SLSQP's superior performance ratio of value to time.

3.2 Large compartment (1800 m^3)

The second case maintains a configuration similar to the first one but introduces more complex parameters for variation, such as the fire Froude number, fire area, and specific constraints that were not feasible with CData. These include maintaining a final pressure within a restricted range of -20Pa to -200Pa and adjusting the chemical composition of the fire, as previously discussed. This approach allows for a deeper exploration of the effects of these complex parameters on the simulation outcomes, offering insights into their impact and the overall behavior of the fire under varied conditions.

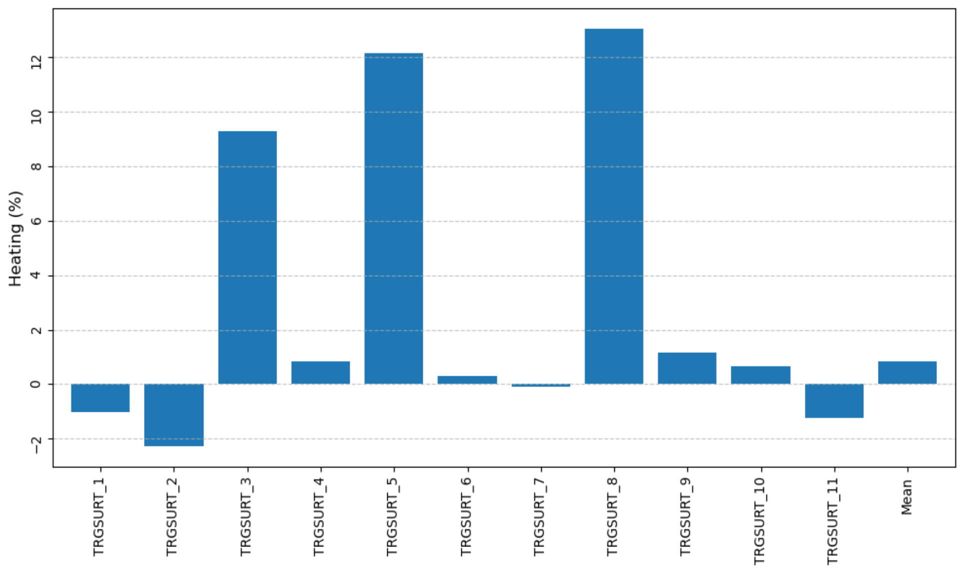


Figure 6: Large compartment relative heating in (%) between FireDoE and SLSQP

As in the first scenario, differences between both methods are not that pronounced, but for some specifiv target like TRGSURT_5 and TRGSURT_8 we obtain a relative heating above 12%. In average SLSQP method perform slightly better while being faster to compute.

Methods	Time (seconds)
FireDoE (100k)	25 920
SLSQP: eps=1E-3	7 363

Table 4: Large compartment (1800 m3, 24 parameters), elapsed time for computing

3.3 Small compartment (120 m^3)

Last compartment is a smaller version of the 1800 m^3 compartment with about the same inputs parameters but dimensioned for this volume. For example the input/output flow rate in m^3 has been lowered to stay between 1 to 10 volume of the compartment per hour.

