TIPE: Modélisation informatique de la consommation d'un véhicule hybride sur un parcours défini.

Victor Hecquet - MP Option SII - 18397

Epreuve de TIPE

Session 2021

Est-ce qu'à l'aide d'un modèle simplifié et informatisé, je suis en mesure de prédire qu'en adoptant un véhicule hybride je ferai baisser ma consommation ?

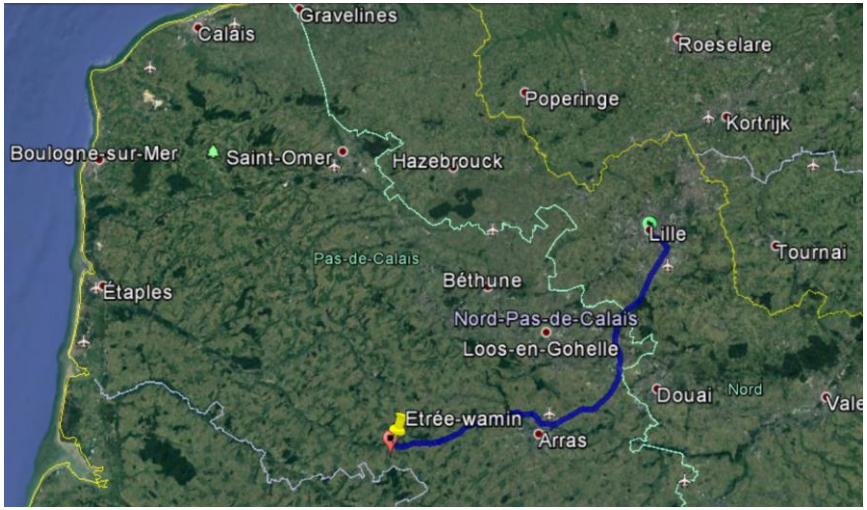


Figure 1

Plan :

I. 1ère Modélisation informatique

II. Précision des phénomènes et insertion des constantes physiques

III. Applications Numériques et Résultats

1.1 Contextualisation : Le trajet

Figure 1: Trajet avec Google Earth

- Distance : 81,7km
- Dénivelé: 122m
- Durée moyenne: 1h10’
- Vitesse moyenne : 100km/h

Discrétisation de l'étude :

Durée du trajet : 1h10min ⇒ N = 4200s + Pas =1s

On évalue au cours du temps :

- **Res[t] (Litres)**
- **Bat[t] (Pourcents)**
- **Mode[t]** (10 pour mode électrique ou 0 pour thermique)

1.2 Les bases du 1er modèle

À t=0:

Rentre :

- **Valeur initiale de la batterie** (%)
- **Volume initial du réservoir** (L)

1.2 Les bases du 1er modèle

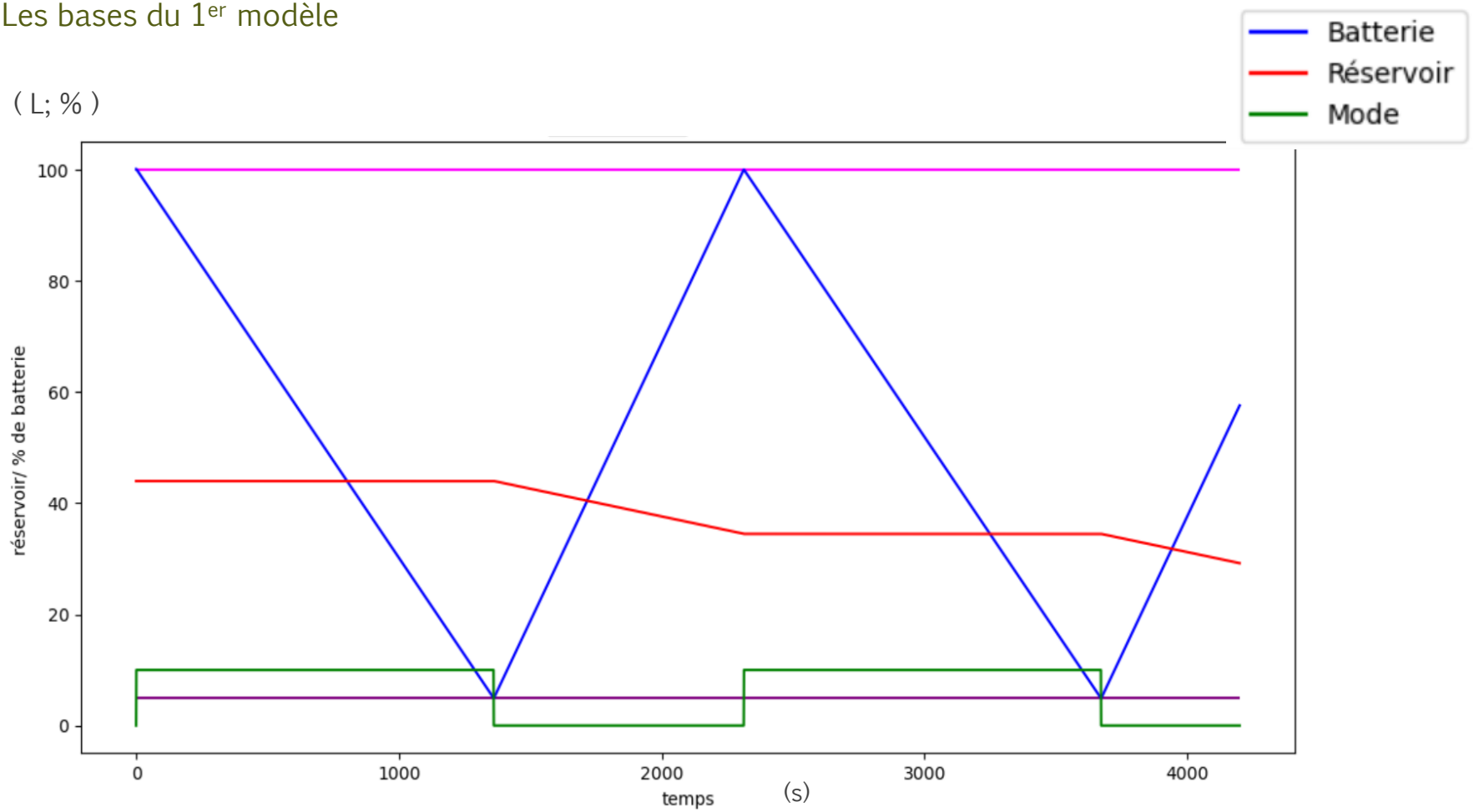
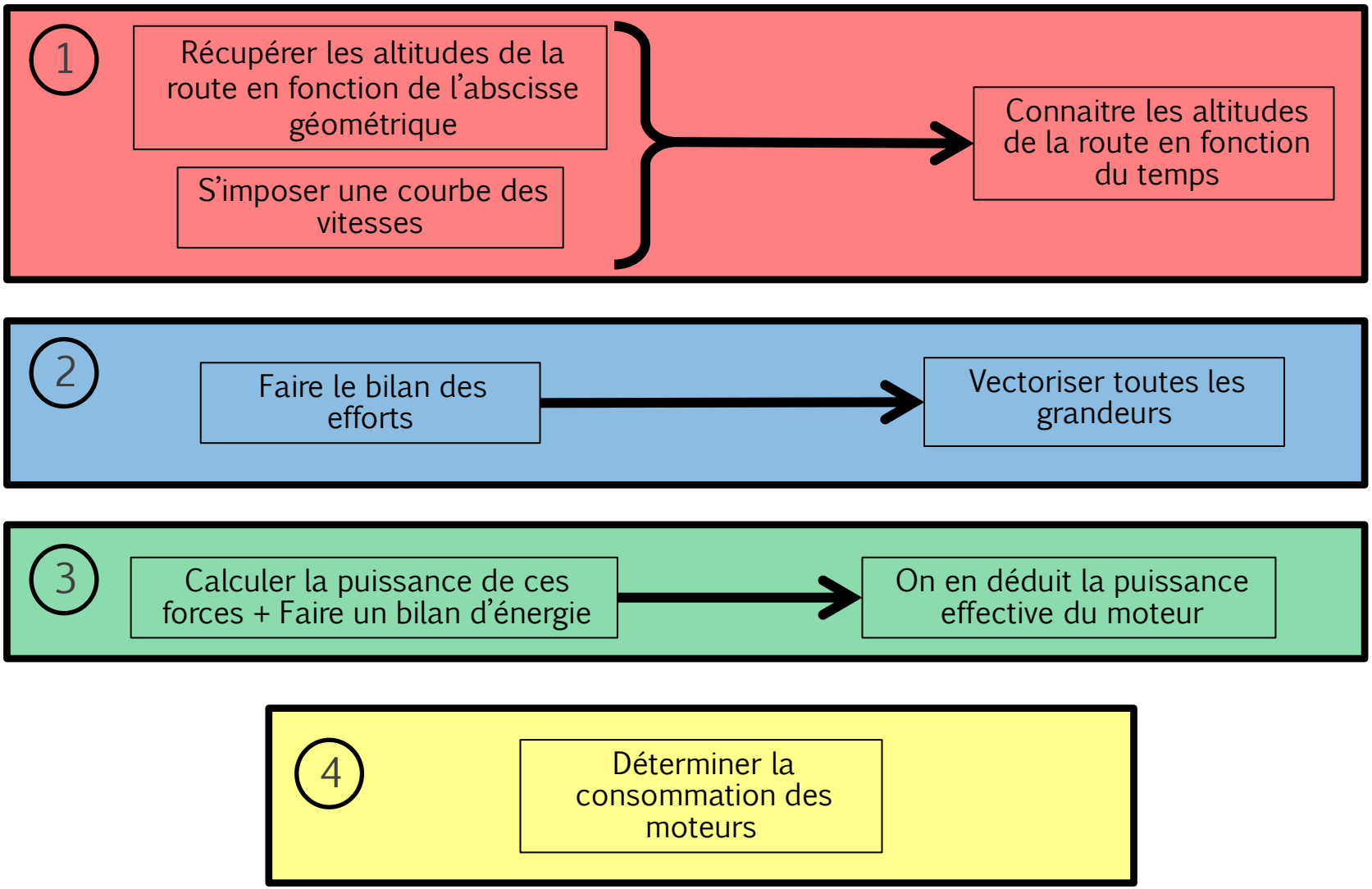


