DOUBRE Maxime
ZHUO Maxime

Compte rendu multi-agents

Besoins initiaux

Nous avons comme environnement une salle de cours qui contient des lumières, des volets et des capteurs.
Notre système est composé d'agents Lumières et Volets afin de gérer la luminosité dans la salle, donc dans notre environnement.

Les besoins sont de gérer les niveaux des lumières et des volets afin qu'on puisse gérer la luminosité de la salle tout en minimisant la consommation des lumières.

Les mots-clés sont les suivants :
- consommation
- gestion
- luminosité
- niveau

Dans notre système nous avons une contrainte de seuil de luminosité. Ce seuil est rentré par l'utilisateur au début du main et ne peut plus être modifié pendant la simulation.

Caractérisation de l'environnement

Entités passives	Entités actives
Capteurs	Lumières
	Volets

Nos capteurs sont une entité passive et ne sont pas une classe en elle-même. Ce sont une simple méthode dans notre classe environnement *'Classe_connectee(Environnement)'* qui s'appelle *'getLumCapte()'*.

Nos lumières et volets sont des classes et sont des entités actives car elles agissent par elles-mêmes. Elles ont des méthodes *'on_perceive'*, *'on_decide'* et *'on_act'* qui leur permettent de changer leur niveau selon la luminosité captée dans la salle.

Schéma des flots de données sur un cycle :

amas	environnement	Lumière	Volet
<=	getLumCaptee()		
<=	==	getNiveau()	
<=	==	==	getNiveau()
setPrio()	==	=>	=>
		on_perceive()	on_perceive()
	getLumCaptee()	=>	=>
		on_decide()	on_decide()
		on_act()	on_act()
<=	getLumCaptee()		
<=	==	getNiveau()	
<=	==	==	getNiveau()
maj_conso()			

Au début l'amas a une méthode *'on_cycle_begin'* dans laquelle il demande à l'environnement la luminosité captée et le niveau des volets et des lumières.
A partir de ces niveaux il décide de changer les priorités des volets et des lumières afin que si on veut augmenter la luminosité et que les volets ne sont pas totalement ouverts on les ouvre en priorité avant d'allumer les lumières.

Après ça on a les méthodes des volets et des lumières *'on_perceive'*, *'on_decide'* et *'on_act'* qui changent les niveaux des lumières et des volets

Pour finir on a la méthode *'on_cycle_end'* de l'amas qui demande la lumière captée par l'environnement et les niveaux des lumières et des volets et qui met à jour les consommations.

Notre environnement est :

- non accessible : info sur l'état de l'environnement incomplètes.
- non déterministe : si un volet s'ouvre il peut induire une augmentation de la luminosité mais aussi un changement de niveau d'une lumière dans le cas ou il a permis une trop grande entrée de lumière extérieur.
- dynamique : l'environnement change d'état lorsque l'heure est changée. Il ne reste donc pas inchangé tant que l'on a pas d'actions des agents.

- continue : nous n'avons aucune idée à l'avance de quand notre simulation sera finie ou de combien de changements nous effectuerons.

Identification des agents

Nous n'avons que des entités actives :
- Volets
- Lumières

	Lumières	**Volets**	**Capteur**
Autonome ?	Oui	Oui	Non : pas de décisions
Poursuit un but local ?	Contrôler la luminosité intérieur qu'elles génèrent	Contrôler la luminosité venue de l'extérieur	Capter la luminosité
Interaction avec d'autres entités ?	Capteurs	Capteurs	Volets, Lumières, Amas
Vue partielle ?	Oui (n'a pas les paramètres du volets, n'a que la valeur captée de luminosité)	Oui (n'a pas les paramètres des lumières)	Oui
Négociation ?	Non	Non	Non

Dans un vrai système multi-agent on ne pourrait pas décider des priorités dans l'amas donc nous devrions intégrer ces méthodes dans nos agents et il y aurait des intéractions et des négociations entre eux afin de gérer ces priorités
Dans notre simulation nous avons géré cela dans notre amas ce qui fait qu'il n'y a pas d'intéractions entre nos agents.

	Lumières	**Volets**	**Capteur**
agit dans environnement dynamique ?	Oui	Oui	Oui
confrontation à des échecs ?	Non	Non	Non
Situations non coopératives ?	Non	Non	Non

Concevoir les agents

Nos deux types d'agents sont les Lumières et les volets. Ils ont tous les deux la même architecture et se gèrent de la même façon car on raisonne dans notre simulation avec un niveau qui correspond à leur pourcentage. Pour le volet 30% veut dire qu'il est ouvert à 30% et pour les lumières 30% veut dire que la lumière est allumée à 30% de sa puissance.