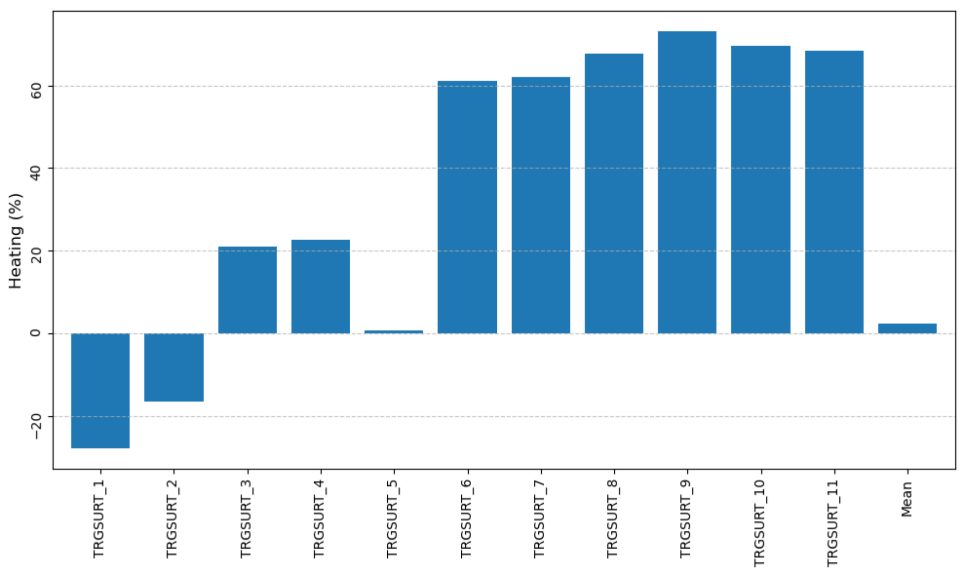


Figure 7: 120 m^3 compartment relative heating in (%) between FireDoE and SLSQP

Results obtain with the first target TRGSURT_1 and TRGSURT_2 are higher when using FireDoE. But overall when considering others target we observe a significantly higher maximum temperature when using the SLSQP method. This difference becomes even more pronounced when we examine the relative heating across all targets.

Methods	Time (seconds)
FireDoE (100k)	33 696
SLSQP: eps=1E-3	214 201

Table 5: Small compartment (120 m3, 12 parameters), elapsed time for computing

Moreover, the maximum value for TRGSURT_1 is not the maximum that SLSQP algoritgm was able to find because the optimization process took way too much time 3.3. Nevertheless, SLSQP method was very long to compute because CFAST was most of the time in under-ventilated cases.

Latin Hypercube Sampling

For stochatics methods, results reveal the efficiency of Latin Hypercube Sampling (LHS) in the 1800 m^3 scenario, where half of the target values achieve their maximum within the LHS area, showcasing LHS's effectiveness in capturing significant regions of interest with fewer samples.

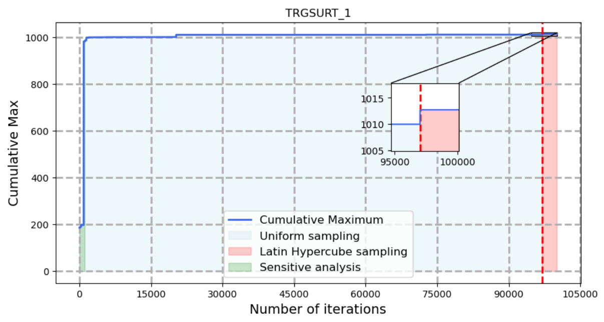


Figure 8: Cumulative maximum for TRGSURT_1 (1800 m^3 compartment)

However, in the second scenario, LHS does not perform as well. Since none of the target reach their maximum value within the LHS area.

Sensitive analysis of Zukoski number

We also conduct a small sensitive analysis on the inputs parameters that will variate, specially on Zukoski number wich is a parameters that is not available from directly from CFAST inputs.

