ADDRESSING CRITICAL DEPENDENCIES IN THE PROBABILISTIC PERFORMANCE ASSESSMENTS OF MULTI-PURPOSE SYSTEMS WITH PyCATSHOO

Hassane CHRAIBI[a*], Dominique VASSEUR[a], Tu Duong LE DUY[a] and Mickaël HASSANALY[a]

[a] EDF, Paris Saclay Lab - PERICLES - Palaiseau, France

Abstract: EDF has recently developed PyCATSHOO, a tool aimed to the assessment of hybrid complex systems. Based on Piecewise Deterministic Markov Processes. This tool offers the ability to take into account deterministic physical phenomena in the probabilistic performance assessments of complex systems. It thus allows a realistic consideration of the dynamics of the modelled systems. In particular it makes possible accurate modelling of critical dependencies often present in multi-objective infrastructures such as electrical systems. After having identified the most frequent among these dependencies, we propose, in this article, to show the ability of PyCATSHOO to address them. In this objective a simplified test-case has been used and a set of operating scenarios has been developed to highlight situations where the dependencies to be studied arise. The simulation of these operating scenarios shows that their modelling with PyCATSHOO is faithful to reality. It also shows the PyCATSHOO ability to account for complex dynamics and thus, shows its ability to perform assessments without resorting to simplifying assumptions. An overall assessment of system performances has then been made. It shows that a realistic consideration of critical dependencies within multi-objective systems implies that the results of probabilistic evaluations of their performances are sometimes counterintuitive.

Keywords: Critical dependencies - dynamic probabilistic performance assessment - PyCATSHOO

1. INTRODUCTION

Most of critical infrastructures worldwide are interconnected and can be seen as a global multi-purpose systems with interdependent subsystems [1]. This interdependency should be taken into account when assessing the compliance of these installations to their different requirements.

An integrated probabilistic performance assessment (PPA) of multi-purpose systems needs to address several kinds of critical dependencies between the system objectives. These dependencies which have been identified and classified may come from various sources:

- Different purposes met by the same system are therefore subject to the same environmental constraints.
- Components may be “shared” by several purposes. These common components may be of three types:
  - Identical components used by all the system purposes but each is dedicated to only one of them.
  - Components shared by all the system purposes.
  - Components used for particular system purpose and, thanks to cross connecting devices, can be useable by the other ones.
- Common or inter-connecting components may exist.
- Resources may be common in terms of operating and maintenance teams.

But the shared resources, when it comes to repair or to achieve a backup action, the shortage of time required by these actions, the cascade effect that may be initiated by external events or intrinsic failures, as well as the increasing risk of the operator action failure due the critical context, can be

* hassane.chraibi@edf.fr

addressed by boolean approaches only with conservative assumption. Moreover, such methods only address dependencies in simplified way in order to make their modelling and quantification possible.

Some studies already made the switch to dynamic approaches. Thus, in [2], the authors introduced a methodology which combines Monte Carlo simulation and continuous time Markov Processes to model mutual dependencies. Such studies represent definitely a significant progress toward more realism. But due to the complexity of the systems and phenomena studied, these approaches are still based on the "scenario vision" of the conventional boolean approach, and consequently can inherit its limits in terms of exploration of possible states. The idea here is rather to rely on a modelling faithful to the vision of systems as sets of components interacting together and with the external environment. This is the purpose of a new method recently developed by EDF R&D division for hydraulic domain.

This method is based on the distributed hybrid stochastic automata and is implemented in the PyCATSHOO tool [3][4]. PyCATSHOO offers the modelling and quantification mechanisms that help dealing with several kinds of dependencies including the two-way interactions inside a system between discrete stochastic events and deterministic continuous physical phenomena.

The objective of this paper is to highlight some of these mechanisms and to illustrate their possible use in a dependability assessment of a simplified test-case which represents a two-area electricity supply system with PyCATSHOO tool.

In section 2, different types of critical dependencies involving components and human actions are discussed. Then, the modelling of these dependencies in a dynamic manner is illustrated with PyCATSHOO tool. Section 3 evaluates the probability of the loss of electricity supply of two consumption areas by Monte Carlo simulation implemented in this tool. Some conclusions and perspectives are finally given in section 4.

2. DEPENDENCY MODELLING WITH PYCATSHOO

We have chosen to illustrate the use of the dependency mechanisms provided by PyCATSHOO by dealing with **six types of critical dependencies** involving components or human actions.

Type 1: A device started according to the physical state of the system.

Type 2: A device usage shared between multiple areas and assigned to one of them according to a priority criteria which evolves over time.

Type 3: A human action shared between the same kind of activities in two different areas with different priorities which evolve over time.

Type 4: A human action shared between two kinds of activities with different but predefined priorities.

Type 5: An action which duration depends on a severity measure of the operating context.

Type 6: An action which failure probability depends on a severity measure of the operating context.

First of all we will be describing the study case and showing where these dependencies intervene.