Figure 2: Graphique primaire d'évolution des réserves en énergie

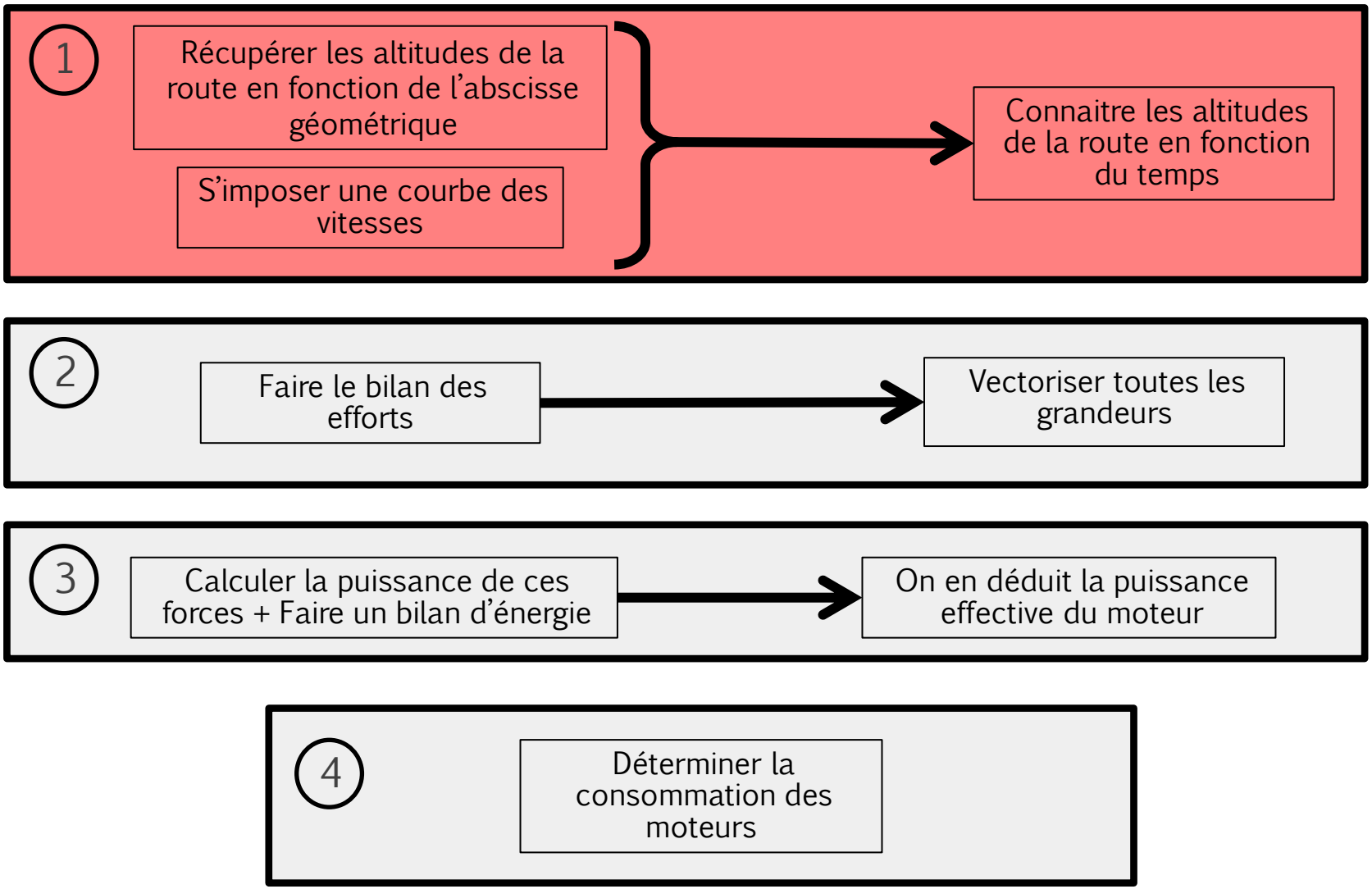
Valeurs arbitraires:

Décharge de la batterie = 0.06 (%/s)	Recharge de la batterie =0.1 (%/s)	Consommation en essence = 0.01 (L/s)

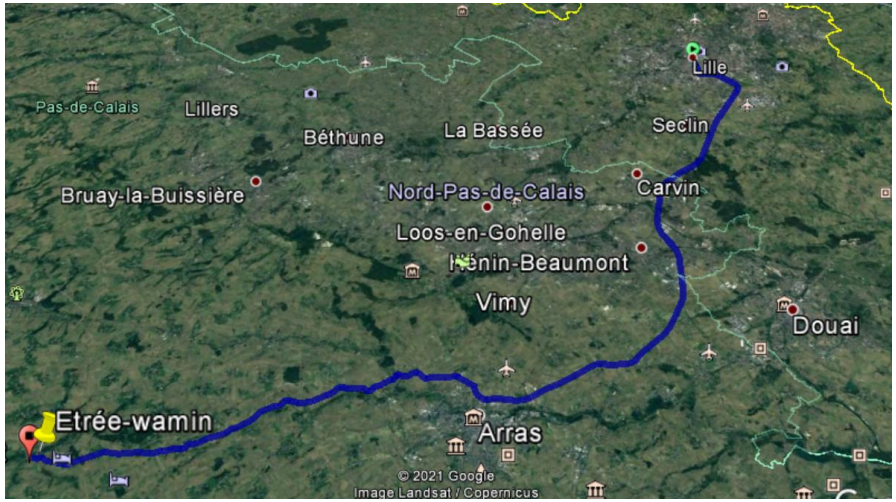
2. Comment calculer la consommation au cours du temps ?



2.1 Liste des altitudes de la route



2.1 Liste des altitudes de la route



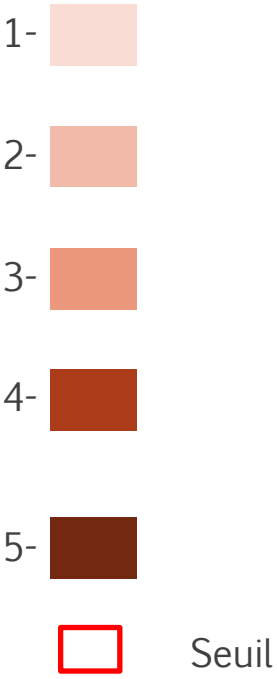
Utilisation de **Google Earth** afin d'obtenir **gratuitement** le **profil d'élévation** d'un trajet.

But: créer une liste des ordonnées de la route.

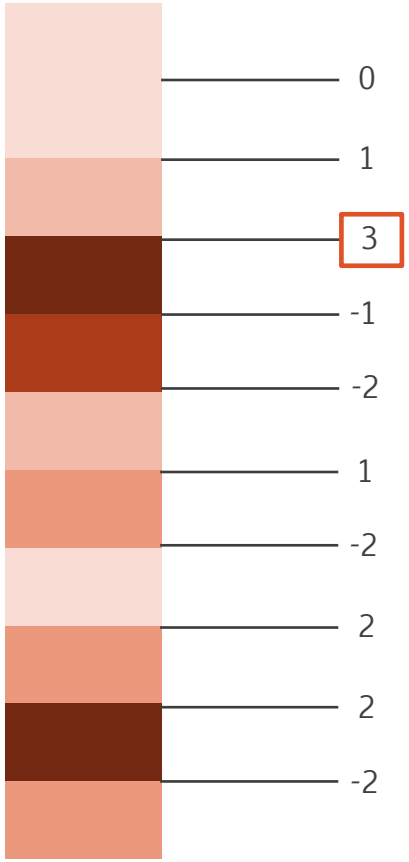
Figure 3: Profil des altitudes

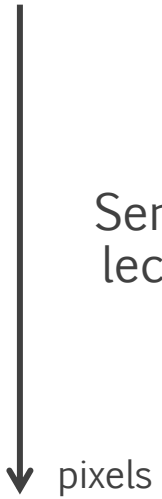
2.1 Liste des altitudes de la route

Notons les intensités de la couleur comme suit:



Gradient d'intensité de couleur





Pour **récupérer la frontière** :

J'utilise une fonction «**dérivée**» qui prend en note la variation de teinte dans la colonne.

