Projet Optimisation sous incertitude

YADEL Gaya - CANNAVACCIUOLO Nello

Table des matières

1 Présentation et modélisation du problème :

Ce projet porte sur l'optimisation du choix des générateurs d'électricité afin de minimiser le coût total tout en répondant à une demande régionale incertaine. Il s'agit de déterminer les capacités optimales pour quatre types de générateurs (gaz, charbon et nucléaire). Deux types de coûts sont pris en compte : l'investissement initial pour l'acquisition des générateurs et le coût de production de l'énergie. Une distinction clé est faite entre les décisions d'investissement, qui doivent être prises à l'avance, et les décisions de production, qui peuvent être ajustées une fois la demande connue, illustrant ainsi une approche décisionnelle en deux étapes.

1.1 Fonction Objectif

Nous cherchons à minimiser le coût total défini comme suit :

$$\min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} f_j \beta_i y_{i,j} \tag{1}$$

1.2 Contraintes

— Contrainte de capacité minimale :

$$\sum_{j=1}^{n} x_j \geq M \tag{2}$$

— Contrainte budgétaire :

$$\sum_{j=1}^{n} c_j x_j \leq U \tag{3}$$

— Contraintes de production :

$$\sum_{i=1}^{m} y_{i,j} \leq x_j, \quad \forall j \in \{1, \ldots, n\} \tag{4}$$

$$\sum_{j=1}^{n} y_{i,j} = \mu_i, \quad \forall i \in \{1, \ldots, m\} \tag{5}$$

1.3 Données du problème :

- $U = 220$ est le budget,
- $M = 15$ est la capacité minimale (en kW) à assurer,
- $\mathbf{c} = (10, 7, 16, 6)$ sont les coûts de capacité par générateur,
- $\mathbf{f} = (40, 45, 32, 55)$ sont les coûts de production par générateur,
- $\beta = (1, 0.6, 0.1)$ sont les proportions de l'année correspondant à la durée de chaque segment,
- $\mu = (3.0013, 4.0, 5.0)$ sont les moyennes de la demande par segment.

1.4 Les scénarios

Scénario 1

$\mathrm{d}_1 values = (0.0, 0.5, 1.5, 3.0, 4.5, 5.5, 7.0, 7.5)$

$d_1 probs = (0.0013, 0.0215, 0.2857, 0.3830, 0.2857, 0.0215, 0.00125, 0.00005)$

Scénario 2

$\mathrm{d}_2 values = (0.0, 1.5, 2.5, 4.0, 5.5, 6.5, 8.0, 8.5)$

$d_2 probs = (0.0013, 0.0215, 0.2857, 0.3830, 0.2857, 0.0215, 0.00125, 0.00005)$

Scénario 3

$\mathrm{d}_3 values = (0.5, 1.0, 2.5, 3.5, 5.0, 6.5, 7.5, 9.0, 9.5)$

$d_3 probs = (0.00005, 0.00125, 0.0215, 0.2857, 0.3830, 0.2857, 0.0215, 0.00125, 0.00005)$

2 Réponse à la question 1

2.1 Objectif du Code

Le script Julia implémente et résout un modèle d'optimisation linéaire **déterministe** pour la planification des capacités de production électrique. L'objectif est de déterminer les capacités optimales (x_j) à installer pour chaque type de générateur j (parmi $n = 4$ types) afin de minimiser un coût total combiné, tout en satisfaisant une demande électrique moyenne et en respectant des contraintes budgétaires et de capacité minimale.

Ce modèle est dit "déterministe" car il prend en compte la demande électrique uniquement via ses valeurs **moyennes** (μ_i) pour chaque segment de temps i (parmi $m = 3$ segments), sans considérer l'incertitude ou la variabilité de cette demande. La solution obtenue, (x_j^*), représente la meilleure décision d'investissement *si l'on suppose que la demande future sera exactement égale à sa moyenne*.