Compétences (connaissances sur un domaine) :

Nos agents ont des connaissances sur le seuil de luminosité voulu dans la salle, la luminosité actuelle de la salle et leur niveau.
On retrouve donc dans les classes agents les attributs :

- niveau
- valeur_captee (la valeur donnée par le capteur)
- seuil

Aptitudes (faculté à raisonner) :

Nos agents raisonnent à partir de leur connaissance afin de savoir s'ils doivent augmenter leur niveau ou le diminuer. Ce raisonnement fait appel à leur niveau, la luminosité captée et au seuil afin de savoir ce qu'il doit faire.
Selon le raisonnement, il change l'attribut *'chaine'* qui contient l'agissement à venir de l'agent : "augementer", "diminuer" ou "".

Langage d'interaction :

Nos agents agissent directement sur leur niveau suite à la décision prise grâce à leur niveau, la valeur captée et le seuil.
Ils ont cependant des appels à des fonctions de l'environnement qui permettent d'obtenir la valeur captée de la luminosité ou encore leur priorité.

Représentation du monde :

Les agents ont en attribut leur environnement et ont une représentation partielle de celui-ci grâce à leurs attributs *'valeur_captee'* et *'seuil'*.

Situations coopératives :

Dans notre simulation nous n'avons pas de situations non coopératives, notamment grâce aux priorités.

Cycle de vie d'un agent :

- **on_perceive() :**
    - On fait appel à la fonction ‘getLumCaptee()’ de notre environnement pour obtenir la luminosité dans la salle
    - On initialise notre *‘chaine’* à “”
- **on_decide() :**
    - On regarde notre priorité et donc si on doit changer notre niveau ou pas
    - Si oui :
        - Selon la valeur captee et le seuil on met *‘chaine’* “augmenter”, “diminuer” ou “non”
- **on_act() :**
    - Selon *‘chaine’* on augmente notre niveau d’agent de 5 ou alors on le diminue de 5
    - On remet notre *‘chaine’* à “”

Préparer la simulation

Nous allons faire une simulation continue entre 8h et 21h dans laquelle nos agents devront se gérer afin de maintenir un seuil de luminosité à peu près constant dans la salle.
Nous devrons simuler la luminosité extérieure qui est extérieure au système mais qui agit sur la luminosité que les volets laissent entrer dans la salle.

Pour la simulation nous utiliserons pyAmak et donc python afin de coder nos classes amas, environnement et agents.

Pour les tests nous effectuerons un premier test référence pour voir le comportement de notre système puis des tests en augmentant l’importance de la luminosité venant de l’extérieur, puis des tests en changeant le seuil demandé dans la simulation.
Nous afficherons plusieurs graphes afin de voir notamment l’évolution des volets et lumières au cours de la simulation mais aussi l’évolution de la consommation qui est le principal enjeu de cette simulation.