2.1. An Electricity Supply System (ESS) for two consumption areas

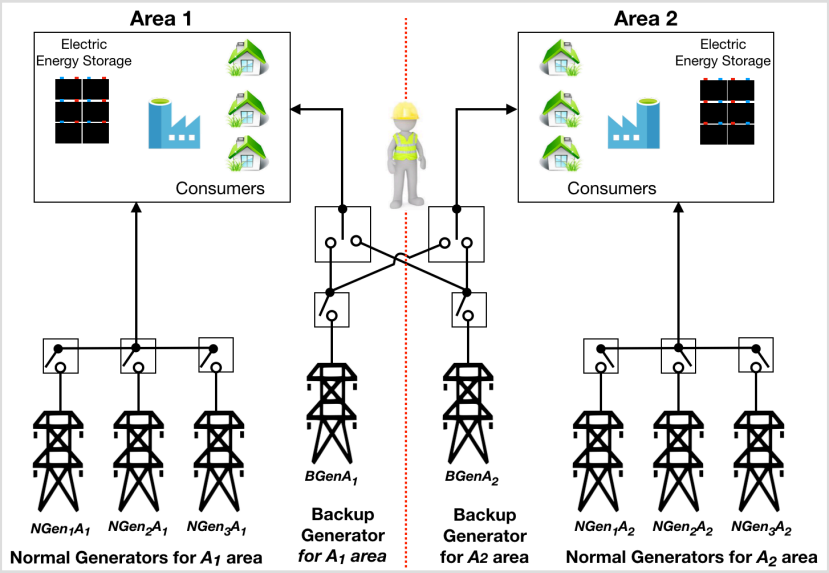
As illustrated in figure 1, this system comprises two electricity consumption areas supplied by a transmission subsystem. In addition to domestic and industrial consumers, a consumption area comprises an electricity storage device. The latter compensates for the lack of supply and stores the supply surplus within the limits of its capacity. Consumption in an area, in the nominal operating mode, is ensured by the components $NGenA_{(n:\ 1,2,3)}A_{(x:\ 1\ or\ 2)}$†. In addition to operating in passive

† Ex. : $NGen_3A_2$ designates the **N**ormal **Gen**erator number **3** in the area number **2**

redundancy, we assume that $NGenA_{(n:\ 1,2,3)}A_x$ start respectively when the electric load exceeds respectively 0 MJ/h, 50MJ/h and 150 MJ/h. These assumptions introduce the **type 1** dependency.

We also assume that both areas may experience power leak, or unexpected consumption, according to exponential probability distribution. Such a leak may also be "repaired" also according to exponential probability distribution. Finally we have introduced a common cause failure for $\mathbf{NGen_1A_1}$ and $NGen_1A_2$.

Figure 1: An electricity Supply System- A model in two consumption areas' context



If the power balance in an area A_1, respectively A_2, becomes negative, the backup component $BGenA_1$‡, respectively $BGenA_2$, has to be started by an operator. If $BGenA_1$ fails to operate, the $BGenA_2$ will be solicited. $BGenA_2$ is indeed considered as a backup for $BGenA_1$ and vice-versa, except when booth areas need a $BGenA_x$ while only one is available. In this case the priority goes to the area with the highest electric load. This introduces a **type 2** dependency.

We assume that there is only one operator who has two missions. The first one consists in repairing failed components. Several components may need repair at the same instants either in the same area or in different areas. In the latter case, the operator must favour the component which belongs to the area with the highest electric load. This introduces the **type 3** dependency.

The second operator mission consists in starting a $BGenA_x$ (alignment) when required. This second mission is in general needed when some other components fail to operate. This means that repair and alignment actions may compete. To solve this conflict, we adopt the rule where operating instructions favour the alignment action. This introduces the **type 4** dependency.

In our model, the alignment is the ultimate action that can be done to avoid loss of supply i.e. negative balance. This action starts in general when all the other actions have failed and when the accumulated loss of electricity supply becomes high. This action has therefore to be accomplished in a stressful context. Such a context may increase the duration of the alignment action as well as its failure probability. This observation introduces the **type 5** and **6** dependencies.

‡ $BGenA_1$ designates the **B**ackup **Gen**erator in the area number **1**

2.2. PyCATSHOO SFPCS modeling elements

We will not describe here all the modelling concepts of PyCATSHOO. For such details, the reader can refer to the papers [3] & [4] and check out pycatshoo.org website. However, we will just remind the key concepts.

The first concept, which is common to almost all modelling languages are state variables. We have to inventory all of those which characterise the system, either continuous or discrete ones. As PyCATSHOO is based on the framework of the Piecewise Deterministic Markov Process (PDMP), the first modelling means consists in declaring a set of automata in order to model the dynamic behaviour of every system component. The second modelling means consists, when needed, in declaring the flow of the PDMP which is a set of explicit or differential equations that govern the continuous state variables. This must obviously be done in the perimeter of components in order to minimise the complexity that the modeller has to face. PyCATSHOO will then take over to piece back automatically the pieces together.