2.1 Liste des altitudes de la route

Puis je **remplis la liste des ordonnées** de gauche à droite

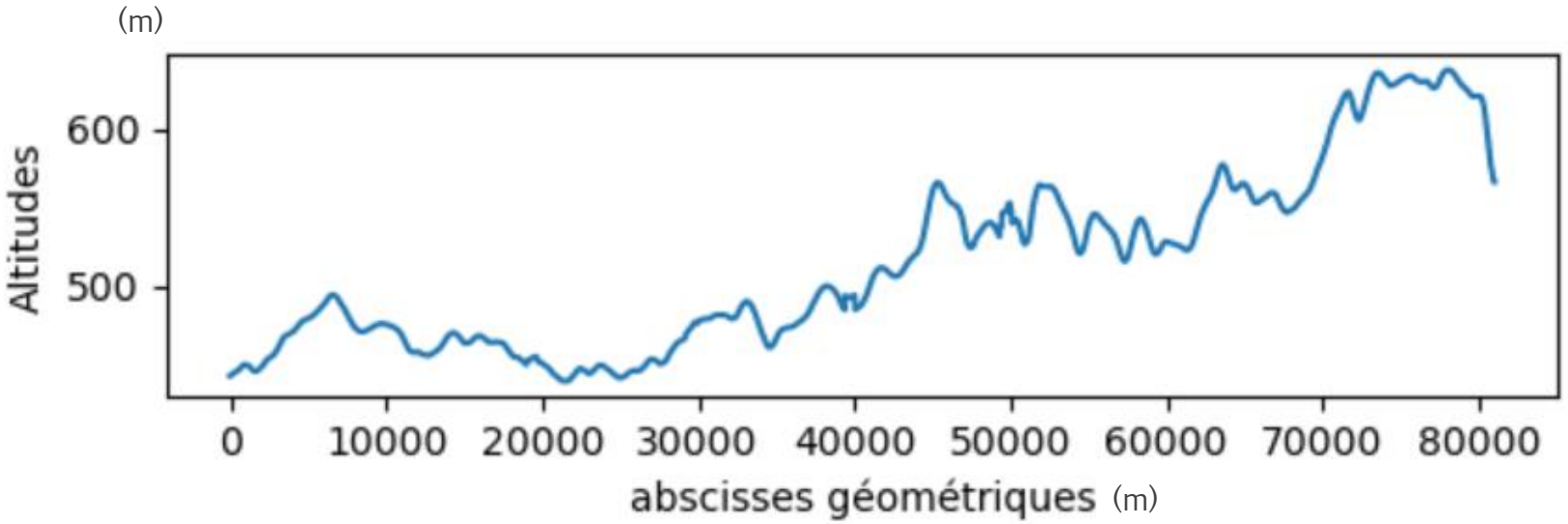


Figure 4: Liste des altitudes

Résultat obtenu contient du **bruit** dû aux voitures et ponts sur le trajet

On applique un **passe Bas** et un **Lissage mobile**

2.1 Liste des altitudes de la route

On obtient :

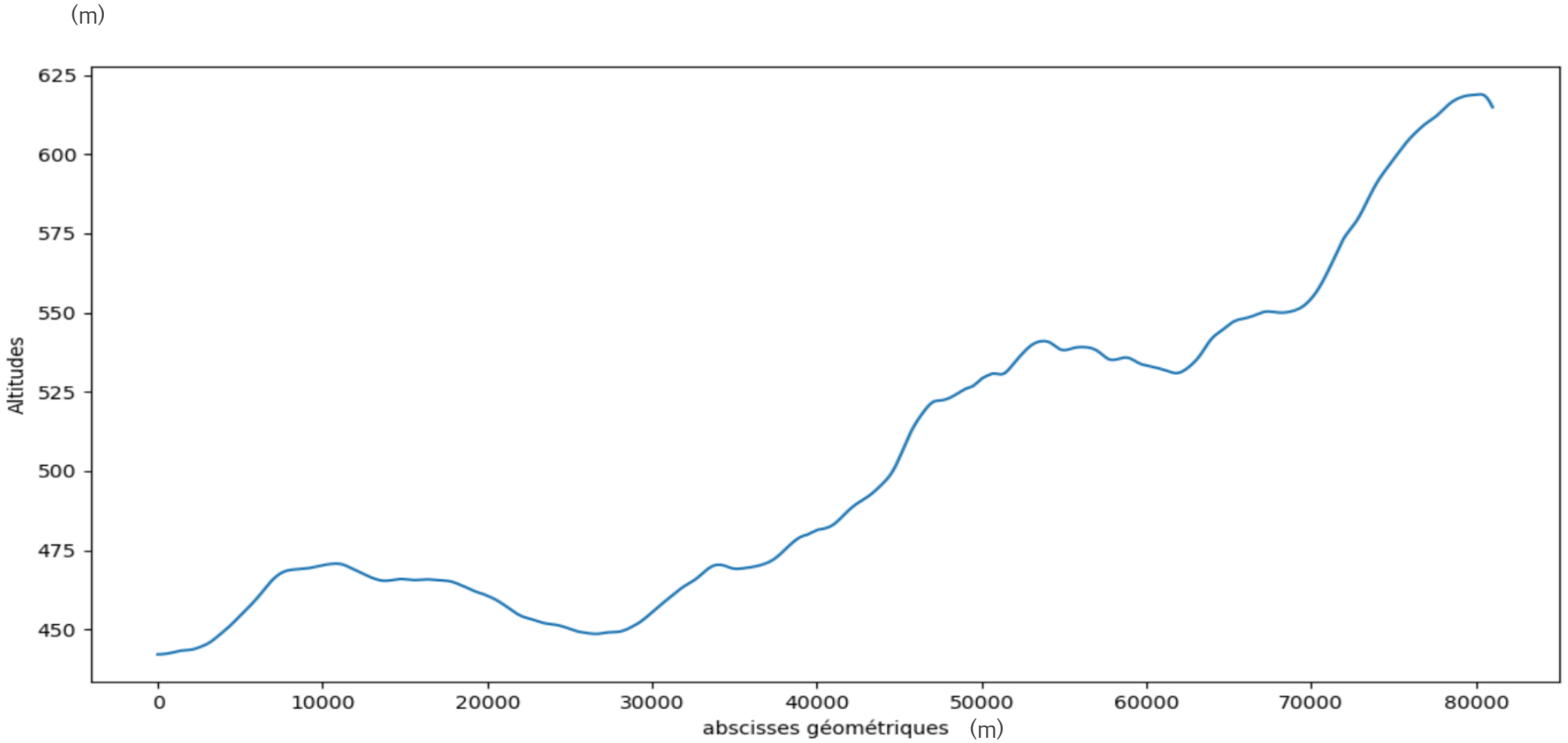


Figure 5: Altitudes lissées

Remarque : Celle-ci est plus fidèle à la réalité.

2.1 Liste des altitudes de la route - en fonction du temps -

J'impose maintenant crée une liste de vitesses: $vit[t]$

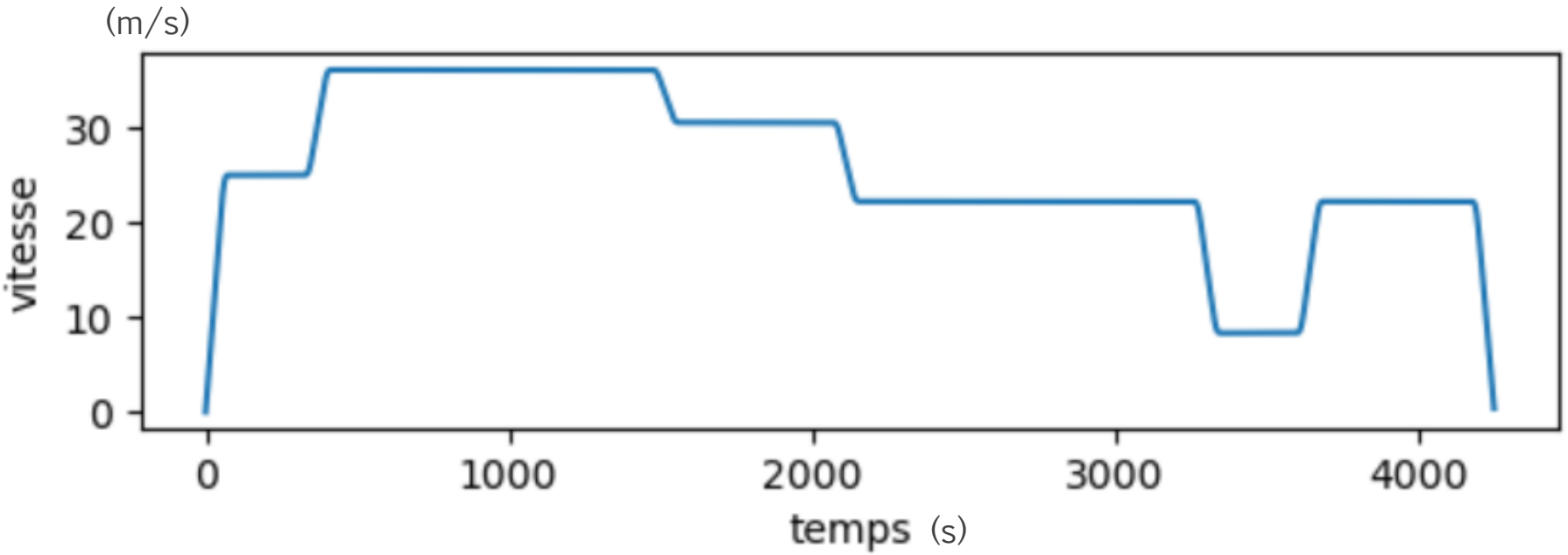


Figure 6: Courbe des vitesse en fonction du temps

2.1 Liste des altitudes de la route

Grâce à la liste des vitesses on va pouvoir **Créer 3 listes** :