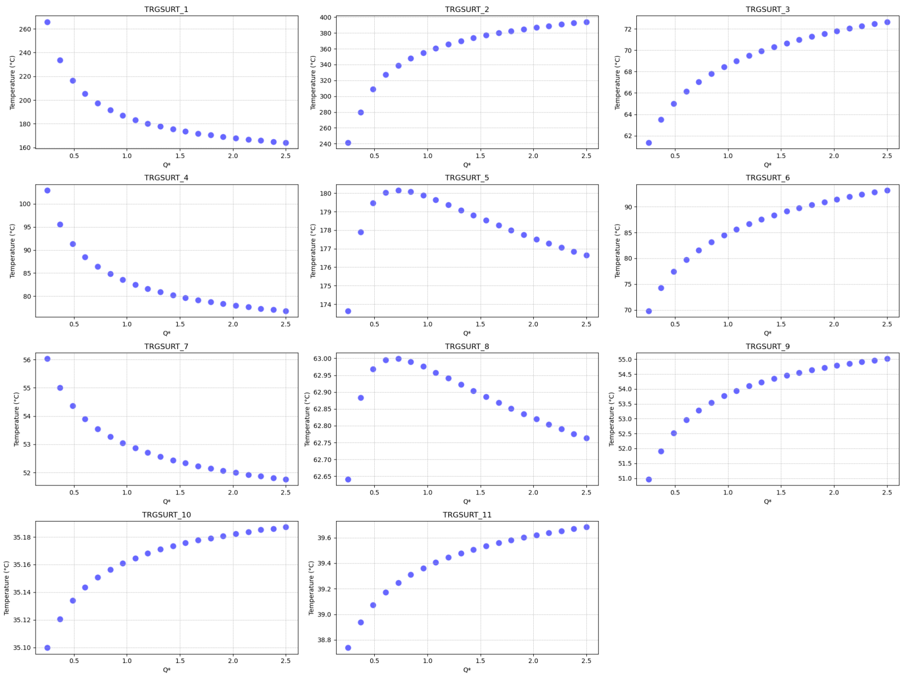


Figure 9: Zukokski influence over target temperature

While there is limit to this sensitive analysis because this method can't identify interactions between variables since it varies only one factor at a time. The main point here is not to understand deeply how zukoksi number influence the target outputs but rather it is to know or not if this parameters has an influence or not and as this small sensitive analysis show, the zukoksi number can have an influence over the target temperature.

Conclusion

Optimization methods can be seen as a valuable complement to stochastic methods in identifying penalized cases. A significant drawback of these methods is their iterative nature, which depends on previous steps to compute the next one. Therefore, if they encounter a scenario that requires an extensive computation time, they may never finish, unlike stochastic methods where a maximum runtime is typically set. If the process doesn't complete within this timeframe, that scenario can be discarded and another sample taken. Nevertheless, optimization methods have the capacity to identify penalizing scenarios that are often significantly more severe than those detected by stochastic methods. Using both methods for determining uncertain parameters is particularly critical in the field of nuclear engineering.

A way to bypass CFAST run would be to create a machine learning model that can predict CFAST outputs. This approach has already been validated for certain outputs such as ULT (Upper Layer Temperature) [13] [12]. However, it has not yet been proven effective for predicting the target surface temperature. Furthermore, only SLSQP and Nelder-Mead algorithms have been tested, and our initial scenario demonstrates that the results obtained are dependent of the algorithm and its parameters. Applying global optimization algorithm such as simplicial homology algorithm (shgo) [3] could be highly beneficial in this context, but also very time consumming.

Appendix

Numerical value

Large compartment (1800 m^3) simplified, 12 parameters					
Target	CData (100k)	FireDoE (100k)	SLSQP : eps=1E-3	SLSQP : ftol=1E-9, eps=1E-4	Nelder Mead
TRGSURT_1	212.987	217.773	209.907	219.32	219.138
TRGSURT_2	547.118	551.349	553.879	554.222	553.303
TRGSURT_3	116.801	120.022	119.71	118.087	119.86
TRGSURT_4	107.49	110.657	111.803	106.678	112.828
TRGSURT_5	257.066	262.1	263.454	259.87	262.891
TRGSURT_6	157.217	160.66	159.395	158.26	161.977
TRGSURT_7	76.1434	78.0673	76.306	73.8142	81.1081
TRGSURT_8	92.9357	95.9317	95.9949	97.1132	97.793
TRGSURT_9	83.9596	86.9319	88.506	85.6749	88.513
TRGSURT_10	56.3953	59.6929	57.3668	49.1089	60.5492
TRGSURT_11	64.774	67.9563	68.0907	62.9191	68.9103
Mean	161.171545	164.649191	164.037491	162.278845	166.079145