The last important task will ensure the communication between the components. It consists in creating communication channels between a component and the rest of the system. These channels are structured inside the so called message boxes.

In the following section we will not go into the exercise of the variable definition even if such a task may require particular attention in order to distinguish, for instance, discrete variables of a component from its states. We will not describe the message box either. We will mainly focus on the dynamic behaviour modelling by giving some excerpts of automata definition and equations declarations.

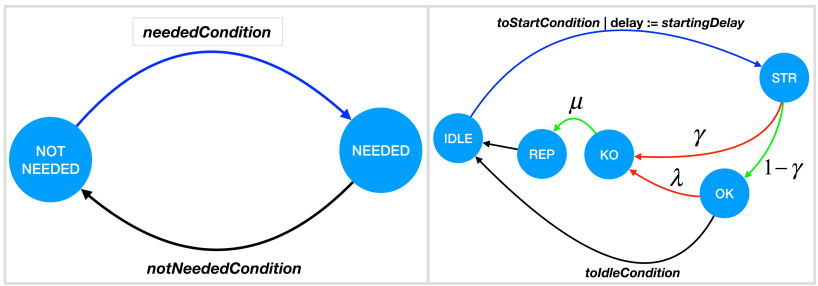
2.2.1 Active components modeling

Active components are those behind $\boldsymbol{NGen_nA_x}$ and $\boldsymbol{BGenA_x}$. They all have a dynamic behaviour that can be modelled by the automaton given in figure 2.

An active component is initially not needed (**NOTNEEDED**). It stays in this state until a particular condition '***neededCondition'*** becomes true. It then enters the state **NEEDED** and returns to **NOTNEEDED** state when another particular condition '***notNeededConditon'*** becomes true.

An active component is also initially in an **IDLE** state. When a particular condition is satisfied '***toStartCondition'***, the component enters a starting state **STR**. This entering may take a duration equal to the parameter ***startingDelay***. This parameter is in general equal to 0 except for $\boldsymbol{BGenA_x}$ components where this transition corresponds to the alignment operation and requires an operator intervention. The starting action may fail with a probability γ which makes the component enter in the state **KO**, otherwise, with the probability $1-\gamma$, it enters the state **OK** and starts functioning. When it is in **OK** state the component may fail with a failure rate λ and then goes to **KO** state. It also may be no longer required if the condition '***toIdleCondition'*** becomes true. It then returns to **IDLE** state. The component may obviously be repaired when it is in state **KO** with a repair rate μ. It then goes in a furtive state **REP** and finishes in **IDLE** state.

Figure 2: Generic Behaviour of an active component



The conditions mentioned above are to be declined according to the nature of the component. Hence, for a $NGen_nA_x$ component the four conditions may be written as follows :

neededCondition :=
electric load > electric load threshold or is a backup for a faulty component.

notNeededCondition :=
electric load <= electric load threshold and not a backup for a faulty composent

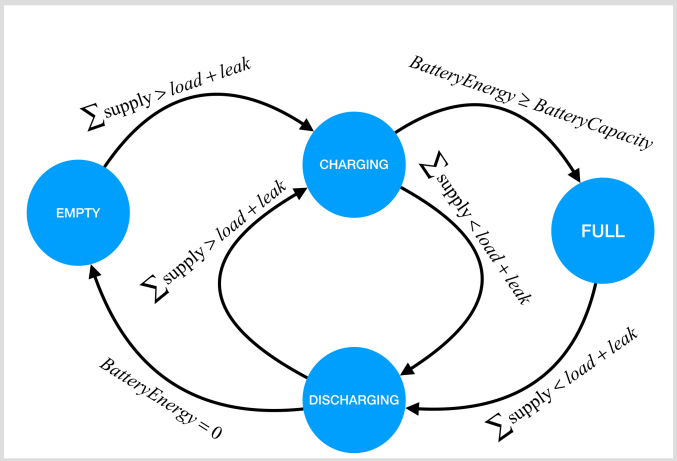
toStartCondition :=
in **NEEDED** state

toIdleCondition:=
in **NOTNEEDED** state

2.2.2 Consumption Area modelling

As shown above, the states of the consumption area components determine the behaviour of all the system. Indeed, the power balance determines if $BGenA_x$ components are needed or not, and the electric load determines when the $NGen_nA_x$ components are started. The electric load also determines which area has the priority when repair or alignment are required by both areas.

Figure 3: A Consumption area automaton : Thresholds' crossings



Several automata have then been created to precisely characterise the different states of an area in order to give an accurate instantaneous picture of this component to the system components whose behaviour depends on its states. The automaton of figure 3 is one of them. It marks the thresholds' crossing whose detection is required to start battery charging or the use of the latter to compensate for the lack of electric supply.