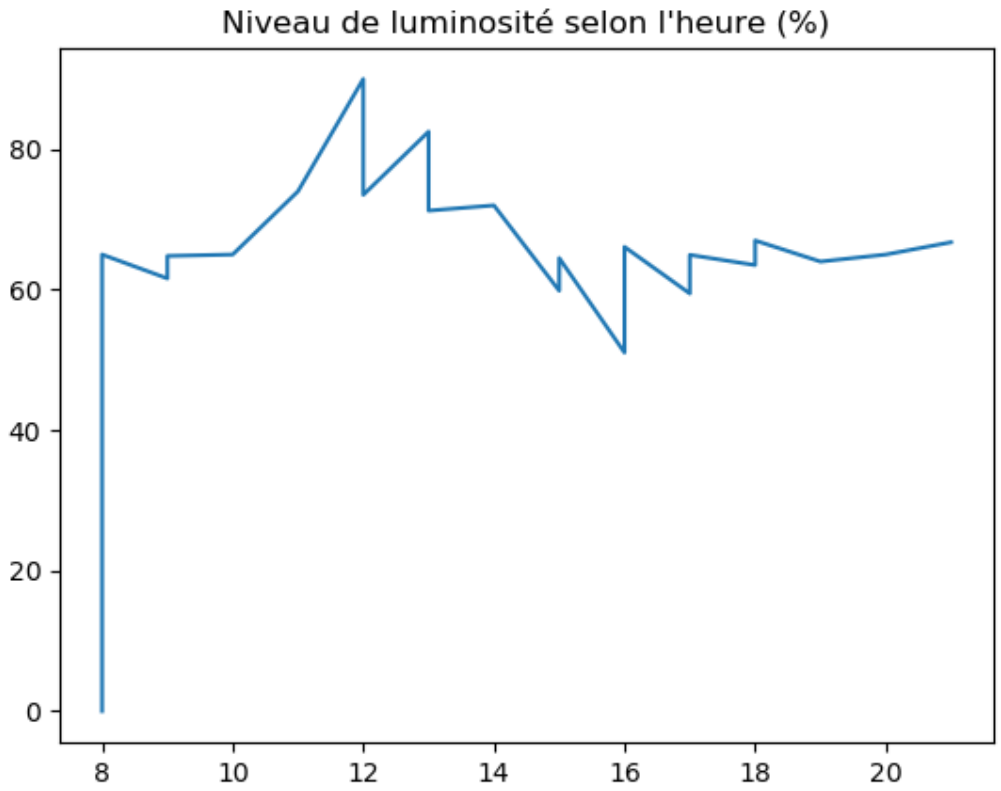
Simulation et Graphes

Nous avons fait plusieurs simulations afin de vérifier que l’objectif de minimiser la consommation de la salle était atteint.
Nous tirons de nos simulations plusieurs graphes :

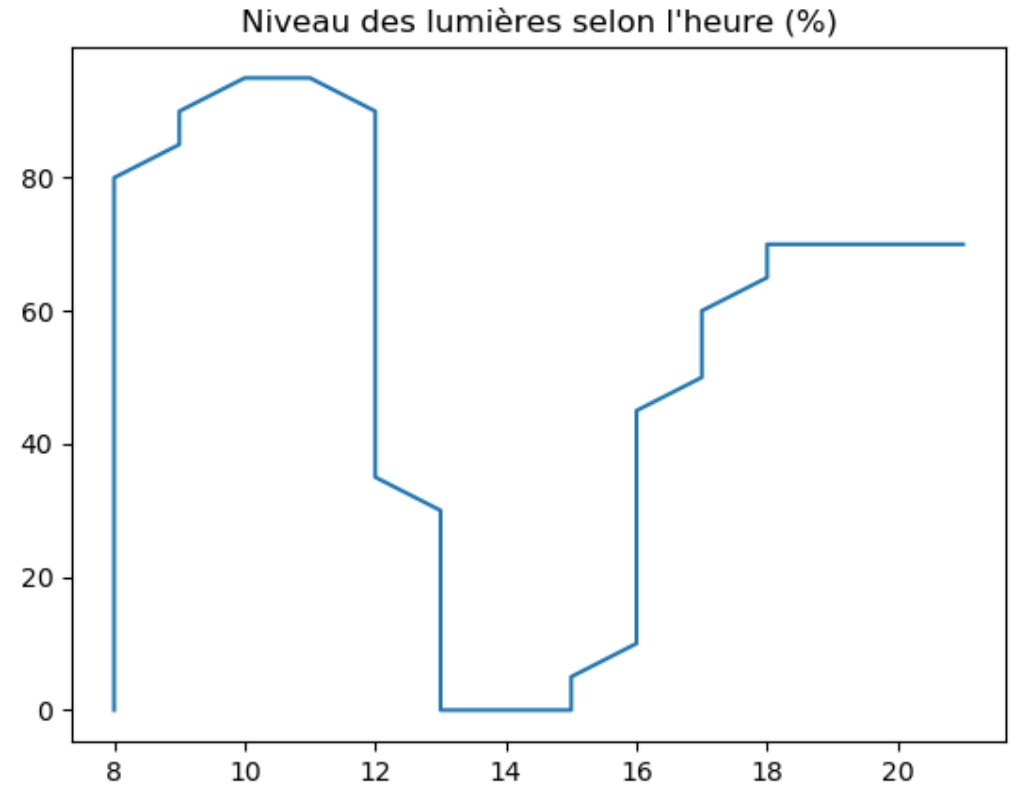
- Graphe de luminosité dans la salle
- Graphe de niveau des lumières
- Graphe de niveau des volets