Large compartment (1800 m^3), 24 parameters		
Target	FireDoE (100k)	SLSQP : eps=1E-3
TRGSURT_1	1012.73	1002.19
TRGSURT_2	1017.7	994.653
TRGSURT_3	119.093	130.145
TRGSURT_4	318.324	320.945
TRGSURT_5	285.139	319.767
TRGSURT_6	317.702	318.685
TRGSURT_7	116.282	116.16
TRGSURT_8	102.657	116.033
TRGSURT_9	115.251	116.569
TRGSURT_10	46.4505	46.7514
TRGSURT_11	52.2725	51.6256
Mean	318.509182	321.229455

Small compartment (120 m^3), 24 parameters		
Target	FireDoE (100k)	SLSQP : eps=1E-3
TRGSURT_1	1145.6	826.934
TRGSURT_2	1177.77	982.32
TRGSURT_3	454.461	549.258
TRGSURT_4	222.804	272.96
TRGSURT_5	197.697	198.878
TRGSURT_6	121.413	195.649
TRGSURT_7	110.242	178.578
TRGSURT_8	114.428	191.84
TRGSURT_9	111.134	192.327
TRGSURT_10	113.732	192.9
TRGSURT_11	113.807	191.684
Mean	353.008	361.211636

Variate Parameters for Bounds Optimization

The following table lists the input parameters, their respective bounds, and the initial value used in the bounds optimization process:

Large compartment (1800 m^3) simplified (12 parameters)			
Inputs parameters	Inital value (x_0)	Bounds (min, max)	Units
PRESSURE	101325.0	(101125.0, 101325.0)	Pa
RELATIVE_HUMIDITY	0.5	(0.0, 1.0)	%
MATERIAL_EMMISIVITY	0.9	(0.7, 1.0)	
MATERIAL_CONDUCTIVITY	1.75	(1.0, 3.0)	$kW/(mK)$
MATERIAL_SPECIFIC_HEAT	870E-3	(700E-3, 1121E-3)	$kJ/(kgK)$
MATERIAL_DENSITY	2300	(2000, 2500)	kg/m^3
TARGET_EMMISIVITY	0.9	(0.2, 1.0)	
TARGET_CONDUCTIVITY	15.0	(14.0, 23.0)	$kW/(mK)$
TARGET_SPECIFIC_HEAT	500E-3	(475E-3, 582E-3)	$kJ/(kgK)$
TARGET_DENSITY	7900	(7850.0, 8239.0)	kg/m^3
MECH_VENTS_INPUT_FLOW	1	(0.9, 1.1)	m^3/s
MECH_VENTS_OUTPUT_FLOW	1.2	(0.9025, 1.375)	m^3/s