2.2 Étapes Clés du Code

2.2.1 Initialisation et Définition des Données

- **Importation des bibliothèques :** Charge `JuMP` pour la modélisation et `GLPK` comme solveur d'optimisation linéaire.
- **Paramètres du problème :** Définit les constantes du modèle :
  - $n = 4$: Nombre de types de générateurs.
  - $m = 3$: Nombre de segments de demande.
  - $c = (c_j)_{j=1..n}$: Coûts d'investissement (€/kW).
  - $f = (f_j)_{j=1..n}$: Coûts de production (€/kWh).
  - $\beta = (\beta_i)_{i=1..m}$: Proportions de temps des segments (utilisées dans le calcul du coût de production).
  - $\mu = (\mu_i)_{i=1..m}$: Vecteur des demandes **moyennes** par segment (kW).
  - $U = 220$: Budget maximal d'investissement.
  - $M = 15$: Capacité totale minimale à installer (kW).

2.2.2 Construction du Modèle d'Optimisation

- **Création du Modèle :** Un objet `Model` JuMP est créé, en spécifiant `GLPK.Optimizer` comme solveur.
- **Variables de Décision :** Deux ensembles de variables sont définis :
  - $x_j \geq 0$ (`x[1:n]`) : Capacité à installer pour chaque type de générateur j. Ce sont les décisions d'investissement (première étape).
  - $y_{ij} \geq 0$ (`y[1:m, 1:n]`) : Quantité de production (ou allocation de puissance) du générateur j pour le segment i. Ce sont les décisions opérationnelles (deuxième

étape) pour satisfaire la demande *moyenne* μ_i.

— **Fonction Objectif :** L'objectif est de minimiser le coût total, défini comme la somme du coût d'investissement et d'un coût lié à la demande :

$$\min \left(\sum_{j=1}^{n} c_j x_j \right) + \left(\sum_{i=1}^{m} \sum_{j=1}^{n} f_j \beta_i y_{ij} \right)$$

— Le premier terme (`cout_dinvestissement`) est le coût total d'installation des capacités x_j.

— Le second terme (`cout_de_la_demande`) représente le coût opérationnel. Note : La multiplication par β_i (proportion du temps) mais pas par la durée totale T fait que ce terme n'est pas strictement un coût annuel total en Euros, mais plutôt une mesure de coût pondérée par la proportion de temps. C'est la formulation spécifique utilisée dans le code fourni.

— **Contraintes :** Plusieurs contraintes limitent les choix possibles :

1. **Capacité Minimale :** La somme des capacités installées doit atteindre le minimum requis M : $\sum_{j=1}^{n} x_j \geq M$.
2. **Budget :** Le coût total d'investissement ne doit pas dépasser le budget U : $\sum_{j=1}^{n} c_j x_j \leq U$.
3. **Limite de Production par Générateur :** Pour chaque générateur j, la somme de sa production (ou allocation) sur tous les segments i ne peut excéder sa capacité installée x_j : $\sum_{i=1}^{m} y_{ij} \leq x_j, \quad \forall j$. (C'est la contrainte spécifique utilisée ici).
4. **Satisfaction de la Demande Moyenne :** Pour chaque segment i, la production totale de tous les générateurs j doit être égale à la demande *moyenne* μ_i : $\sum_{j=1}^{n} y_{ij} = \mu_i, \quad \forall i$.

2.2.3 Résolution du Modèle

— L'instruction `optimize!(model)` lance le solveur GLPK pour trouver la solution optimale (x^*, y^*) qui minimise la fonction objectif tout en respectant toutes les contraintes.

2.3 Affichage des Résultats

— Le code affiche les valeurs optimales trouvées :

— `value.(x)` : Le vecteur des capacités optimales $x^* = (x_1^*, ..., x_n^*)$. C'est la **solution déterministe** `x_det`.

— `value.(y)` : La matrice des productions optimales $y^* = (y_{ij}^*)$ qui satisfont la demande moyenne μ_i.

— `objective_value(model)` : La valeur minimale du coût total combiné (selon la fonction objectif définie) atteinte par la solution optimale.

2.4 Affichage des résultats

FIGURE 1 – Résultats pour la question 1

```
julia> include("qst1.jl")
la solution d´eterministe :[4.0, 5.0, 3.0013, 2.9986999999999995]
Production optimale par segment :[0.0 0.0 3.0013 0.0; 4.0 0.0 0.0 0.0; 0.0 5.0 0.0 0.0]
Cout total optimal :355.55460000000005
```

3 Réponse à la question 2 :

3.1 Objectif du Code

Ce script Julia, encapsulé dans la fonction `evaluer_solution()`, vise à évaluer une solution de capacité fixe `x_det` (préalablement déterminée) face à l'incertitude de la demande. Il calcule une valeur finale qui combine le coût d'investissement de `x_det` avec une somme pondérée des coûts de production obtenus pour les scénarios de demande où la solution `x_det` s'avère suffisante.