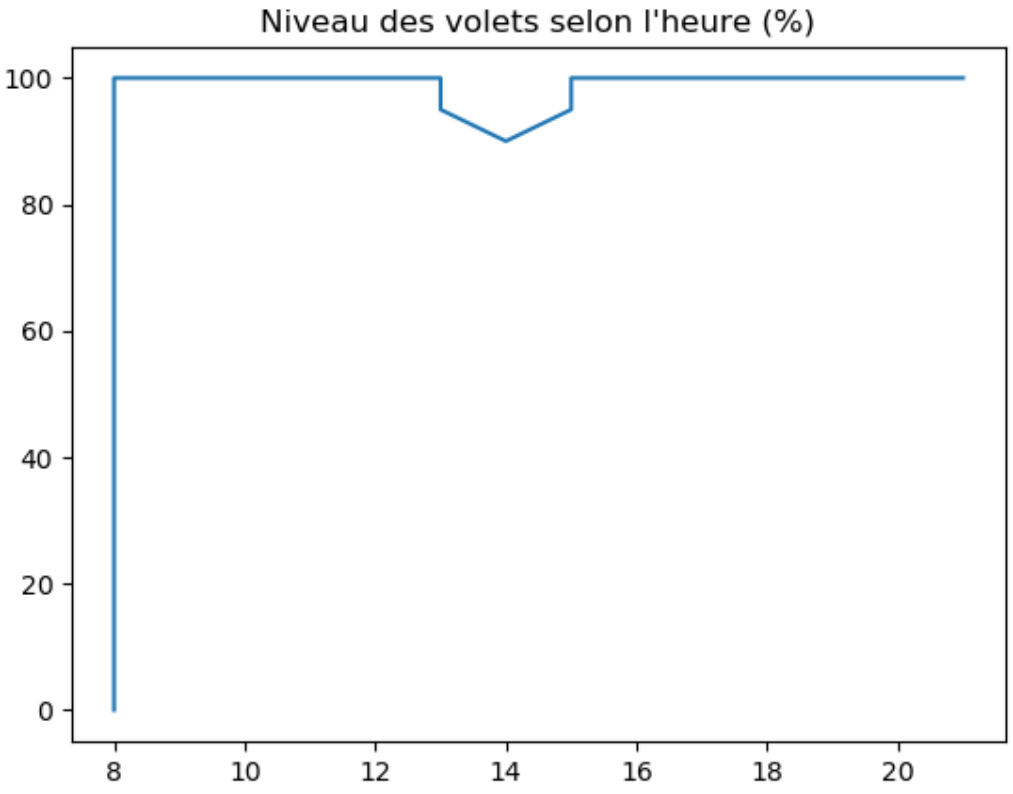
- Graphe de consommation générale
- Graphe de consommation par heure

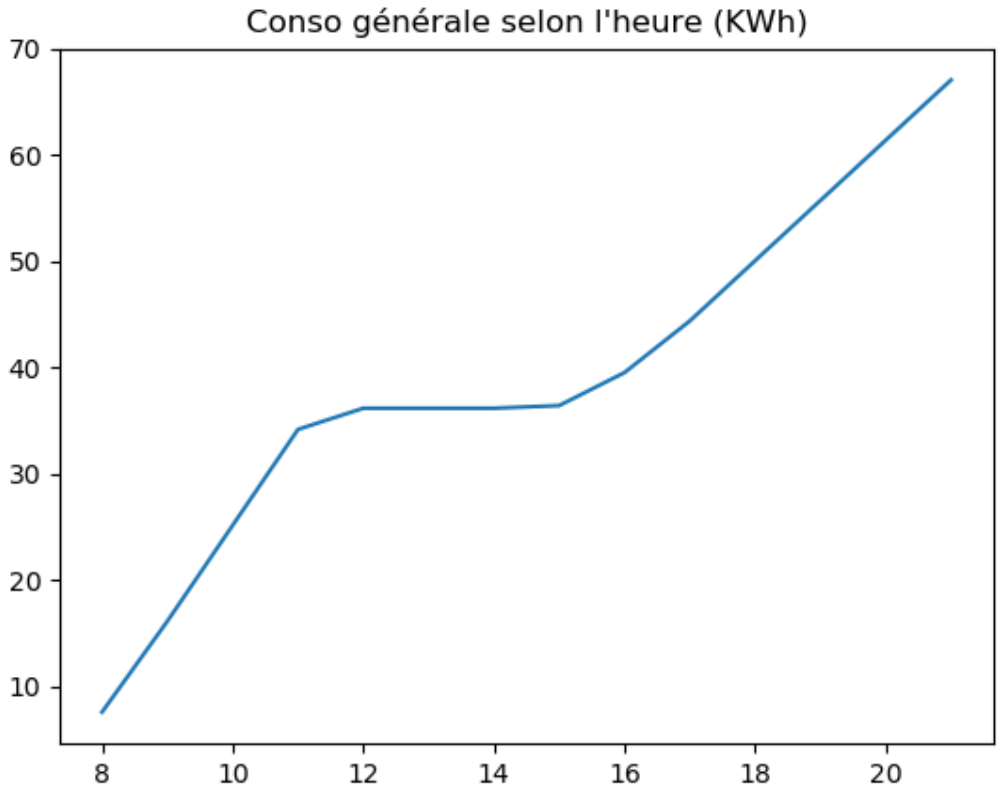
Nous avons donc une première simulation dans laquelle une journée passe normalement avec un seuil de 69% de luminosité.
Nous avons fait cette simulation avec un changement d'importance de la luminosité extérieure. Par exemple, la luminosité de notre salle est fournie à 80% par la lumière extérieure à 14h alors qu'elle l'est seulement à 20% à 8h.
Nous avons dans cette simulation les graphes suivants :