Large compartment (1800 m^3) (24 parameters)			
Inputs parameters	Inital value (x_0)	Bounds (min, max)	Units
PRESSURE	101325.0	(101125.0, 101325.0)	Pa
RELATIVE_HUMIDITY	0.5	(0.0, 1.0)	%
MATERIAL_EMMISIVITY	0.9	(0.7, 1.0)	
MATERIAL_CONDUCTIVITY	1.75	(1.0, 3.0)	$kW/(mK)$
MATERIAL_SPECIFIC_HEAT	870E-3	(700E-3, 1121E-3)	$kJ/(kgK)$
MATERIAL_DENSITY	2300	(2000, 2500)	kg/m^3
TARGET_EMMISIVITY	0.9	(0.2, 1.0)	
TARGET_CONDUCTIVITY	15.0	(14.0, 23.0)	$kW/(mK)$
TARGET_SPECIFIC_HEAT	500E-3	(475E-3, 582E-3)	$kJ/(kgK)$
TARGET_DENSITY	7900	(7850.0, 8239.0)	kg/m^3
MECH_VENTS_INPUT_FLOW	1	(0.9, 1.1)	m^3/s
MECH_VENTS_OUTPUT_FLOW	1.2	(0.9025, 1.375)	m^3/s
MECH_VENTS_FIRST_CUTOFF	200	(100.0, 300.0)	Pa
MECH_VENTS_SECOND_CUTOFF	200	(100.0, 300.0)	Pa
FIRE_RADIATIVE_FRACTION	0.35	(0.2, 0.5)	
WALL_VENTS_BOTTOM	1	(0.1, 3.9)	m
OUTPUT_PRESSURE	-100	(-200, -20)	Pa
Zukoski Number	1.0	(0.25, 2.5)	
PVC	0.5	(0, 1)	
PMMA	0.5	(0, 1)	
PE	0.5	(0, 1)	

Small compartment 120 m^3 (24 parameters)			
Inputs parameters	Inital value (x_0)	Bounds (min, max)	Units
PRESSURE	101325.0	(101125.0, 101325.0)	Pa
RELATIVE_HUMIDITY	0.5	(0.0, 1.0)	%
MATERIAL_EMMISIVITY	0.9	(0.7, 1.0)	
MATERIAL_CONDUCTIVITY	1.75	(1.0, 3.0)	$kW/(mK)$
MATERIAL_SPECIFIC_HEAT	870E-3	(700E-3, 1121E-3)	$kJ/(kgK)$
MATERIAL_DENSITY	2300	(2000, 2500)	kg/m^3
TARGET_EMMISIVITY	0.9	(0.2, 1.0)	
TARGET_CONDUCTIVITY	15.0	(14.0, 23.0)	$kW/(mK)$
TARGET_SPECIFIC_HEAT	500E-3	(475E-3, 582E-3)	$kJ/(kgK)$
TARGET_DENSITY	7900	(7850.0, 8239.0)	kg/m^3
MECH_VENTS_INPUT_FLOW	1	(0.9, 1.1)	m^3/s
MECH_VENTS_OUTPUT_FLOW	1.2	(0.9025, 1.375)	m^3/s
MECH_VENTS_FIRST_CUTOFF	200	(100.0, 300.0)	Pa
MECH_VENTS_SECOND_CUTOFF	200	(100.0, 300.0)	Pa
FIRE_RADIATIVE_FRACTION	0.35	(0.2, 0.5)	
WALL_VENTS_BOTTOM	1	(0.1, 3.9)	m
OUTPUT_PRESSURE	-100	(-200, -20)	Pa
Zukoski Number	1.0	(0.25, 2.5)	
PVC	0.5	(0, 1)	
PMMA	0.5	(0, 1)	
PE	0.5	(0, 1)	