In addition to its automata, the area is characterised by the amount of available energy in the storage device (batteries). Hence we have to declare the equations which account for the evolution over time of this energy. These equations constitute the flow of the PDMP. They are shaped according to the current state of the system as follows:

If the battery is charging :

If current power balance in the area is lesser than battery charging power limit $batteryPowerChargingLimit$ then

$$\frac{dBatteryEnergy}{dt} = balance \quad (1)$$

Else

$$\frac{dBatteryEnergy}{dt} = batteryPowerChargingLimit \quad (2)$$

Else if the battery is discharging

$$\frac{dBatteryEnergy}{dt} = -batteryPower \quad (3)$$

Else

$$\frac{dBatteryEnergy}{dt} = 0 \quad (4)$$

2.2.3 Modelling of a dispatcher component

A component called ***dispatcher*** is introduced in this modelling in order to implement prioritisation rules mentioned above for actions and for components in case of uses conflicts.

The ***dispatcher*** is mainly used to solve possible conflict when two different operator actions are required: two repairs, a repair and an alignment or two alignments.

The ***dispatcher*** is connected to all the components potentially concerned by these actions and to all the operators potentially available for the two units - only one has been considered in our test case-.

When a component needs an operator, the connected ***dispatcher*** is informed through the appropriate message box. It can then know that there is a need for an operator and it can know the nature of this need: repair or alignment. Then, the ***dispatcher*** seeks for a free operator by scanning the state of all the connected operators. If one is available it is immediately booked in order to make it unavailable for other requests. The ***dispatcher*** seeks, among the components waiting for alignement, the one with the highest criteria. Here, the criteria is simply the current value of the power load inside the area. If such a component is found, the ***dispatcher*** marks it with the ***id*** of the operator. The starting condition of this component has to verify if it is marked with the ***id*** of an operator. If yes, such an operator is finally booked for the component and the starting condition of the component becomes true. The component which is, in our case, a $\boldsymbol{BGenA_x}$ component, starts and this starting lasts a duration equal to the value of the parameter ***startingDelay***. During all the starting operation -alignement-, the operator stays in busy state. At the end of the starting operation - alignment- the operator is freed.

If there is no need to an other alignment, a new search cycle is achieved by the ***dispatcher*** but this time for components waiting for repair.

2.3 Dependency scenarios

In this section we will play several scenarios in order to show that the PyCATHSOO modelling mechanisms are actually compliant to the specification we have mentioned above and capture faithfully the dependencies between the two modelled cinsumption areas.

2.3.1 Which component to repair first ?

In this scenario we provoke a simultaneous failure of $NGen_1A_1$ and $NGen_1A_2$ at an instant ***failureTime*** and we run this scenario twice. The first time with ***failureTime*** = 400 h i.e. when the power load inside the area A_1 is higher and, a second time with ***failureTime*** = 4600 h, when the power load inside the area A_2 is higher.

Figure 4 gives the evolution over time of power loads inside A_1 and A_2

Figure 4: Power Loads inside A_1 & A_2

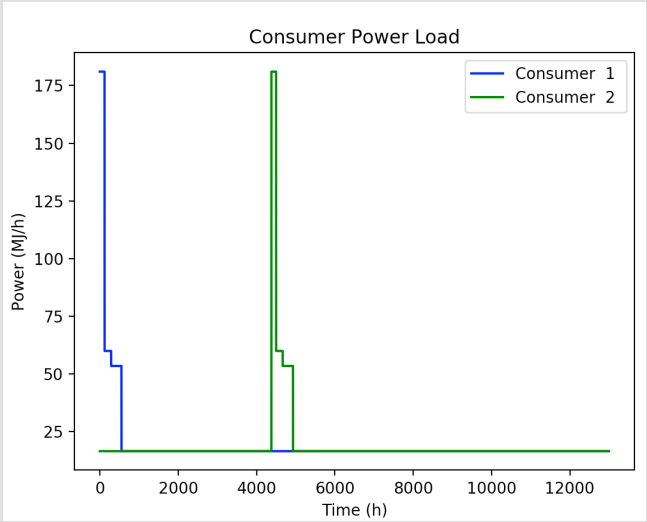


Figure 5 shows the case where the failures of $NGen_1A_1$ and $NGen_1A_2$ occur at 400h which means, according to figure 4, when the power load inside A_1 is higher. We can see that at 400h the power delivered by $NGen_1A_1$ and $NGen_1A_2$ falls to zero. $NGen_1A_1$ is then restarted before $NGen_1A_2$. This means that $NGen_1A_1$ is repaired before $NGen_1A_2$.

Similarly, figure 6 shows the case where the failures occur at 4600h when the power load of A_2 area is higher. In this case, the A_2 area components have the priority for repair over A_1 area components.