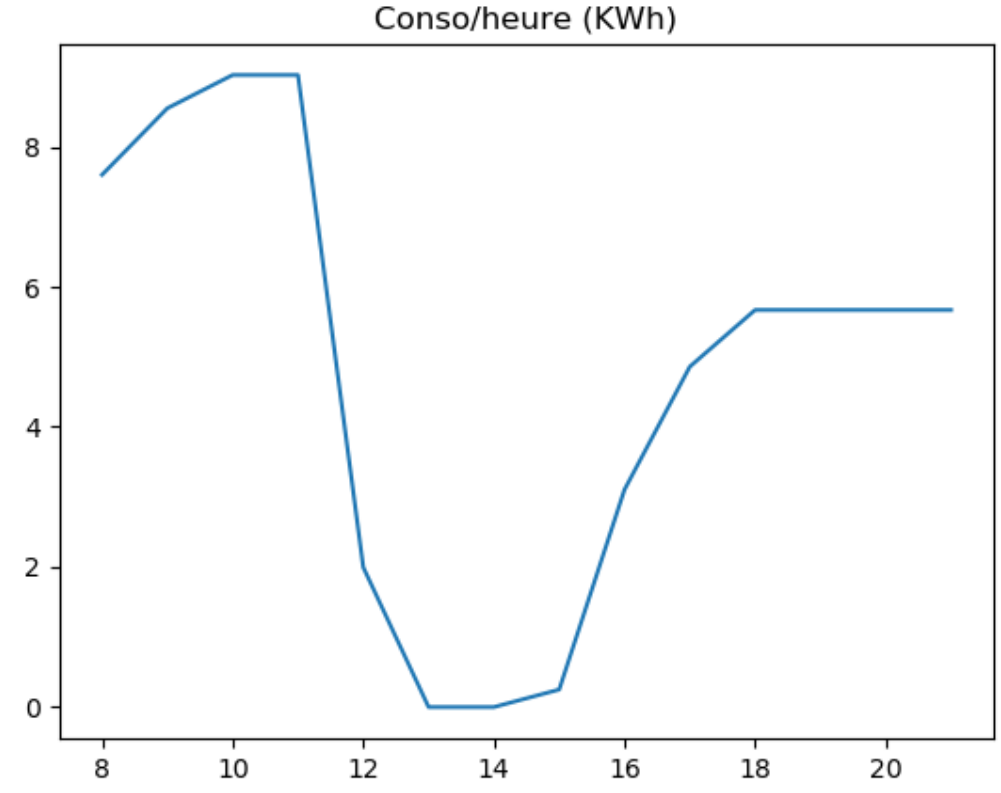


On peut observer sur ce graphique que la luminosité évolue d'elle même entre chaque heure puis qu'elle revient au seuil, ce qui est dû aux agissements de lumières et des volets. On retrouve donc pour chaque heure un pic puis une évolution vers une valeur dans l'intervalle [69-5 ; 69+5].