3.2 Étapes Clés du Code

3.2.1 Initialisation et Données

- **Importation :** Charge `JuMP` et `GLPK`.
- **Paramètres :** Définit les constantes n, m, U, M, c, f, β.
- **Données Scénarios :** Définit les valeurs possibles et les probabilités pour la demande de chaque segment (d_1, d_2, d_3).
- **Solution Fixe :** Le vecteur `x_det` représente les capacités fixes à évaluer.

3.2.2 Préparation du Modèle de Sous-Problème

- **Création du Modèle :** Un **unique** modèle JuMP (`model`) est créé *avant* la boucle sur les scénarios. Ce modèle sera réutilisé et modifié à chaque itération.
- **Variables :** Seules les variables de second étage y_{ij} (`y[1:m,1:n]`) sont déclarées. Les variables x_j ne sont pas des variables de décision ici, mais des valeurs fixes `x_det[j]` utilisées dans les contraintes.
- **Contraintes Initiales :**
  - `puissancemin` et `budget` : Ces contraintes vérifient si la solution fixe `x_det` respecte les conditions de capacité minimale et de budget. Elles sont ajoutées au modèle mais n'affectent pas directement l'optimisation de y_{ij} (elles pourraient rendre le modèle initial infaisable si `x_det` était invalide, mais ce n'est pas leur rôle principal ici).
  - `distribuee[j=1:n]` : $\sum_{i=1}^{m} y_{ij} \leq x_{det,j}$. C'est la contrainte de capacité utilisant les valeurs fixes $x_{det,j}$.
  - `demande[i=1:m]` : $\sum_{j=1}^{n} y_{ij} = 0.0$. C'est la contrainte de satisfaction de la demande. Son membre de droite (0.0) sera **modifié** à chaque itération pour correspondre à la demande du scénario courant.
- **Fonction Objectif :** L'objectif est de minimiser le coût de production $\sum_{i=1}^{m} \sum_{j=1}^{n} f_j \beta_i y_{ij}$, cohérent avec la Q1.

3.2.3 Calcul Initial et Itération sur les Scénarios

- **Coût d'Investissement :** Le coût $\sum c_j x_{det,j}$ est calculé une fois et stocké dans `res`. C'est la base de la valeur finale.
- **Initialisation des Compteurs :** `scenario_index`, `nb_feasibles`, `total_scenarios` sont initialisés.
- **Boucle sur les Scénarios :** Trois boucles `for` imbriquées génèrent les 576 combinaisons de demandes $d = (d_1, d_2, d_3)$.
  - La probabilité p du scénario est calculée.
  - **Mise à Jour de la Contrainte de Demande :** L'instruction clé `set_normalized_rhs(dema d[i])` modifie le membre de droite de la contrainte de demande du modèle `model` pour correspondre à la demande d du scénario courant.
  - **Résolution du Sous-Problème :** `optimize!(model)` demande à GLPK de résoudre le problème de minimisation du coût de production avec la demande d actuelle et la capacité fixe x_{det}.
  - **Vérification et Accumulation :**
    - Si `termination_status == MOI.OPTIMAL`, le scénario est réalisable avec x_{det}.
      - Le coût de production optimal `objective_value(model)` est récupéré.
      - Ce coût est pondéré par la probabilité p et **ajouté** à la variable `res` (qui contient déjà le coût d'investissement).
      - Le compteur `nb_feasibles` est incrémenté.
    - Si le scénario est infaisable, rien n'est ajouté à `res`.

3.2.4 Affichage des Résultats

— Après le traitement de tous les scénarios, le code affiche des statistiques sur le nombre de scénarios traités et la faisabilité des solutions.

