



ÉCOLE
D'INGÉNIEURS
PARIS-LA DÉFENSE

CSTB
le futur en construction

Développement Python et l'Optimisation
ZHANG HAO, 5E ANNÉE

MAJEURE IBO

Introduction de stage

- ▶ **Période de stage** : du 6 avril 2019 au 5 novembre 2019 (6 mois)
- ▶ **Entreprise** : CSTB, Centre Scientifique et Technique du Bâtiment
- ▶ **Département** : DA2E, Direction Analyse et Études économiques
- ▶ **Mission** : développement Python et l'optimisation
- ▶ **Mots-clés** : Python, modélisation, simulation prospective, optimisation, vectorisation
- ▶ **3 périodes de processus :**
 - Traduire le programme VBA à Python
 - Flétrir et chercher des solutions pour optimiser ce programme
 - pratiquer etachever de la réalisation de la vectorisation du programme

Contexte de stage

- ▶ La DA2E du CSTB a développé ces dernières années un modèle (Zephyr-chaleur) d'optimisation des investissements en rénovation énergétique.
- ▶ Ce modèle repose sur une modélisation technico-économique du parc résidentiel français et permet de déterminer les combinaisons d'opérations de **rénovation** :
 - isolation des différentes composantes de l'enveloppe thermiques des bâtiments
 - remplacement des systèmes de chauffage
- ▶ Ce modèle permet aussi de minimiser le coût d'atteinte de différentes cibles de réduction d'émissions à l'horizon 2035 sous diverses **contraintes** :
 - plafond annuel de rénovation
 - enveloppe annuelle d'investissement

Objectifs de stage

- ▶ DA2E espère réécrire le programme VBA original dans un programme Python et effectuer une optimisation plus appropriée, afin de :
 - Améliorer l'efficacité opérationnelle de ce modèle
 - Optimiser la conception de son programme
 - Mieux potentiellement coopérer avec d'autres modèles d'analyse d'autre direction de CSTB

Mes missions

1. Découverte du modèle Zephyr-chaleur
2. Connaissance des données utilisées dans le projet

Table 1 – Wall Investment

Collective dwelling
Wall Investment
Um
Price including Tax
Individual dwelling
Wall Investment
Um
Price including Tax

Table 2 – Window Investment

Collective dwelling
Window Investment
Uw
Price including Tax
Individual dwelling
Window Investment
Uw
Price including Tax

Table 3 – Roof Investment

Collective dwelling
Roof Investment
Ur
Price including Tax
Individual dwelling
Roof Investment
Ur
Price including Tax

Table 4 – Floor Investment

Collective dwelling
Floor Investment
Ur
Price including Tax
Individual dwelling
Floor Investment
Ur
Price including Tax

Table 5 – Economic Hypothesis

Dwelling Type
Carbon Price
Discount Rate
Inflation Rate
Insulation Limit

Table 6 – Climate Zone

Climate Type
Degrees of Reference

Table 7 – Housing Characteristics

Collective dwelling

Housing Type
Construction Period
Size Factor
Surface
Ceiling Height
Share of Windows
Dwelling per Floor
Floors
Intermittence
Um0
Uw0
Ur0
Uf0

Individual dwelling

Housing Type
Construction Period
Size Factor
Surface
Ceiling Height
Share of Windows
Dwelling per Floor

Table 8 – Heating System

CID

Heating System
Efficiency
A Investment
B Investment
Maintenance Cost
Emission Factor
Energy Price
Same Switch Cost
Different Switch Cost
Delevery Type Cost
Specific Gas Price
Annual Variation

CCD

Heating System
Efficiency
A Investment
B Investment
Maintenance Cost
Emission Factor
Energy Price
Same Switch Cost

Table 9 – Headcount

CID

Dwelling Type
Climate Zone
Construction Period
Size Factor
Heating System
Initial Age
Headcount

CCD

Dwelling Type
Climate Zone
Construction Period
Size Factor
Heating System
Initial Age
Headcount

ID

Dwelling Type
Climate Zone
Construction Period
Size Factor

Mes missions

► 3. Implémentation de la programmation Python

« Dimosim » → programmation orienté objet (POO)

a. Résultats obtenus

- avoir comparé un total de 26 logements de trois types de logements avec différents systèmes de chauffage dans la même zone climatique (1/3), la période de construction (1/7), la taille (1/3) et la durée de vie du système de chauffage (1/20).
- Le GHC minimum des trois types de maisons (CID, CCD, ID) est obtenu en comparant le GHC correspondant à 16 416 types, 14 592 types et 23040 types de systèmes d'isolation (selon les différentes permutations et combinaisons des quatre composants d'isolation thermique de mur, fenêtre, plafond et sol).

Mes missions

► 3. Implémentation de la programmation Python

b. Défauts de conception

- Bien que j'ai obtenu correctement les résultats de calcul de 26 types de logements via la programmation Python, le temps nécessaire à l'ordinateur pour calculer les résultats pour chaque année est d'environ 10 minutes.
- Et je dois faire des prévisions pour 32 760 maisons au total. Si je suis cette procédure, je ne peux pas obtenir les résultats rapidement, même si cela est beaucoup plus long que le temps de calcul du programme original de VBA (environ une à deux heures). Par conséquent, ce programme est inacceptable.