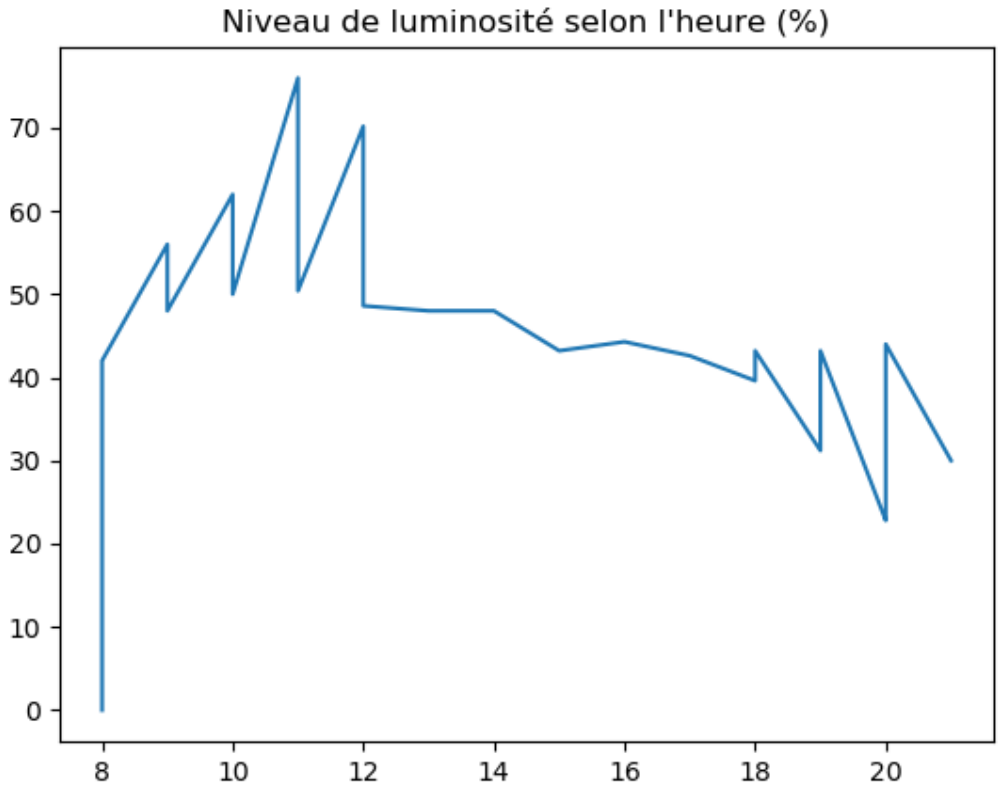

Niveau des lumières selon l'heure (%)
0
20
40
60
80
8
10
12
14
16
18
20


Niveau des volets selon l'heure (%)
0
20
40
60
80
100
8
10
12
14
16
18
20

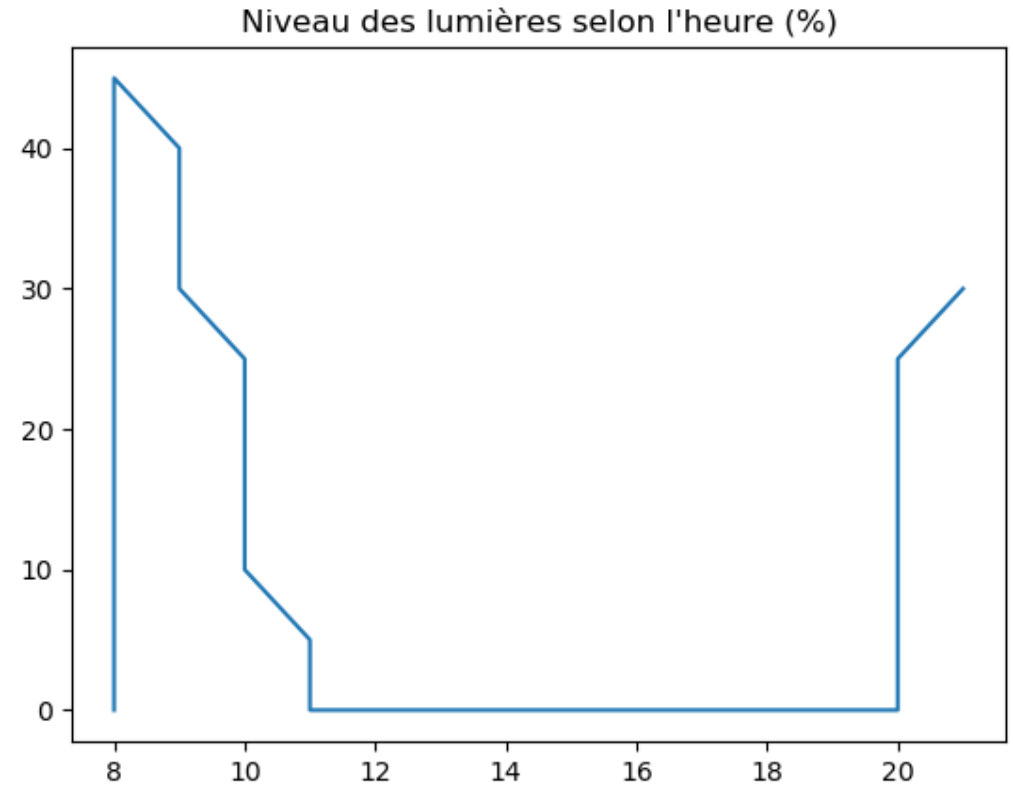

Conso générale selon l'heure (KWh)
70
60
50
40
30
20
10
8
10
12
14
16
18
20

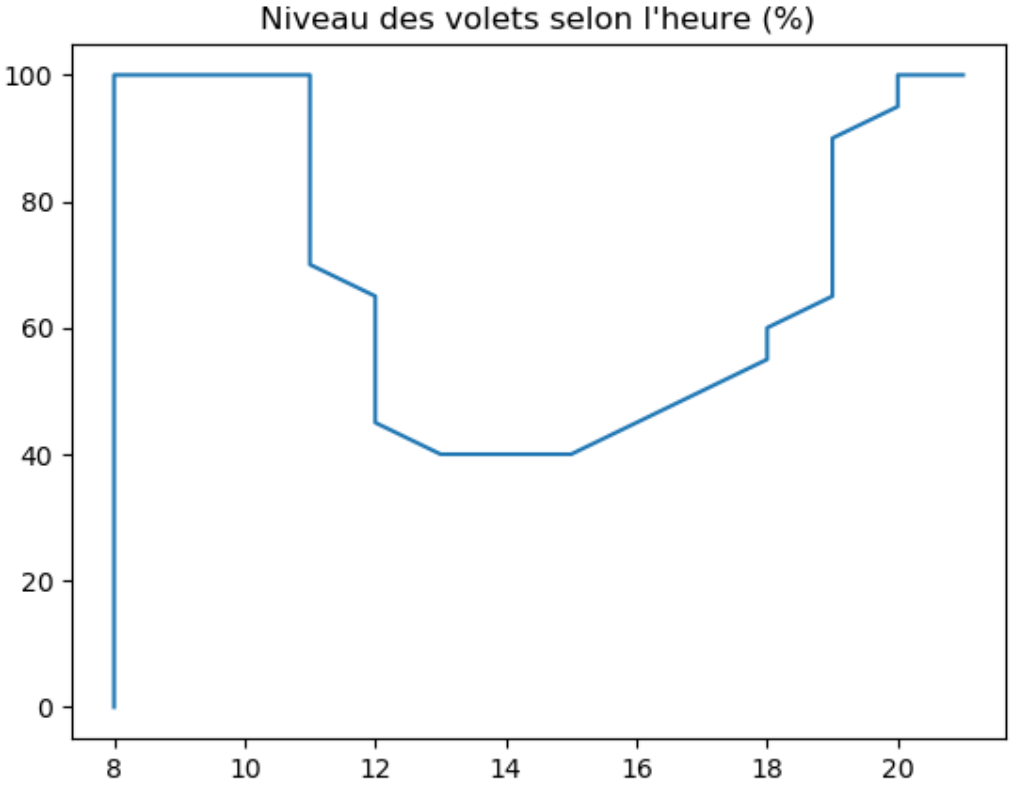

Conso/heure (KWh)
8
6
4
2
0
8
10
12
14
16
18
20

Une deuxième simulation a ensuite été réalisée dans laquelle nous augmentons l'incidence de la luminosité extérieure en multipliant la lumière extérieur par un taux afin de vérifier que les lumières s'éteignent bien quand il y a assez de luminosité et que les volets se baissent bien quand les lumières sont éteintes et qu'il reste encore trop de luminosité dans la salle.
Ce taux a une valeur de 6 dans cette simulation et nous avons un seuil de luminosité de 46%.
Contrairement à la simulation précédente nous avons fait cette simulation avec un ratio constant de luminosité, c'est-à-dire que la luminosité de la salle est calculée à partir de 80% de la lumière intérieure et 20% de la lumière extérieure pendant toute la simulation.
Nous avons les graphes suivants :