Visualisation of the compartment

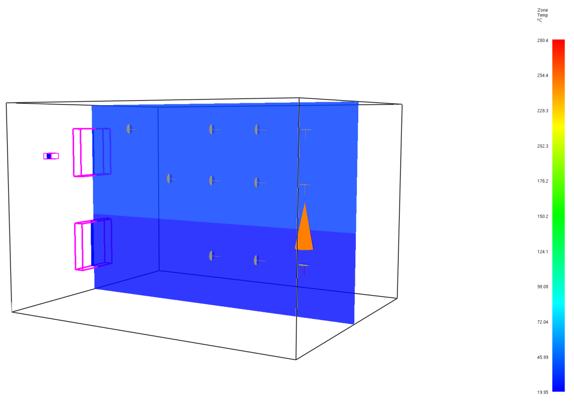


Figure 10: Smokeview visualisation of the small 120 m^3 compartment

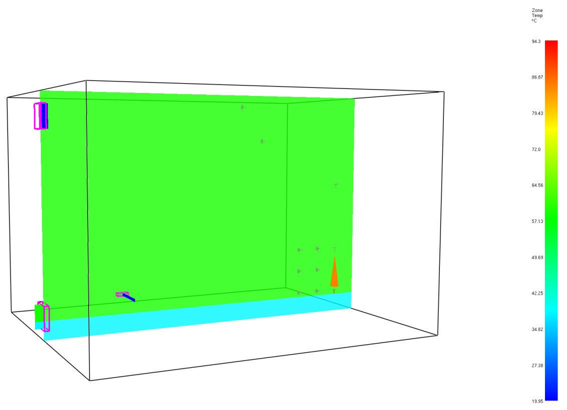


Figure 11: Smokeview visualisation of the 1800 m^3 compartment

Additionnal graphics

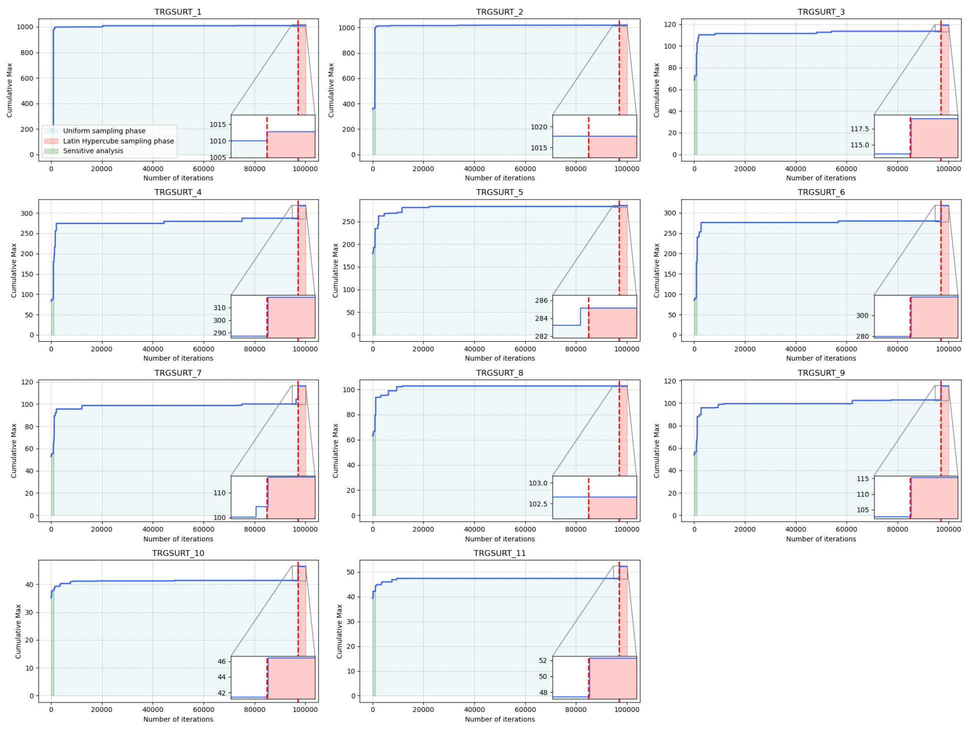


Figure 12: Cumulative maximum for each target (1800 m^3 compartment)