Mes missions

4. Implémentation de la programmation Python avec la vectorisation

a. Retraitements des formules

$$\begin{aligned}
 D_{GHC_1} &= \frac{D_{investment_cost} \times HS_{inflation_rate_1st}}{HS_{discount_1st}} + \frac{D_{investment_cost} \times HS_{inflation_rate_2nd}}{HS_{discount_2nd}} \\
 &\quad + \frac{D_{uen_per_dwelling}}{HS_{efficiency}} \times HS_{energy_bill} + HS_{maintenance} + D_{insulation_cost} \\
 &= (HS_{a_investment} + HS_{b_investment} \times D_{uen_per_dwelling}) \\
 &\quad \times \frac{HS_{inflation_rate_1st} \times HS_{discount_2nd} + HS_{inflation_rate_2nd} \times HS_{discount_1st}}{HS_{discount_1st} \times HS_{discount_2nd}} \\
 &\quad + D_{uen_per_dwelling} \times \frac{HS_{energy_bill}}{HS_{efficiency}} + HS_{maintenance} + D_{insulation_cost} \\
 &= HS_{ghc_uen_condition_1} + D_{unvariable_cost_1}
 \end{aligned} \tag{22}$$

• Where:

$$D_{investment_cost} = HS_{a_investment} + HS_{b_investment} \times D_{uen_per_dwelling} \tag{23}$$

$$D_{unvariable_cost_1} = D_{insulation_cost} + HS_{fixed_cost_1} \tag{33}$$

if $HS_{to_switch} \in \{ER, GB, CGB, OB, COB, EHP2, WB, WPB, EHP\}$
 $\Rightarrow HS_{initial} \in \{ER, GB, CGB, OB, COB, EHP2, WB, WPB, EHP\}$
and $HS_{to_switch} = HS_{initial}$

$$D_{insulation_cost} = D_{insulation_cost_m_w} + D_{insulation_cost_r_f} \tag{34}$$

$$\begin{aligned}
 D_{insulation_cost_m_w} &= \frac{B_{surface_m}}{B_{dwelling_total}} \times D_{price_including_tax_m} \\
 &\quad + \frac{B_{surface_w}}{B_{dwelling_total}} \times D_{price_including_tax_w}
 \end{aligned} \tag{35}$$

$$\begin{aligned}
 D_{insulation_cost_r_f} &= \frac{D_{surface}}{B_{floors} + 1} \times D_{price_including_tax_r} \\
 &\quad + \frac{D_{surface}}{B_{floors} + 1} \times D_{price_including_tax_f}
 \end{aligned} \tag{36}$$

$$HS_{fixed_cost_1} = HS_{a_investment} \times HS_{time_factor_1} + HS_{maintenance} \tag{37}$$

$$HS_{maintenance} = \sum_{i=0}^{39} \frac{HS_{maintenance_cost} \times (1 + HS_{inflation_rate})^{i+year-1}}{(1 + D_{discount_rate})^i}, year \in [1, 20] \tag{38}$$

Mes missions

► 4. implémentation de la programmation Python avec la vectorisation

b. Intégration des données d'original

- La clé de l'intégration des données est l'utilisation d' « `itertools.product()` » qui produit cartésien.
- D'abord diviser les mots-clés liés à la zone climatique (3 au total), à la période de construction (7 au total), à la taille (3 au total), aux systèmes de chauffage (9 au total) et à l'année d'utilisation du système de chauffage (20 au total), générer 5 listes indépendantes.
- Ensuite, fusionner et combiner `itertools.product ()` pour générer une table 11340×5 pour CID et ID ou une table 10080×5 pour CCD.
- Enfin, combiner les données restantes avec `pandas.merge` ou `pandas.concat`.

Mes missions

c. Structure du programme

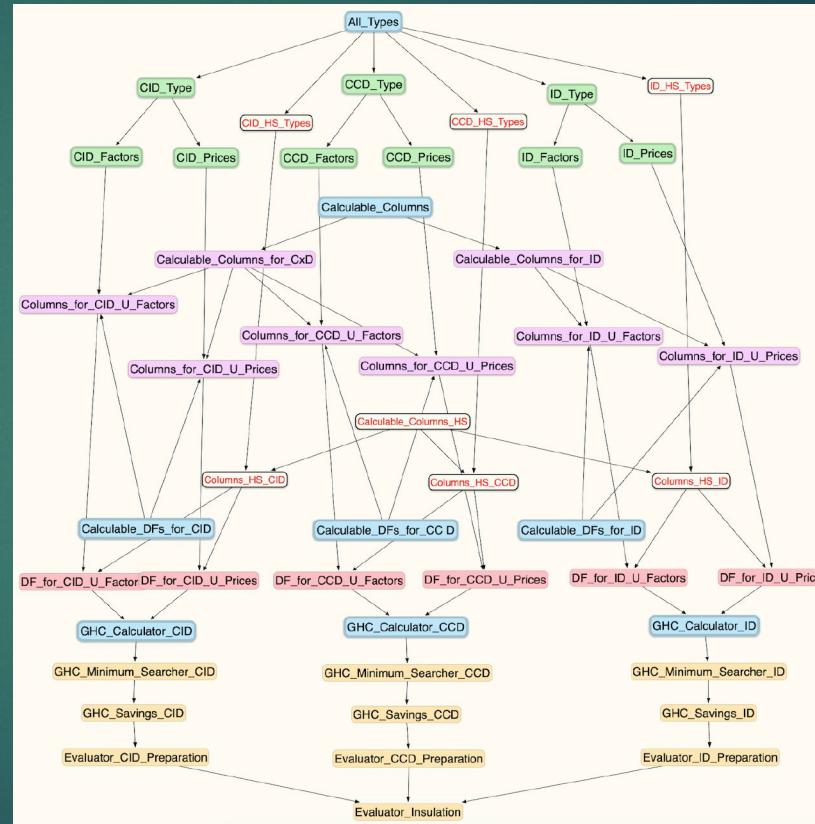
Fusionner les données pour générer une table contenant toutes les données ;