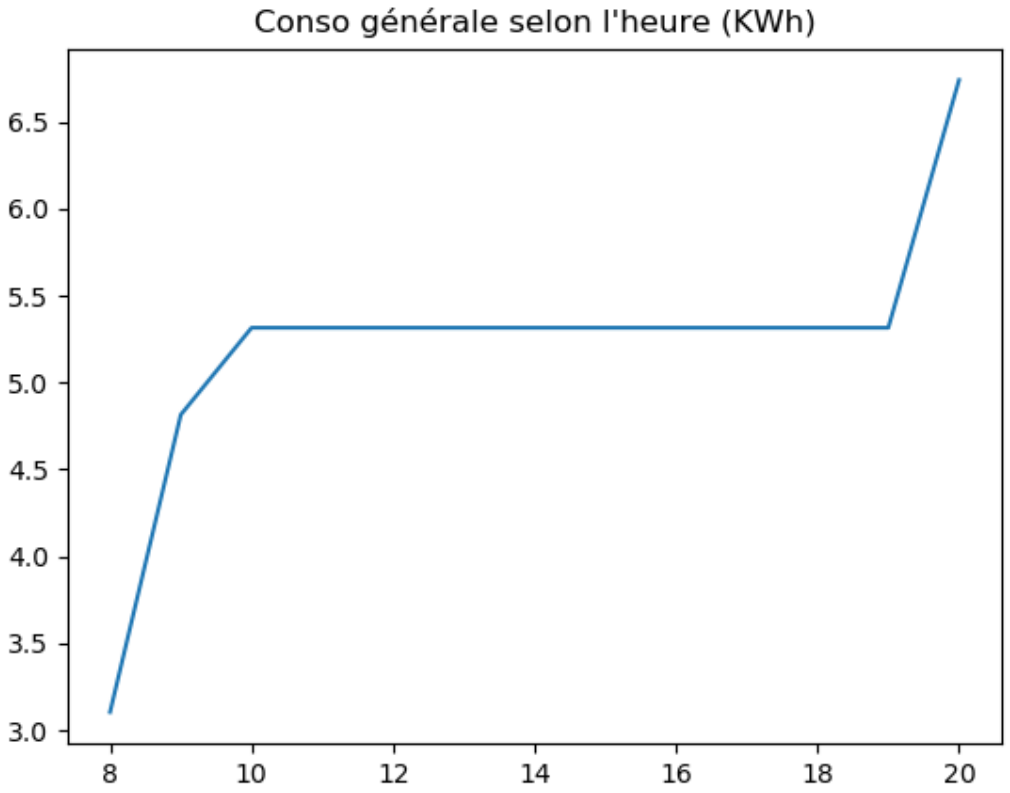


On observe qu'avec un taux de 6, la luminosité produite à l'extérieur est plus importante et a plus d'incidence sur la luminosité de la salle, bien que dans le calcul la valeur de luminosité extérieur ne représente que 20% de la luminosité de la salle. On retrouve cependant les ajustements du aux lumières et aux volets afin de revenir au seuil de 46% de luminosité.

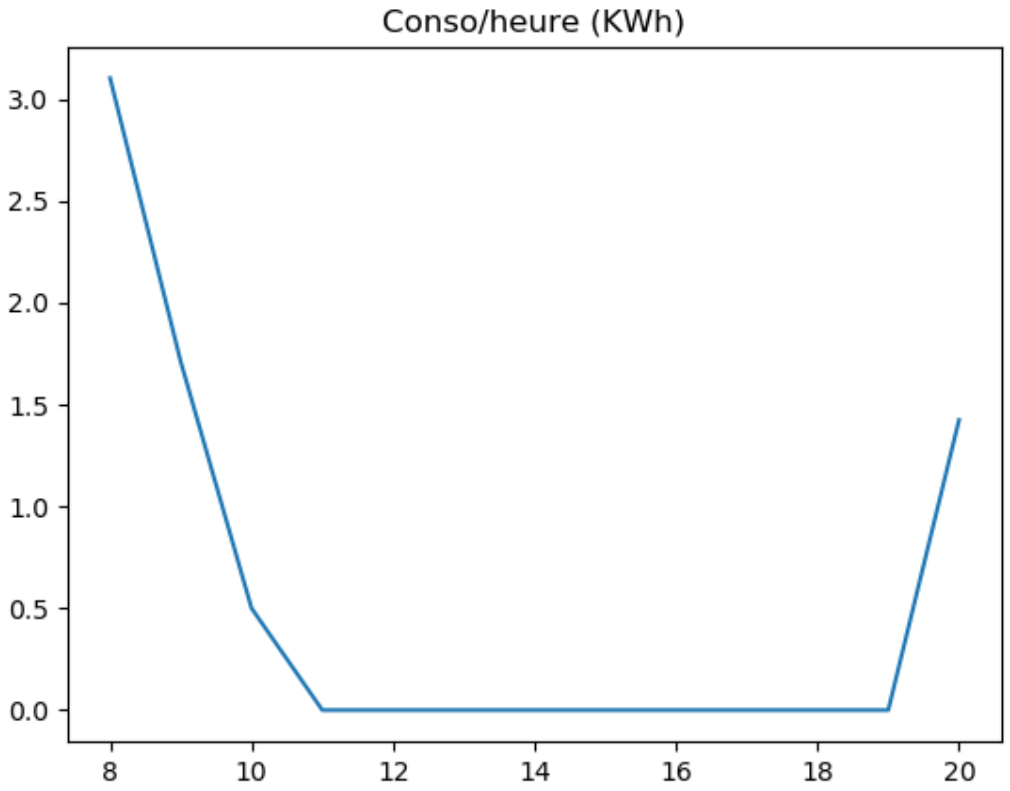

Niveau des lumières selon l'heure (%)
40
30
20
10
0
8
10
12
14
16
18
20


Niveau des volets selon l'heure (%)
100
80
60
40
20
0
8
10
12
14
16
18
20

On peut voir dans le simulation qu'avec un taux de 6 et donc plus d'incidence de la lumière extérieure sur la luminosité de la salle les lumières s'éteignent rapidement (en priorité) avant que les volets ne se ferment aussi s'il reste toujours trop de luminosité dans le salle.
Notre système fonctionne donc comme il faut.



On peut voir que la consommation générale grimpe au début et à la fin de la journée, donc quand la luminosité extérieure est faible, même avec un taux d'incidence de 6.
On peut aussi voir que le reste de la journée, il stagne car les lumières sont éteintes.



Sur ce graphe on peut observer que la consommation par heure diminue fortement jusqu'à devenir nulle lorsque les lumières sont éteintes.