Figure 5: Failures and repairs of $NGen_1A_1$ & $NGen_1A_2$ when A_1 has priority

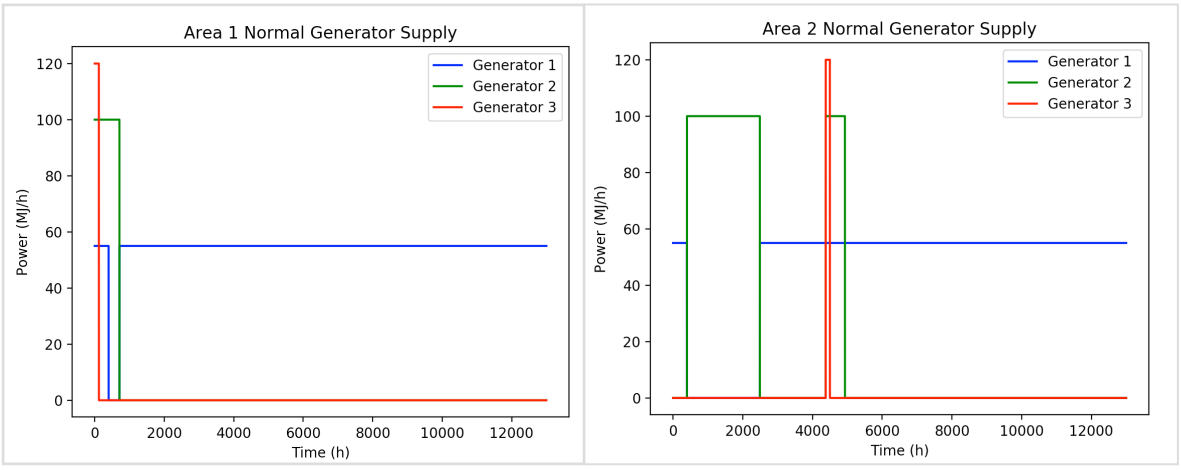
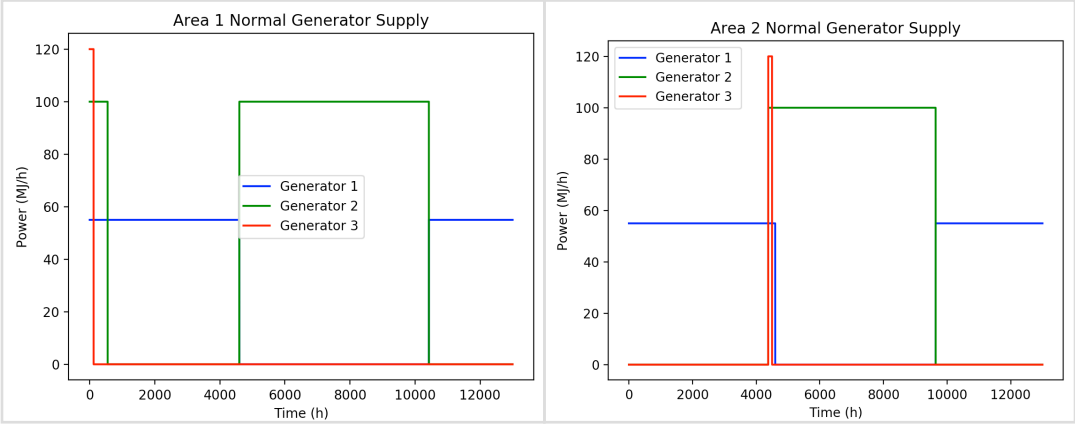


Figure 6: Failures and repairs of $NGen_1A_1$ & $NGen_1A_2$ when A_2 has priority

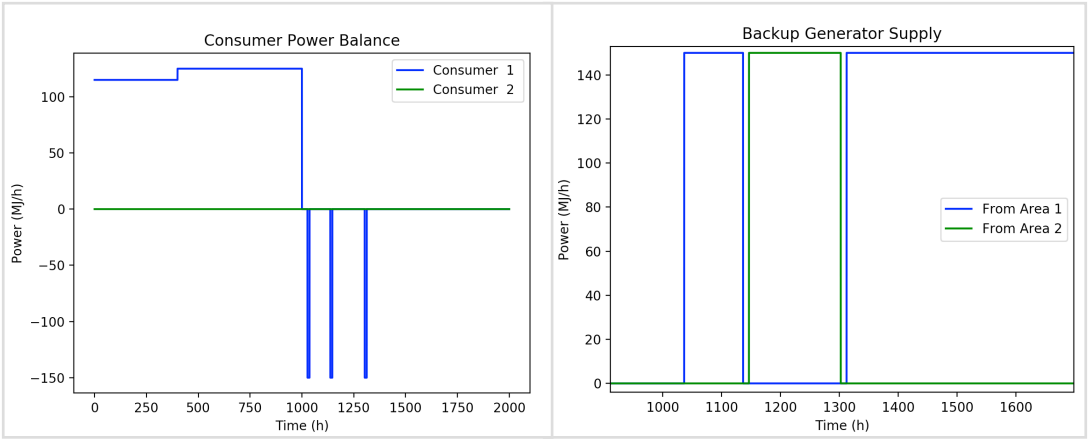


2.3.2 Which action to accomplish first, repair or alignment ?

In this scenario we assume that the power load is null for the second area and that all $NGen_nA_1$ components fail at the instant 1000h without the ability to be repaired. This means that consumption area A_1 will use the battery and then its balance will become and stay negative until the starting of $BGenA_1$.