1 - Distance linéique (longueur sur la courbe): d[t] += d[t-1] + (théorème de Pythagore)

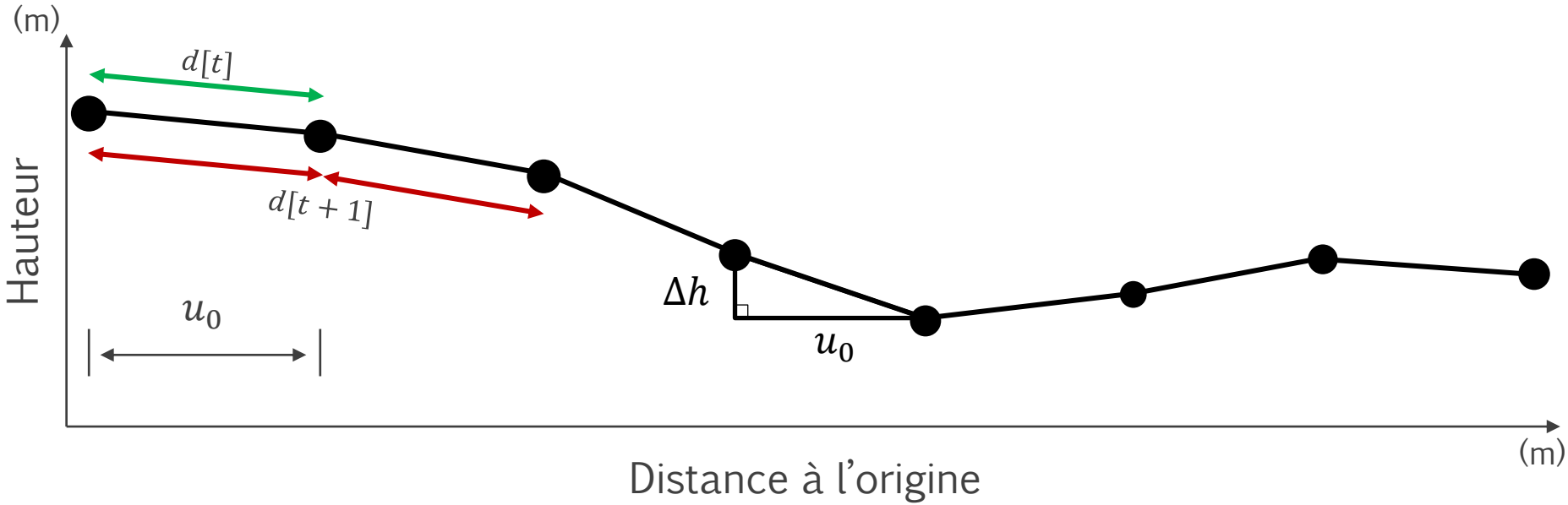


Figure 7: Calcul de la distance linéique

2.1 Liste des altitudes de la route

2- **Distance parcourue:** $L[t] = L[t-1] + 1s \times vit[t]$

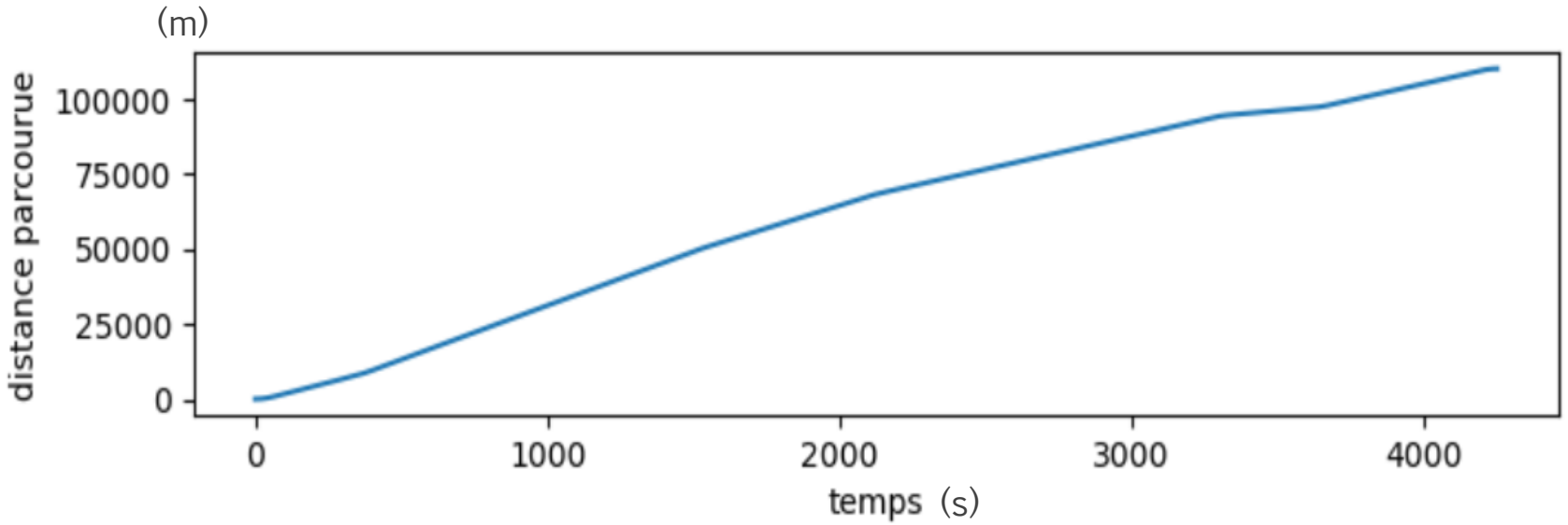


Figure 8: Courbe distance parcourue

2.1 Liste des altitudes de la route

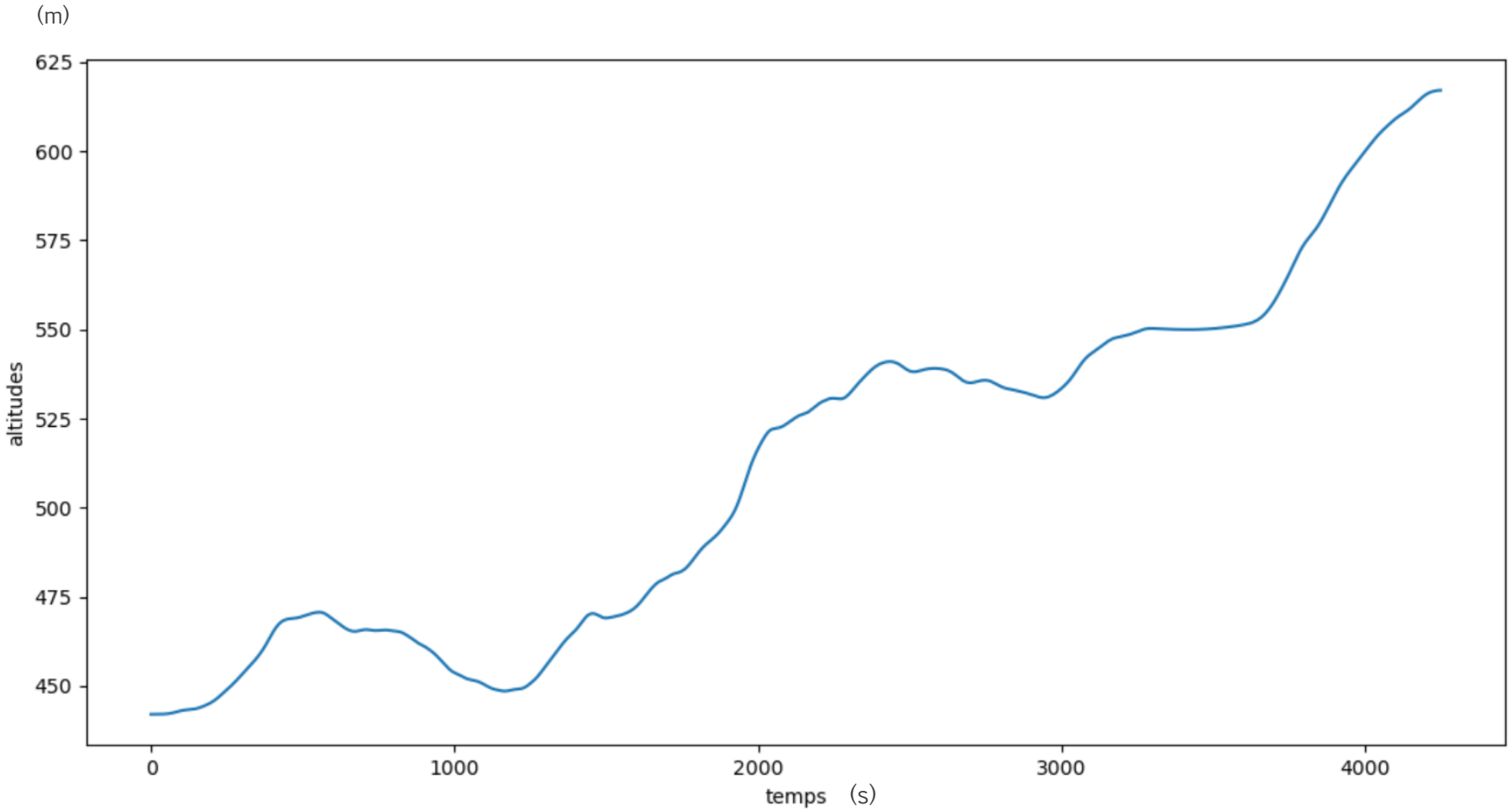


Figure 9: Altitudes en fonction du temps

2.1 Liste des altitudes de la route

Et on crée la liste des angles: **theta_au_temps_t**