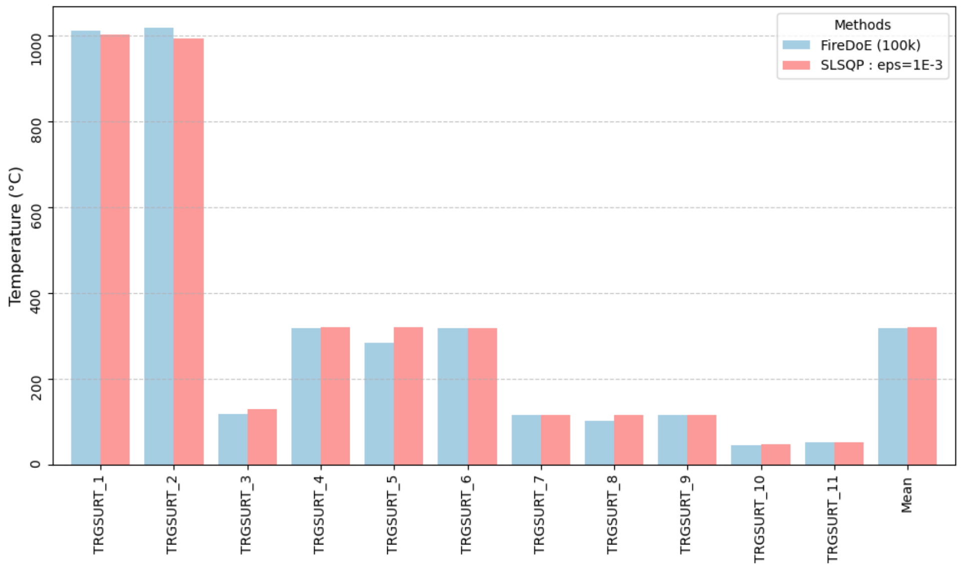


Figure 13: Large compartment (1800 m^3, 24 parameters), Target Surface Temperature Maximum

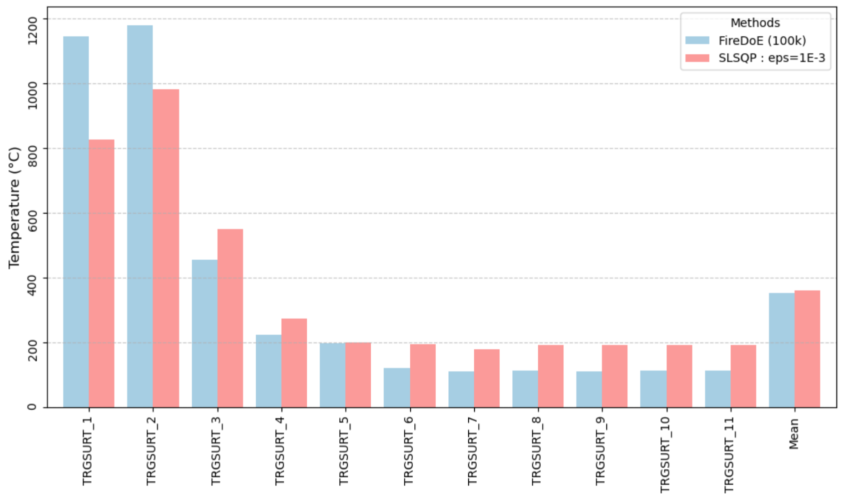


Figure 14: Small compartment (24 parameters), Target Surface Temperature Maximum

Limitations of the study

- Unfortunately FireDoE only works with version 7.1.2 of CFAST while optimization results have been made with version 7.7.3 of CFAST see section 3.3 for more details.
- Due to time constraints, the study was limited to 100,000 runs, a decision that might restrict the thoroughness of the comparison between optimization algorithms and random search. A larger sample could provide a more definitive comparison.

Issue encountered

- In version 7.1.2 of CFAST some arbitraty value have made the model crash [4]. We had to upgrade to version 7.7.3 for this particular reason.
- While upgrading to CFAST version 7.7.3 is that in underventilated cases the computation takes lots of times to finish, a single simulation typically takes between days and weeks. It wasn't the case in 7.1.2 where computation was way faster.
- Initially we also had to use constraints that you specify to the optimizer but that limited our choice of method. Moreover scipy optimize.minimize violate constraints during the optimization, resulting of writing incorrect input file and then crashing CFAST simulation, so we implemented these constraints mannualy inside our obejctive function in order to avoid this issue.
- CData does not allow us to variate lots of important parameters related to the fire. RADIATIVE_FRACTION, CHEMICAL, etc.