We also assume that the $BGenA_1$ fails after 100h of operation. At this instant, the choice opened to the operator is about which action to accomplish : repair the broken $BGenA_1$ or align $BGenA_2$ in order to supply the first area. As we mentioned above we obviously assume that the operating rules require that the alignment has the highest priority. This is what we can check on figure 7.

Figure 7: Alignment of the backup rather than repair

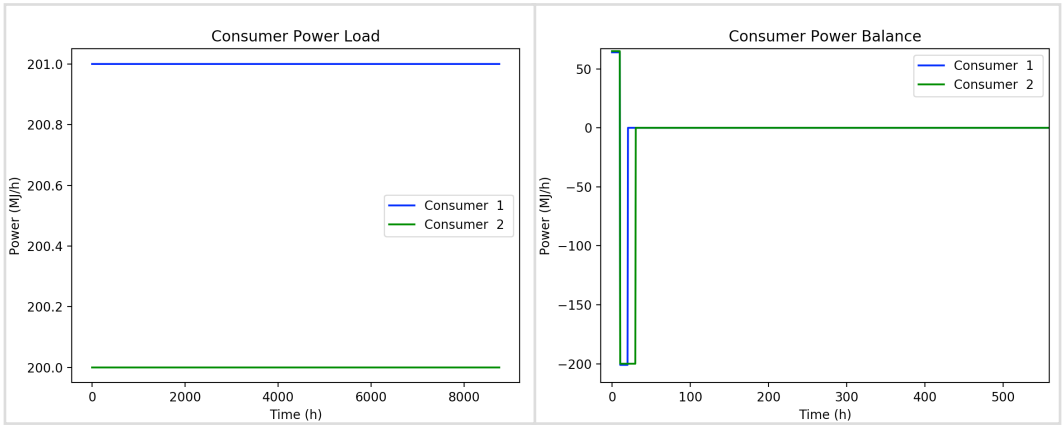


The left side of figure 7 shows the evolution over time of consumption area balances. The balance of consumption area A_2 remains null as we assumed that there is no power load. The balance of consumption area A_1 begins by charging the battery. When the latter is full, the balance jumps to a higher plateau until 1000 h when all the $NGen_xA_1$ are supposed to fail. The balance becomes and remains null as the battery compensates the lack of the electric supply. After that, the balance becomes negative and $BGenA_1$ is started as shown in the right side of figure 7. As mentioned above $BGenA_1$ fails after 100h of operation and we can see that 10h after that i.e. after the time that the operator takes to align $BGenA_2$, the latter is started. Later, $BGenA_2$ is stopped and $BGenA_1$ is started. This means that the operator proceeded to alignment of $BGenA_2$ before starting to repair $BGenA_1$ as required by operating rules.

2.3.3 Which consumption area must be supplied first ?

In this scenario we assume that all the $NGen_nA_x$ of both consumption areas fail at 10h without the ability to be repaired. We also assume that no one of $BGenA_x$ can fail and that the electric load in consumption area A_1 is higher than in A_2. In this context, as there is only one operator, the latter has to choose to which area the priority must go. The answer comes from the operating rules. This means that the area with the highest power load has to be supplied first. This is what we can see on figure 8. We can see in this figure that the balance in the area A_1 begins to increase first. The balance in the area A_2 begins increasing 10 hours later. The lag of 10 hours is due to the time required by the operator to achieve the alignment action.

Figure 8: Alignements' priority



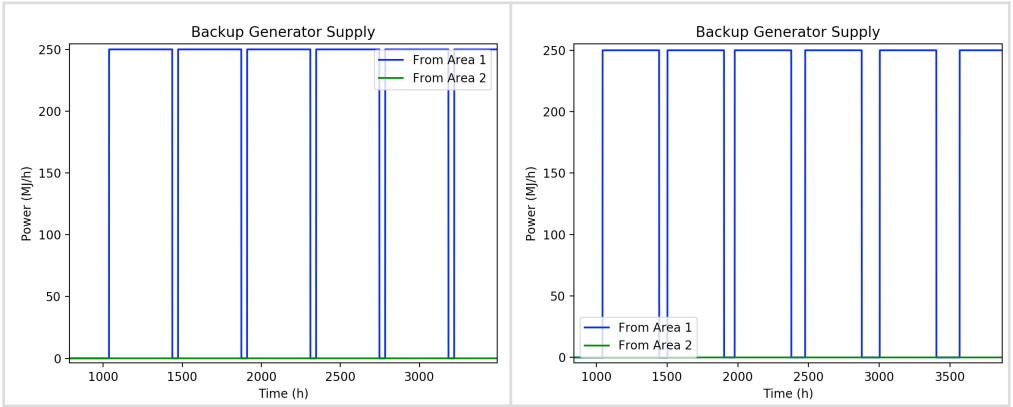
2.3.4 The alignment duration increases with the severity of the context

The chosen indicator to monitor the severity of the context for human operators is the accumulated lack of electric supply. We assume here that this severity make operators taking more time to align a $BGenA_x$. We have proposed a formula which bind the alignment delay to the accumulated lack of electric supply. This formula, obviously, exaggerates the dependency between these two variables in order to make the effect of this dependency clearly visible.

$$alignmentDelay = 10 \times (10^{-4} - (10^{-4} - 1) \times e^{-delayEvolutionRate \times accumulatedElectricLack}) \quad (5)$$

To highlight this kind of dependency we have adopted a scenario where all the $NGen_nA_x$ components fail without possible repair. We also assume in this scenario that only the A_1 area has positive power load.