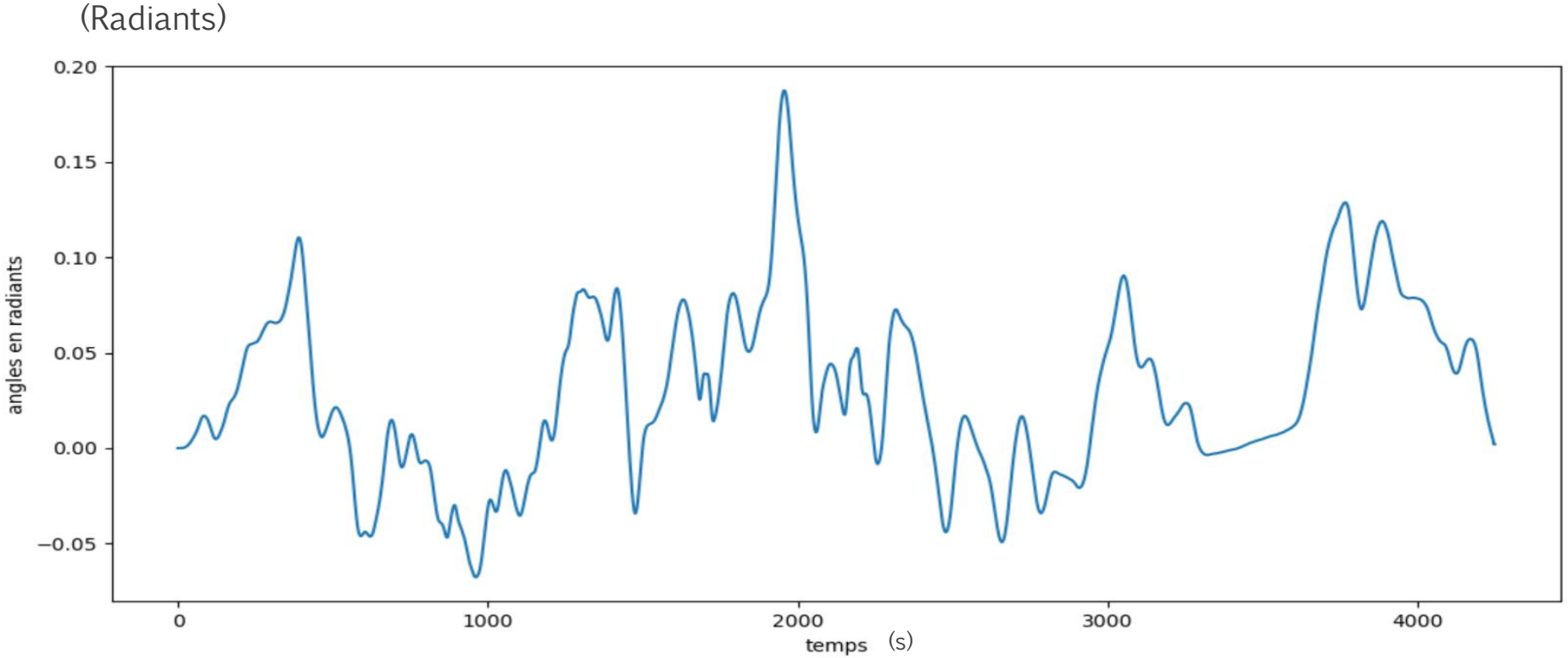
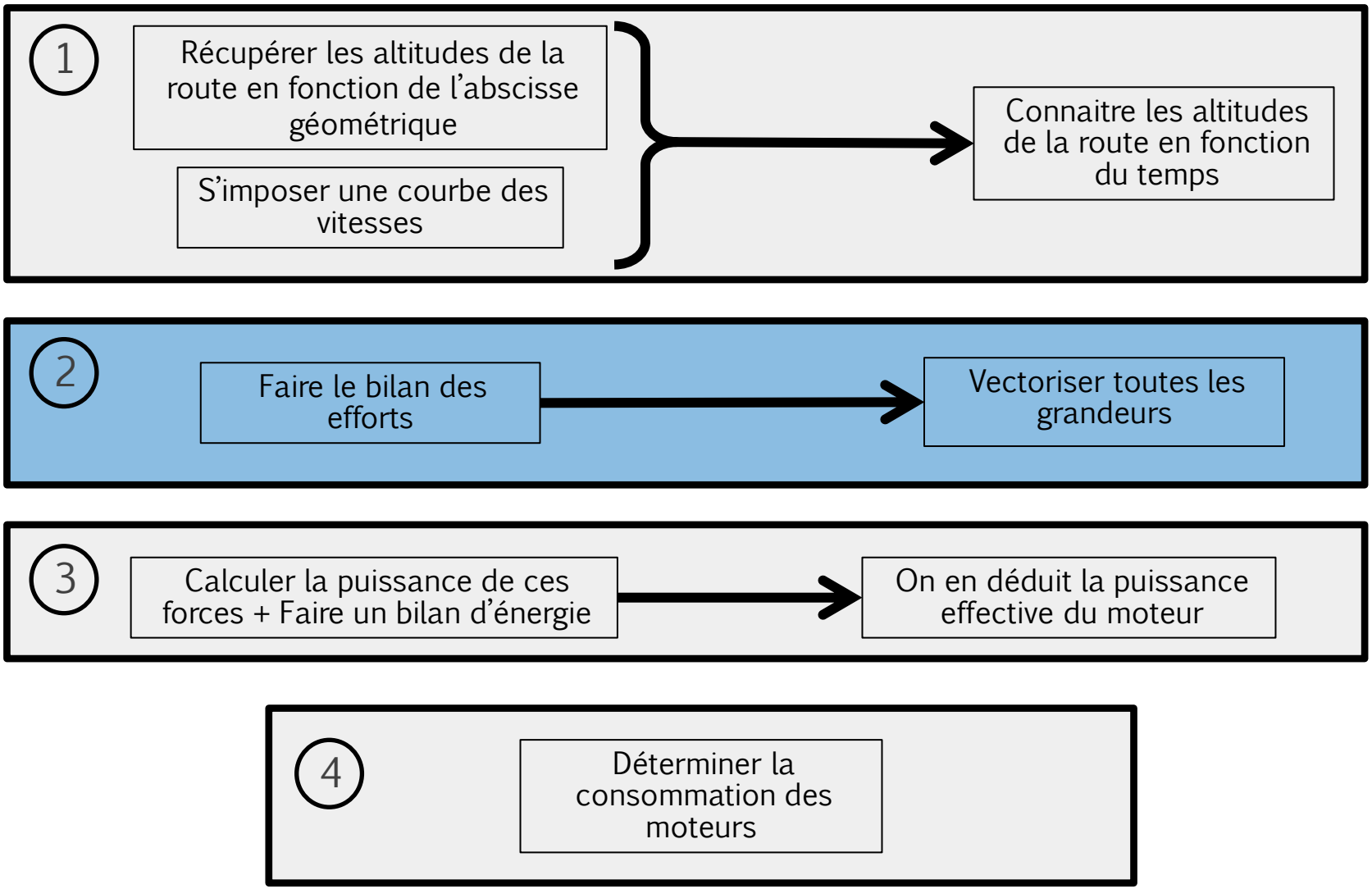


Figure 10: Courbe des angles de la route par rapport à l'horizontale au cours du temps

2.2 Le bilan des efforts



2.2 Plan de travail : Le bilan des efforts

Choix d'un modèle d'hybride:

- Peugeot 208

Figure 11: Photographie de la Peugeot 208 Modèle 1

Globalement :

Capacité du réservoir : **44 L**

Puissance maximale développée: = 100ch

Capacité de la batterie : =**11,8 kWh**

Autonomie en électrique: ≈ **60 km**

Masse à vide = **1403 kg**

Cx = 0.32

2.2 Le bilan des efforts

Système : {Véhicule} ;