Générer des colonnes pouvant être calculées sans itération ;

Par itération, générer les colonnes pouvant être itérées par calcul ;

Toutes les colonnes itérables générées sous forme de table pour participer au calcul ;

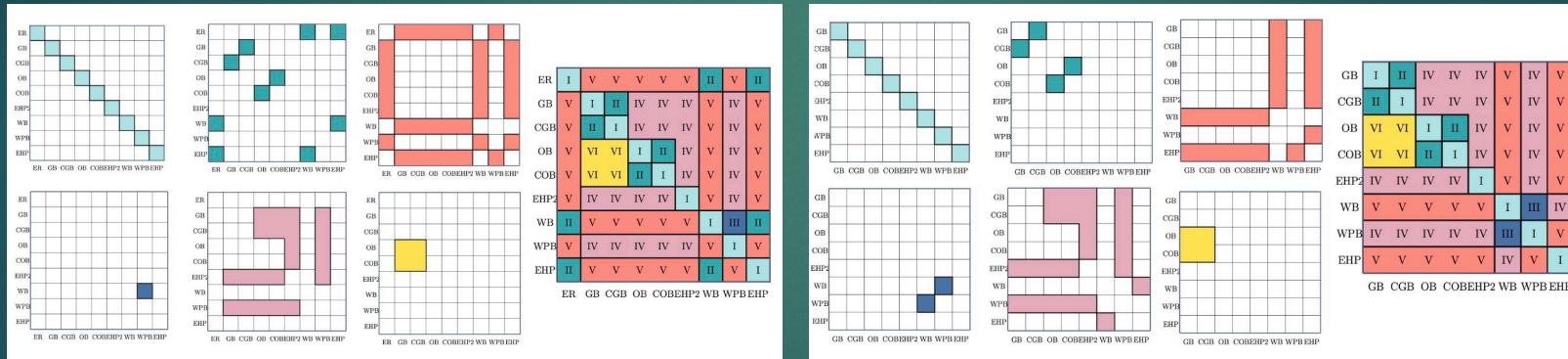
Selon les conditions correspondantes, différentes formules sont automatiquement sélectionnées dans le calcul.



Mes missions

4. Implémentation de la programmation Python avec la vectorisation

d. Programmation vectorisée comme un puzzle



Mes missions

► 4. implémentation de la programmation Python avec la vectorisation

e. Effets opérationnels et recommandations futures

- Grâce au traitement de la programmation vectorisée, la vitesse d'exécution finale du programme peut atteindre environ 10 minutes par année de test.
- En raison du nombre considérable de calculs effectués à chaque fois, il a été constaté lors de l'examen du logiciel complet au cours des années qu'il nécessitait 4-5Go de mémoire.
- Pour un ordinateur général, la vitesse du programme varie en fonction de la mémoire ou du processeur de l'ordinateur.
- En raison de la durée du stage, je n'ai pas eu la possibilité de combiner des méthodes parallèles et vectorisées pour augmenter la vitesse des opérations.
- Future : threading de Python

Conclusion

- ▶ Ce stage a été une chance pour moi, après plusieurs mois de la recherche afin que je puisse commencer mon premier stage formel dans le domaine d'informatique en France. Heureusement, j'ai pu être admis par cette offre de CSTB, et passer une inoubliable durée de 6 mois avec une bonne équipe.
- ▶ Après six mois de stage, j'ai également commencé à repenser mes objectifs de développement futurs. Après six mois de développement logiciel, en particulier concernant l'optimisation et la vectorisation de programmes, je me suis également intéressé aux structures de données et aux algorithmes. Si j'en ai l'occasion, je pense qu'il serait intéressant de poursuivre mes recherches et mes études dans ce domaine. Parallèlement, au cours de chercher mon prochain stage, j'ai eu l'idée de participer à des tâches plus complexes et plus difficiles, telles que l'apprentissage automatique et même l'intelligence artificielle, une idée que je n'osais pas avoir auparavant.