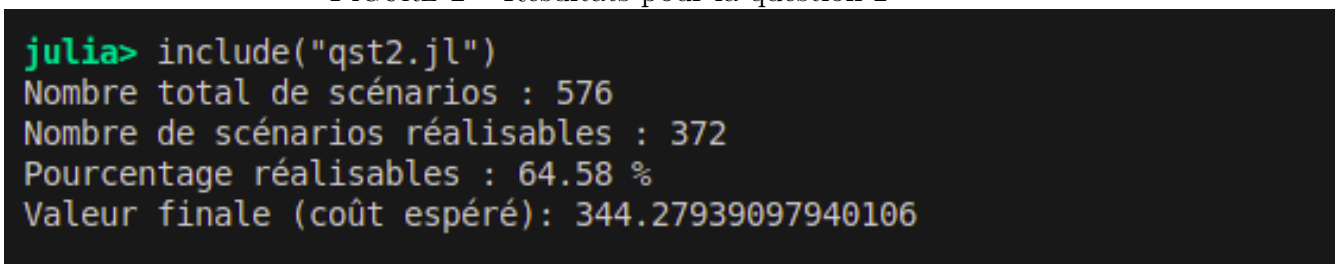
— **Calcul du coût moyen :** Le coût moyen conditionnel des scénarios réalisables est calculé comme suit :

$$E[CoutTotal|Realisable] = \frac{\sum p_s \cdot C_s}{\sum p_s}$$

où C_s est le coût total pour chaque scénario réalisable.

— Si des scénarios réalisables existent, le coût moyen est affiché. Sinon, un message d'erreur est affiché.

FIGURE 2 – Résultats pour la question 2



4 Réponse à la question 3

4.1 Objectif du Code

Le script Julia implémente et résout un modèle d'optimisation stochastique en deux étapes pour la planification des capacités de production électrique, en utilisant la **formulation extensive**. Contrairement au modèle déterministe (Question 1), ce modèle prend explicitement en compte l'incertitude de la demande future en considérant un ensemble de scénarios possibles.

L'objectif est de déterminer les capacités optimales (x_j) à installer (décision de première étape, prise avant de connaître la demande réelle) qui minimisent le coût total combiné, défini comme le coût d'investissement initial plus l'**espérance mathématique** du coût opérationnel futur, calculée sur l'ensemble des scénarios de demande considérés.

Une particularité de ce code spécifique est qu'il applique un **filtrage préalable** des scénarios, ne retenant qu'un sous-ensemble jugé pertinent selon un critère spécifique, avant de construire et résoudre le modèle.

4.2 Étapes Clés du Code

4.3 Initialisation et Définition des Données

- **Importation des bibliothèques :** Charge `JuMP` et `GLPK`.
- **Paramètres du problème :** Définit les constantes n, m, c, f, β, U, M comme dans le modèle déterministe.
- **Données Scénarios :** Les réalisations possibles de la demande (d_{is}) et leurs probabilités brutes (p_{is}) sont définies pour chaque segment i.

4.3.1 Génération et Filtrage des Scénarios

- **Génération Combinatoire :** Trois boucles `for` imbriquées génèrent les 576 combinaisons possibles de demandes ($d_s = (d_{1s}, d_{2s}, d_{3s})$).
- **Filtrage des Scénarios :** Une condition `if total <= M` est appliquée, où `total = d1 + d2 + d3` est la somme des demandes du scénario et $M = 15$ est la capacité minimale requise. Seuls les scénarios dont la somme des demandes est inférieure ou égale à M sont conservés.
- **Stockage :** Les scénarios filtrés (vecteur de demande d_s et probabilité brute p_s) sont stockés dans une liste `scenarios`.
- **Normalisation des Probabilités :** Les probabilités p_s des scénarios *retenus après filtrage* sont normalisées pour que leur somme soit égale à 1. Ceci est fait en divisant chaque p_s par la somme des probabilités brutes des scénarios filtrés (`total_p`).

— Le nombre de scénarios *effectivement utilisés* dans le modèle est affiché.

4.3.2 Construction du Modèle d'Optimisation (Formulation Extensive)

— **Création du Modèle :** Un objet `Model` JuMP est créé avec le solveur `GLPK`.
— **Variables de Décision :**
  — $x_j \geq 0$ (`x[1:n]`) : Variables de première étape (capacités à installer), identiques pour tous les scénarios.
  — $y_{sij} \geq 0$ (`y[1:length(scenarios), 1:m, 1:n]`) : Variables de deuxième étape (production/allocation). Elles sont spécifiques à chaque scénario s retenu, chaque segment i, et chaque générateur j.
— **Fonction Objectif :** L'objectif est de minimiser l'espérance du coût total :

$$\min \left(\sum_{j=1}^{n} c_j x_j \right) + \left(\sum_{s \in \mathcal{S}_{filtr}} p'_s \sum_{i=1}^{m} \sum_{j=1}^{n} f_j \beta_i y_{sij} \right)$$