Au cours du trajet la route est supposée référentiel Galiléen

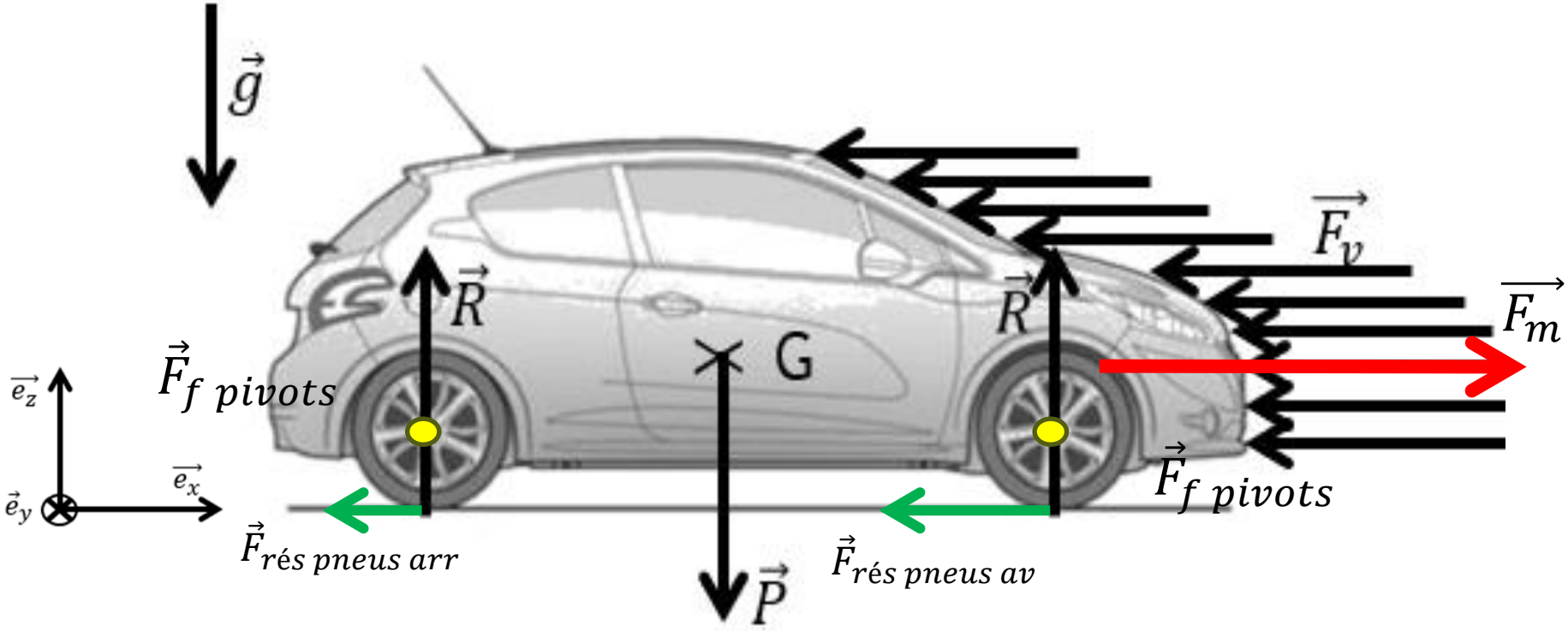


Figure 12: Schéma bilan des forces

2.2 Le bilan des efforts

Bilan des forces :

- Force moteur : $\overrightarrow{F_m}[t]$

- **Force de frottement = solide + fluide**

Cx = **coefficient de pénétration**

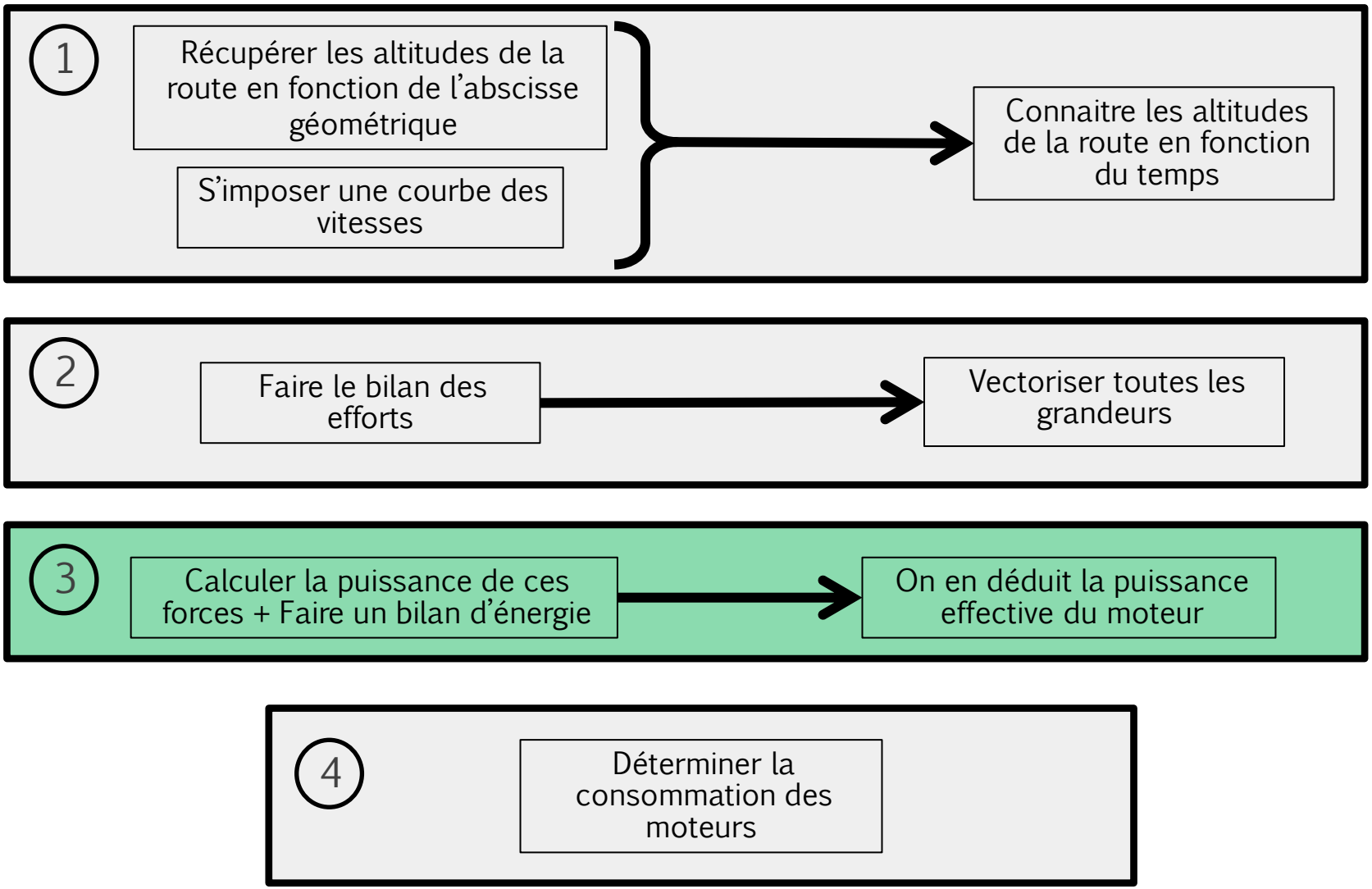
K = **force solide (cste)**

$$\overrightarrow{F_f}[t]: = \begin{cases} = -\frac{1}{2}\ C_x\ \mu\ S\ ||\overrightarrow{v[t]}||\ \overrightarrow{v[t]} & (1) \\ + (-K) * \frac{\overrightarrow{v[t]}}{||\overrightarrow{v[t]}||} & (2) \end{cases}$$

- Réaction de la route: $\overrightarrow{R}$

- Poids de la voiture : $\overrightarrow{P}[t]: = m\,\overrightarrow{g}$ (*avec* $\boldsymbol{m[t]}$ ***aproximé constant***)