Figure 9: Evolution over time of alignment delay



The left part of figure 9 shows the operation of the $\boldsymbol{BGenA_1}$ when the parameter $delayEvolutionRate$ is null i.e. when the alignment duration doesn't depend on the accumulated electric lack. In this case we can see that the lag between the stopping and the starting of the $\boldsymbol{BGenA_1}$ is constant. As for the right part of figure 9, the parameter $delayEvolutionRate$ is not null. We can see in this case that the model accounts for the dependency between the amount of accumulated electric lack and the alignment duration. The latter increases clearly over time.

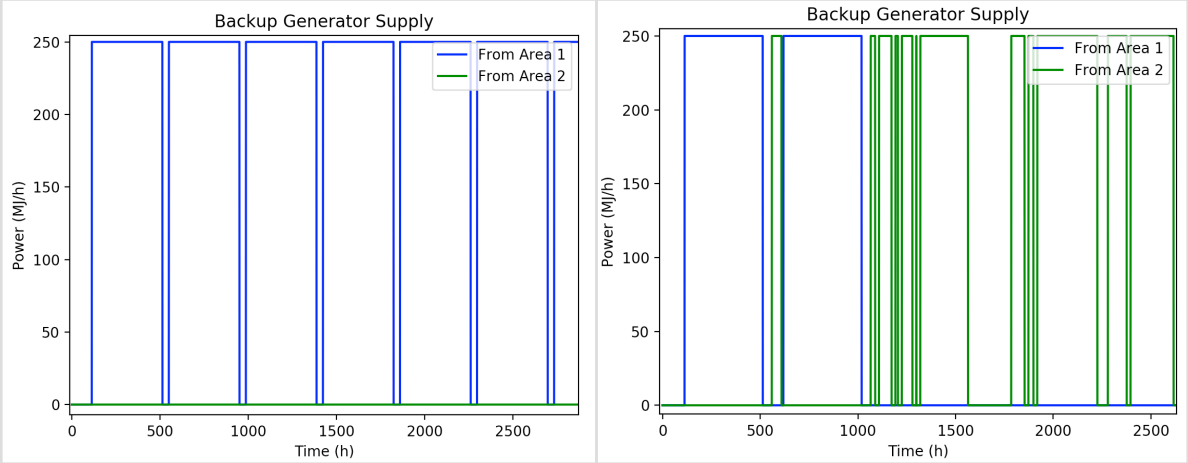
2.3.5 The probability of alignment failure increases with the severity of the context

Here, we have adopted the same scenario as in the previous section except that we bind the failure probability of the alignment action to the severity of the context i.e. to the accumulated electric lack.

$$gamma = gamma_0 + (1 - gamma_0) \times e^{-gammaEvolutionRate \times accumulatedElectricLack} \quad (6)$$

Figure 10 gives the evolution over time of the power supplied by $\boldsymbol{BGenA_1}$ and $\boldsymbol{BGenA_2}$. The left part corresponds to the case where $gammaEvolutionRate$ is null which means that there is no dependency between $gamma$ and the accumulated electric lack. In this case the starting of $\boldsymbol{BGenA_1}$ succeeds as its $gamma$ is still low and constant. Conversely, in the right part of figure 10, the $gammaEvolutionRate$ is not null, which means that probability of the starting failure of $\boldsymbol{BGenA_1}$ increases over time. In this case we can see that after the second succeeded starting of $\boldsymbol{BGenA_1}$ the latter fails to start due to the increasing of its $gamma$ and the $\boldsymbol{BGenA_2}$ is systematically started as a backup.

Figure 10: Evolution of failure probability of alignment operation

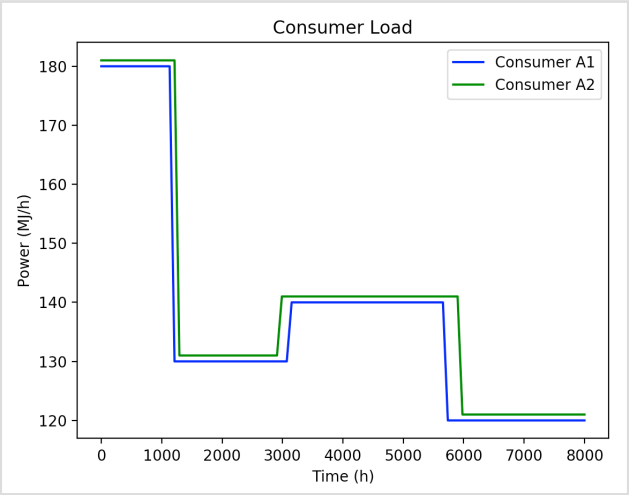


3. A PROBABILISTIC PERFORMANCE ASSESSMENT OF TWIN AREAS ELECTRIC SUPPLY SYSTEM

In this section we simulate 8000 hours of a twin areas electric supply system in order to evaluate the probability of experiencing a negative power balance in one of the two areas or both of them. We will address three situations. In the first one we assume that there is a positive power load in both areas. Figure 11 gives the evolution over time of these power loads.