References

[1] EPRI/NRC-RES Fire PRA Methodology for Nuclear Power Facilities: Volume 2: Detailed Methodology. Technical Report EPRI TR-1011989 and NUREG/CR-6850, Electric Power Research Institute (EPRI) and U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research (RES), Palo Alto, CA and Rockville, MD, 2005.

[2] Aspen Technology, Inc. *Aspen Plus User Guide.* Aspen Technology, Inc., 2023.

[3] Stefan C. Endres, Carl Sandrock, und Walter W. Focke. A simplicial homology algorithm for Lipschitz optimisation. *Journal of Global Optimization*, 72(2) S. 181–217, March 2018. ISSN 1573-2916. doi: 10.1007/s10898-018-0645-y. URL: `http://dx.doi.org/10.1007/s10898-018-0645-y`.

[4] CFAST GitHub. Issue 669: [Error: at t (=r1) and stepsize h...], 2017. URL: `https://github.com/firemodels/cfast/issues/669`.

[5] Simo Hostikka und Olavi Keski-Rahkonen. Probabilistic simulation of fire scenarios. *Nuclear Engineering and Design*, 224(3) S. 301–311, October 2003. ISSN 0029-5493. doi: 10.1016/s0029-5493(03)00106-7. URL: `http://dx.doi.org/10.1016/S0029-5493(03)00106-7`.

[6] Kevin B McGrattan, Howard R Baum, Ronald G Rehm, Anthony Hamins, Glenn P Forney, Jason E Floyd, Simo Hostikka, und Kuldeep Prasad. *Fire dynamics simulator–Technical reference guide.* National Institute of Standards and Technology, Building and Fire Research . . . , 2000.

[7] M. D. Mckay, R. J. Beckman, und W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, 42(1) S. 55–61, February 2000. ISSN 1537-2723. doi: 10.1080/00401706.2000.10485979. URL: `http://dx.doi.org/10.1080/00401706.2000.10485979`.

[8] J. A. Nelder und R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4) S. 308–313, January 1965. ISSN 1460-2067. doi: 10.1093/comjnl/7.4.308. URL: `http://dx.doi.org/10.1093/comjnl/7.4.308`.

[9] Richard D Peacock. *CFAST:: the consolidated model of fire growth and smoke transport.* 1993. doi: 10.6028/nist.tn.1299. URL: `http://dx.doi.org/10.6028/NIST.tn.1299`.

[10] James G. Quintiere. *Fundamentals of Fire Phenomena.* Wiley, March 2006. ISBN 9780470091159. doi: 10.1002/0470091150. URL: `http://dx.doi.org/10.1002/0470091150`.

[11] A Subbiah und O Laldin. Genetic algorithm based multi-objective optimization of electromagnetic components using COMSOL® and MATLAB®. In *COMSOL Conference Proceedings*, 2016.

[12] Wai Cheong Tam, Eugene Yujun Fu, Richard Peacock, Paul Reneke, Jun Wang, Jiajia Li, und Thomas Cleary. Generating Synthetic Sensor Data to Facilitate Machine Learning Paradigm for Prediction of Building Fire Hazard. *Fire Technology*, 59(6) S. 3027–3048, August 2020. ISSN 1572-8099. doi: 10.1007/s10694-020-01022-9. URL: `http://dx.doi.org/10.1007/s10694-020-01022-9`.

[13] Clarence Worrell, Louis Luangkesorn, Joel Haight, und Thomas Congedo. Machine learning of fire hazard model simulations for use in probabilistic safety assessments at nuclear power plants. *Reliability Engineering & System Safety*, 183 S. 128–142, March 2019. ISSN 0951-8320. doi: 10.1016/j.ress.2018.11.014. URL: `http://dx.doi.org/10.1016/j.ress.2018.11.014`.