2.3 Calcul des puissances



2.3 Calcul des puissances

1- $\boldsymbol{P_{F_f}}$: *La puissance de la force de frottements*

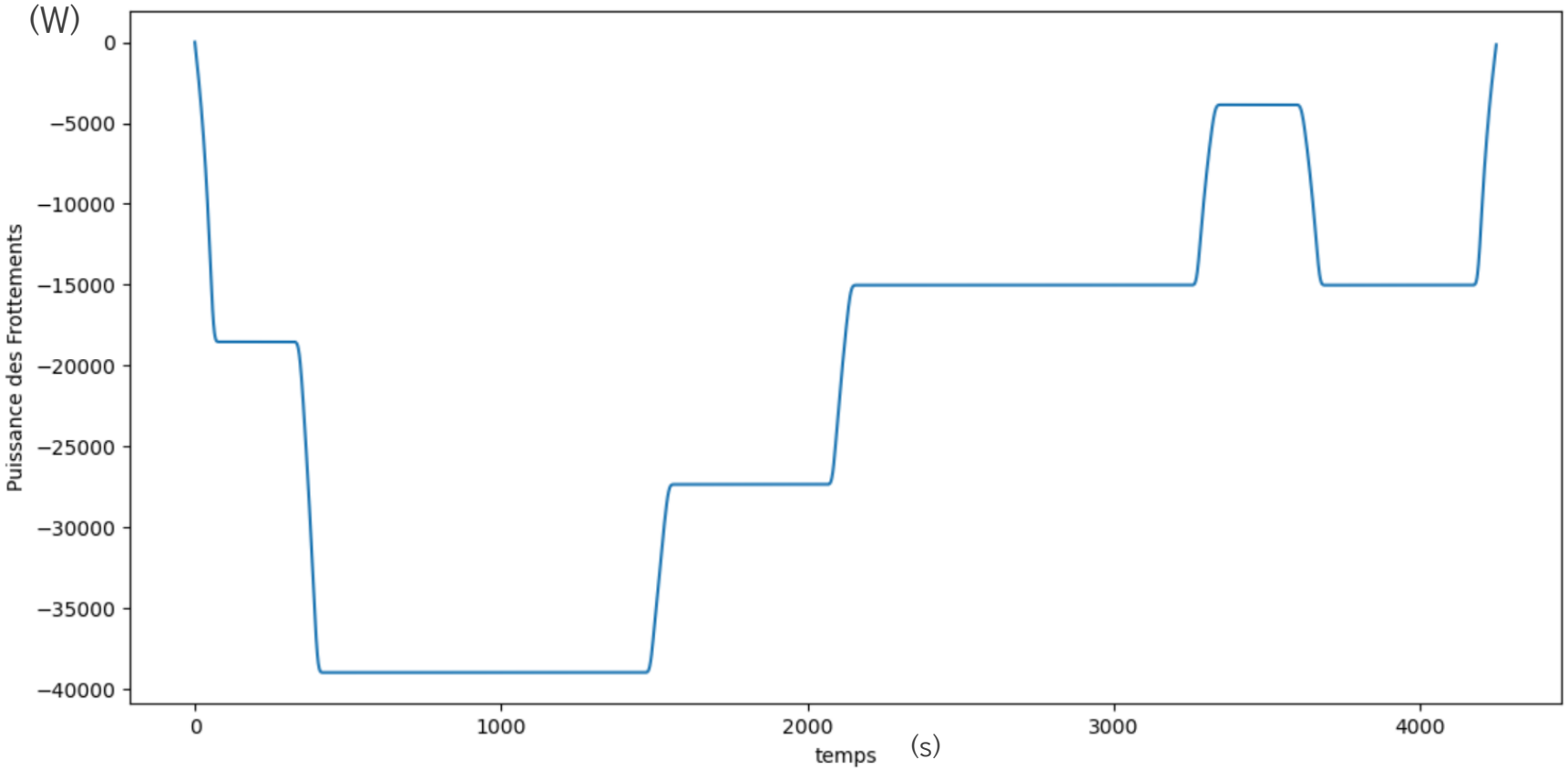


Figure 13: Courbe de la puissance des frottements

2.3 Calcul des puissances

2- $\boldsymbol{P_{Poids}}$: *La puissance du poids*

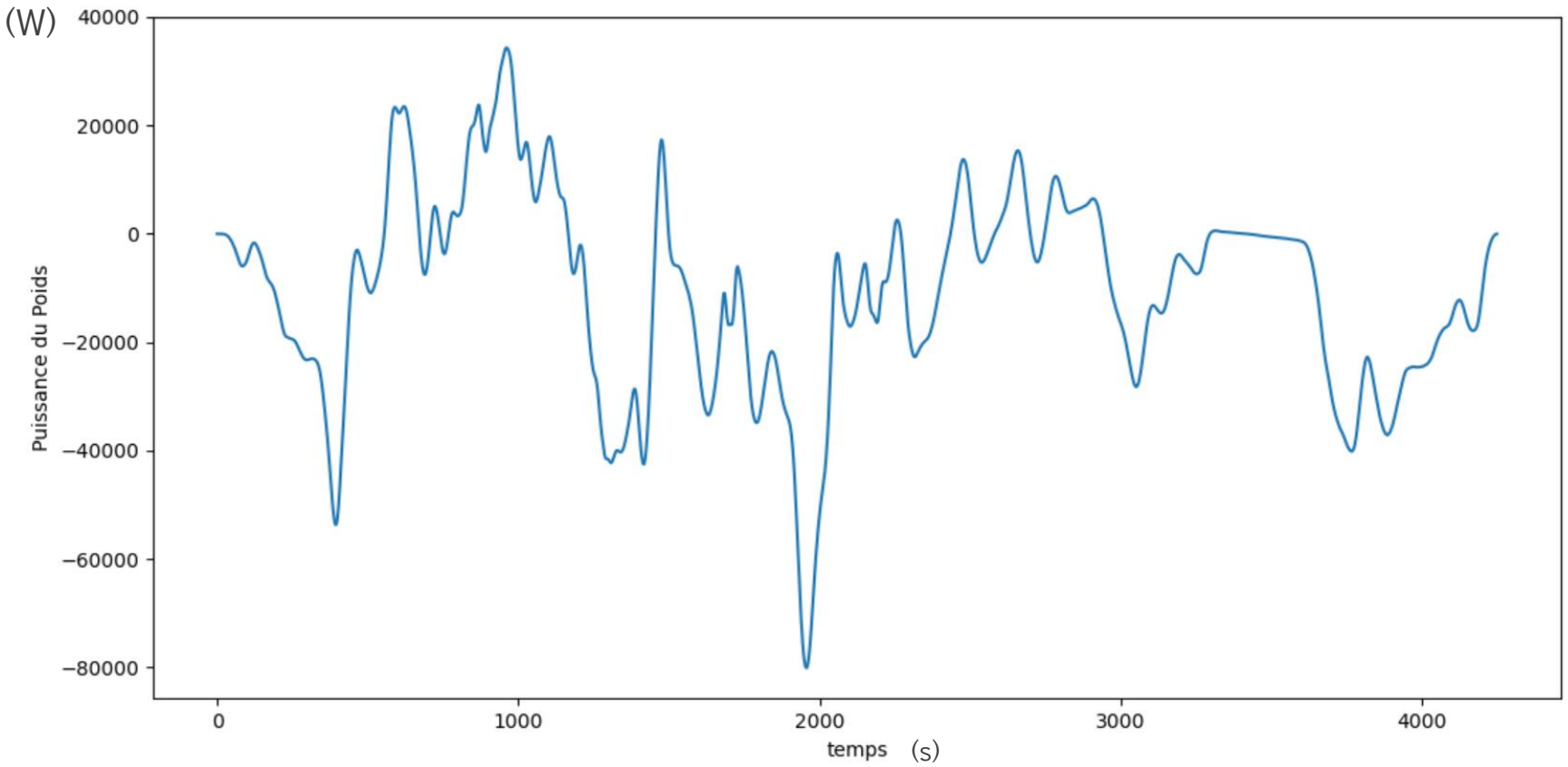


Figure 14: Courbe de la puissance du poids

2.3 Calcul des puissances

3- P_R: *La puissance de la réaction du support*

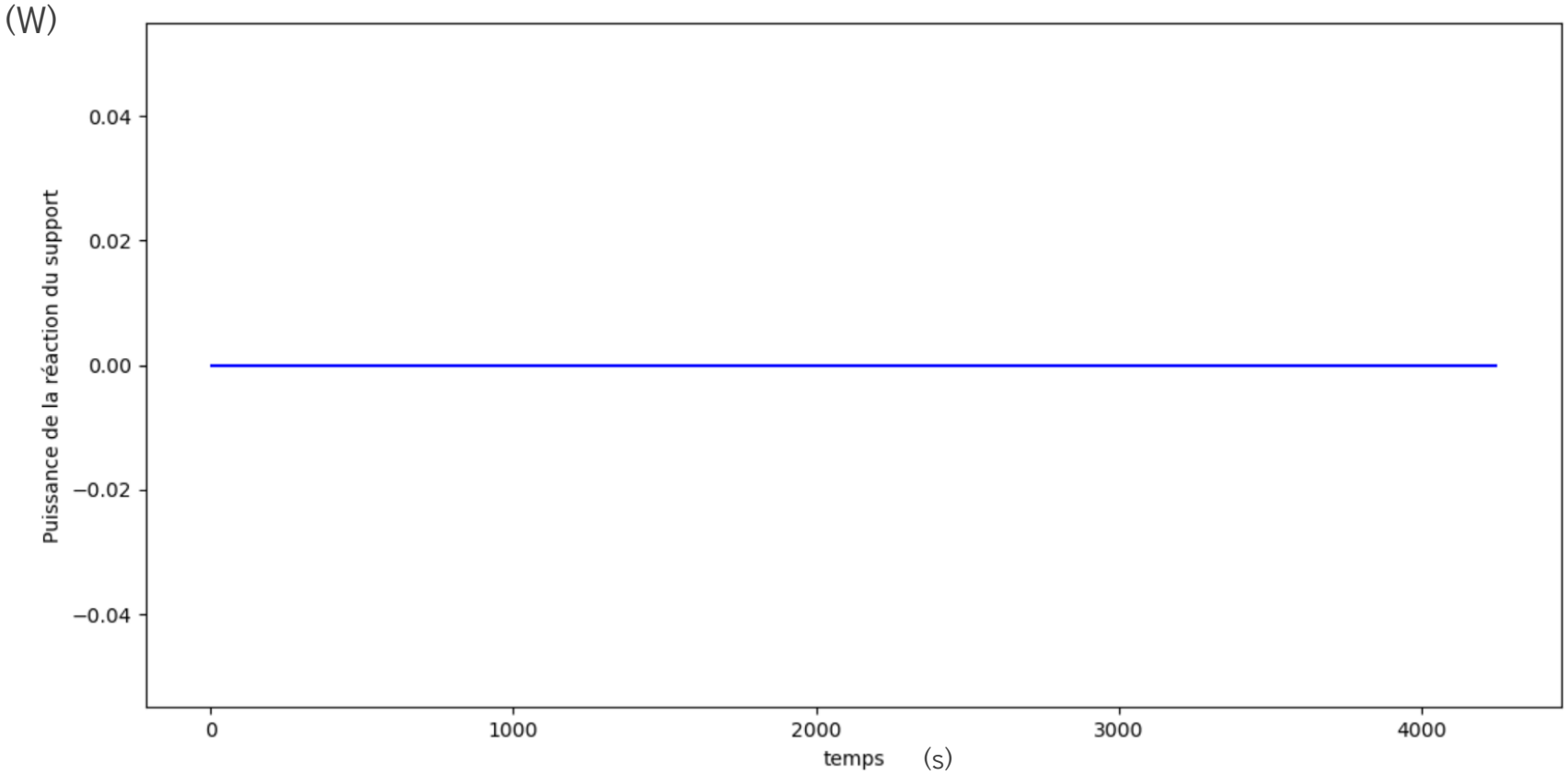


Figure 15: Courbe de la réaction du support

Application du théorème de l'énergie cinétique :

$$\frac{dE_c}{dt} = \frac{d}{dt}\left(\frac{1}{2} m \vec{v}^2[t]\right) = m \ \overrightarrow{a[t]} \,.\, \overrightarrow{v[t]} = P_m[t] + P_{Fv}[t] + P_{Poids}[t]$$

$$d'où : \quad \boxed{P_m[t] = m \ \overrightarrow{a[t]}.\overrightarrow{v[t]} - P_{Fv}[t] - P_{Poids}[t]}$$