In the two other situations we assume that only one area has a positive power demand with the same profile as that given in figure 11.

Figure 11: Evolution over time of power loads



3.1 Situation 1 - PPA in actual twin-area context

Table 1 gives the probability of the occurrence of an electric supply lack i.e. negative balance in two-area context as well as its 99% confidence interval.

Unlike what we might have expected at first sight, we can observe that the order between the probability values of the undesired events in the two areas is the opposite of the order between the power loads. In this case, the intrinsic risks in the two areas are close but, as the power load is always higher in A_2 area, all the repairs and alignments resources will be primarily assigned in favour of the latter which then presents a lesser risk of power supply loss.

Obviously, this observation cannot be generalised. For other data set the results may be different, in particular when the power loads in the two areas are significantly different. But only system modelling and simulation can say it and this explains the importance of the approach offered by PyCATSHOO.

Table 1: Probability and confidence interval of the undesired events in twin-area context

	Undesired Event in A_1 area	**Undesired Event in A_2 area**
Probability of undesired event	8,02E-2	6,55E-2
Confidence interval of 99%	6,99E-4	6,37E-4

3.2 Situation 2 - PPA in one area context

As mentioned above, in this section we repeat the assessment for both areas taken separately. This means that we reduce to zero the power load in the other area and we inhibit the use of its ***BGen*** component as a backup for the ***BGen*** of the considered area.

Table 2 gives the probability of the occurrence of a situation where the balance becomes negative in the area with the positive power load as well as its 99% confidence interval.

We can see that, when taken separately, the probability of undesired events is lesser than it is when the system simultaneously supplies both areas. But this difference is more pronounced for A_1 area than it

is for A_2 area. Indeed, in the context of twin-area, A_2 has the priority which means that in case where both areas simultaneously need repair or alignment, A_2 is favoured. This made the risk of power lack in A_2 lesser at the expense of A_1 which does not take full advantage of its own resources.

Table 2: Probability and confidence interval of the undesired events in one-area context

	Undesired Event in A_1 area	**Undesired Event in A_2 area**
Probability of undesired event	6.13E-2	6,48E-2
Confidence interval of 99%	6,18E-4	6,30E-4

4. CONCLUSION

This paper illustrated the possible use of PyCATSHOO tool to deal with different types of critical dependencies involving components and human actions of multi-purpose systems. This tool based on the framework of the Piecewise Deterministic Markov Process offers a paradigm of distributed stochastic hybrid automata. The modelling of these dependencies and the probabilistic performance assessment of a simplified test-case have therefore been carried out in a dynamic manner.

The proposed method relies on a modelling faithful to the vision of the systems as sets of components interacting together and with the external environment to achieve a realistic assessment.

The study case shows that the probabilistic assessment results depend strongly on the operating rules (technical and organisational constraints) especially in degraded situation and multi-purpose context where the dependencies are exacerbated. More importantly, these results show that a such assessment in multi-purpose context cannot be a simple extrapolation of the results in a one purpose context, but rather a multi-purpose context requires the use of a model integrating a faithful image of subsystems interdependencies.

In this paper, we focused on the risk related to the loss of an electric supply system of two different consumption areas by considering a simplified system in order to provide with a good understanding of the PyCATSHOO ability to address critical dependencies in a multi-purpose system performance assessments. Actual systems i.e. more complex systems require more complex modelling. PyCATSHOO provides means to deal with such problems. Indeed up to now other large and complex systems in hydropower-generation domain were successfully addressed at EDF by this tool. Nevertheless, significant modelling effort is still needed for such systems.

References

[1] Xing Liu, Ionela Prodan, Enrico Zio. *RESILIENCE ANALYSIS OF INTERCONNECTED SYSTEMS BY A SET-THEORETIC APPROACH. Congrès Lambda Mu 19, Oct 2014, Dijon, France* ⟨*hal-01108225*⟩

[2] Jang and al. *"Dynamic approach on multiunit probabilistic risk assessment using continuous Markov and Monte Carlo method"*. In Proceedings of the ANS international topical meeting on probabilistic safety analysis, Pittsburgh, PA, (2017)

[3] Hassane CHRAIBI, Jean Christophe HOUDEBINE, Alain SIBLER. PyCATSHOO: "*Toward a new platform dedicated to dynamic reliability assessments of hybrid systems*". PSAM 2016

[4] Hassane CHRAIBI. "*Dynamic reliability modelling and assessment with PyCATSHOO: Application to a test case"*. PSAM 2013