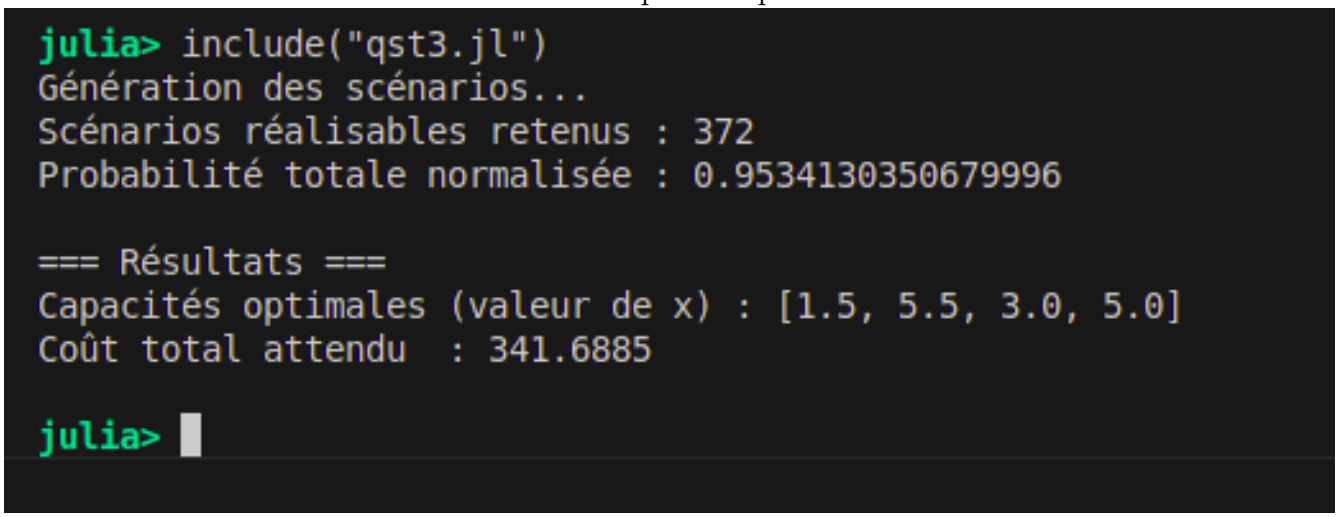
où $\mathcal{S}_{filtr}$ est l'ensemble des scénarios retenus après filtrage, et p'_s sont les probabilités normalisées.
  — Le premier terme (`invest_cost`) est le coût certain d'investissement.
  — Le second terme (`expected_prod_cost`) est l'espérance mathématique du coût opérationnel (calculé avec $f_j \beta_i y_{sij}$, cohérent avec Q1), calculée sur les scénarios *filtrés* et avec leurs probabilités *normalisées*.
— **Contraintes :**
  — **Première Étape :** Les contraintes $\sum x_j \geq M$ et $\sum c_j x_j \leq U$ s'appliquent aux variables x_j.
  — **Deuxième Étape (pour chaque scénario s filtré) :**
    — Limite de Capacité : $\sum_{i=1}^{m} y_{sij} \leq x_j, \quad \forall j$. La production pour le scénario s est limitée par la capacité x_j décidée en première étape.
    — Satisfaction Demande : $\sum_{j=1}^{n} y_{sij} = d_{is}, \quad \forall i$. La demande spécifique d_{is} du scénario s doit être satisfaite.

4.3.3 Résolution et Affichage des Résultats

— `optimize!(model)` résout le grand problème d'optimisation linéaire qui inclut les variables et contraintes de tous les scénarios filtrés.
— Le code affiche la solution optimale trouvée :
  — `value.(x)` : Le vecteur x^* des capacités optimales qui minimisent le coût total attendu *sur les scénarios filtrés*. C'est la **solution stochastique**.
  — `objective_value(model)` : La valeur minimale de la fonction objectif (coût total attendu sur les scénarios filtrés).

4.3.4 Affichage des résultats

FIGURE 3 – Résultats pour la question 3



5 Réponse à la question avec L-shaped

Nous avons fait un test dans notre code (fichier "Lshaped.jl") mais sans résultats convainquants...

FIGURE 4 – Résultats pour la question 3 avec Lshape