2.3 Calcul des puissances

Ainsi : 4 - $\boldsymbol{P_m}$: *La puissance Moteur – électrique ou thermique*

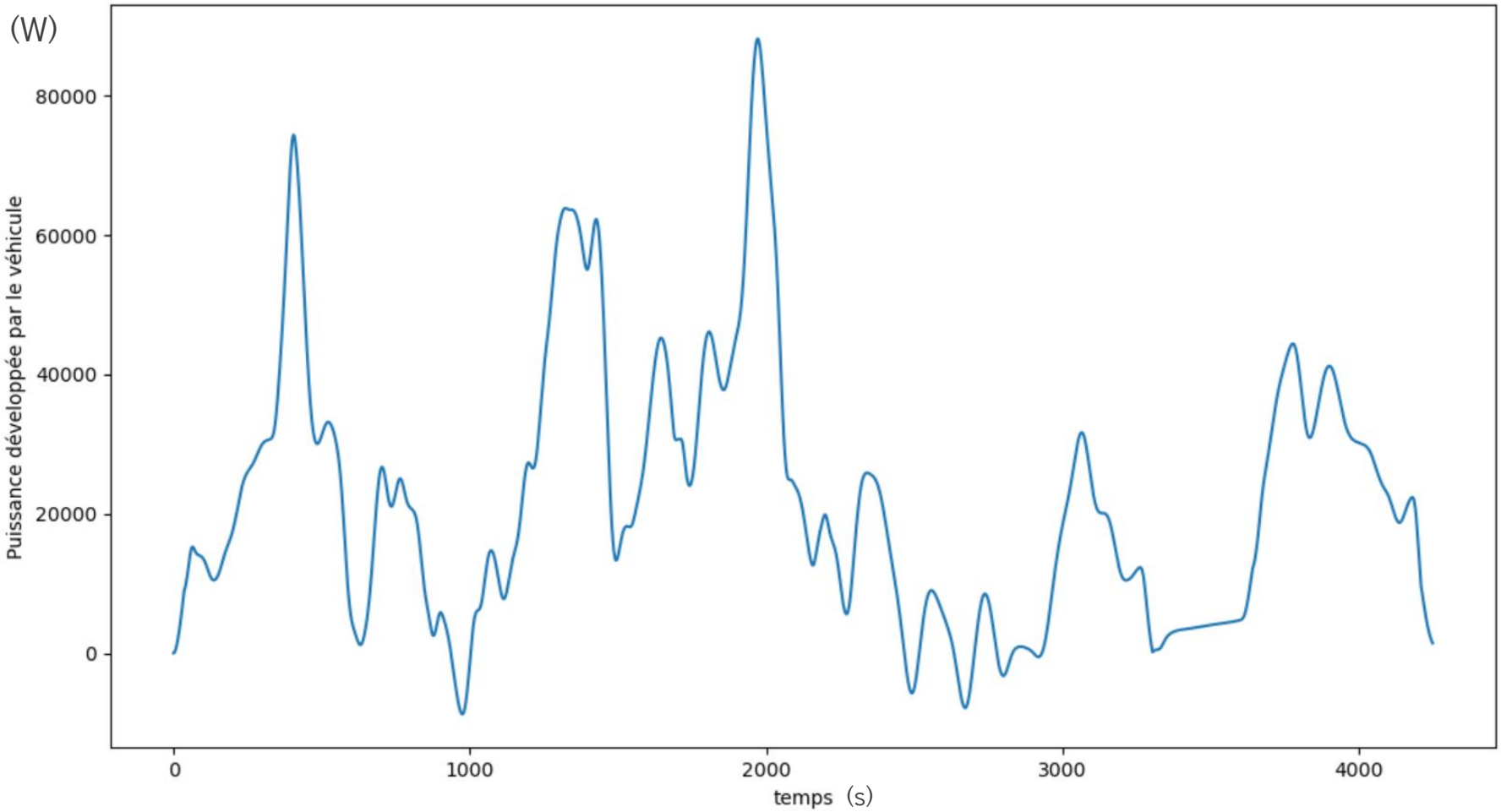
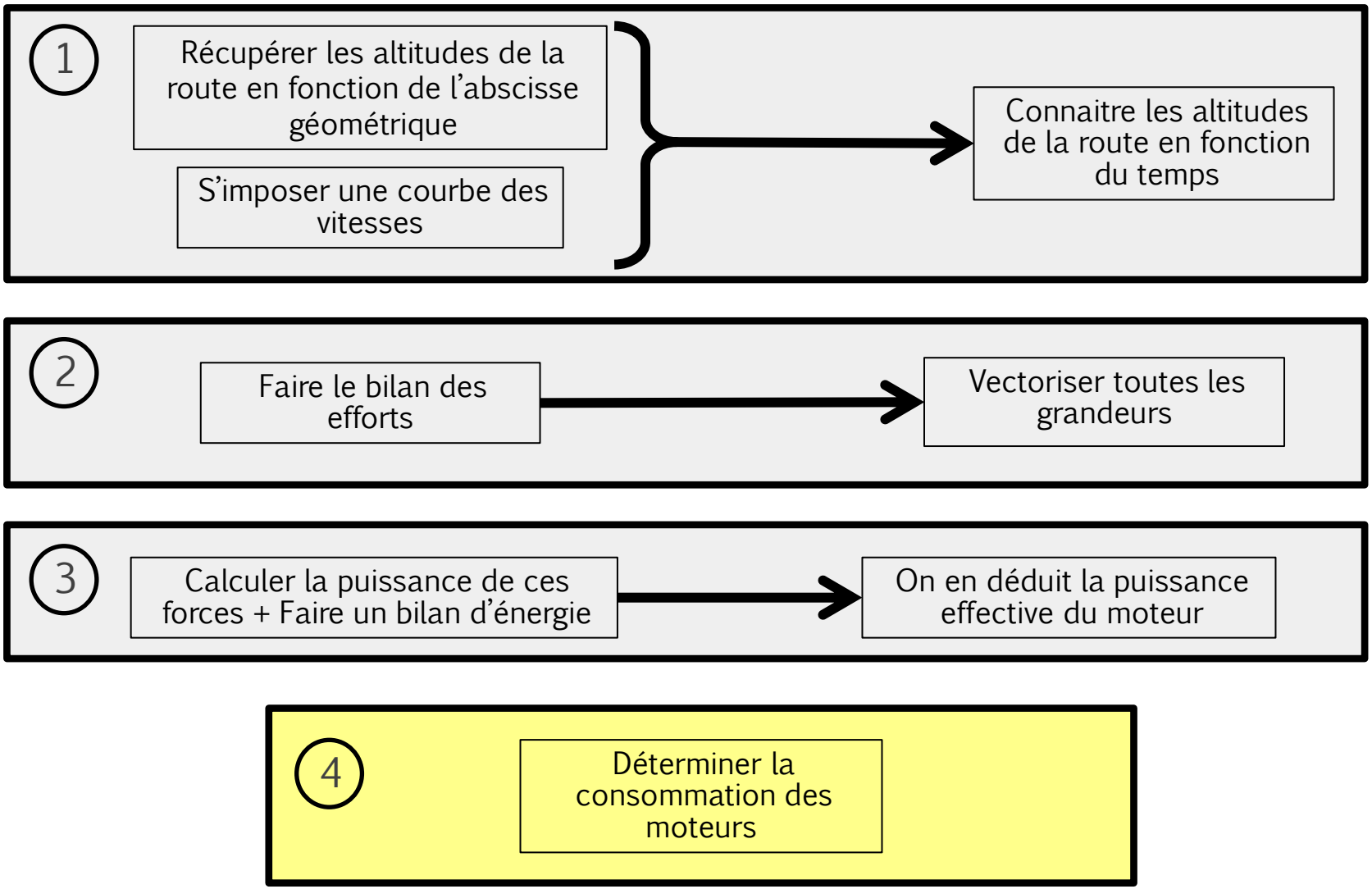


Figure 16: Courbe de la puissance développée par le véhicule

2.4 : Calcul des Consommations



2.4 : Calcul des Consommations - Consommation en essence

combustion du SP 95 = Combustion de l'octane

masse volumique $\boldsymbol{\rho = 0,72\ kg.L^{-1}}$

Rendement énergétique: $\boldsymbol{\eta_{tot} = 28\%}$

$\boldsymbol{M(octane) = 114.10^{-3} kg \cdot mol^{-1}}$

Développant une Puissance : $\boldsymbol{P_m[t]}$

$$C_8H_{18}(g) + \frac{25}{2}O_2(g) \xrightarrow{combustion} 8CO_2(g) + 9H_2O(l) \quad ; \quad \Delta_{comb}H^0 = -5507\ kJ.mol^{-1}$$

$$Vol\,[t] = \frac{M(octane) \times P_m[t]}{\eta_{tot} \times \Delta_{comb}H^0 \times \rho}$$

$(en\ Litres\ par\ seconde)$

2.4 : Calcul des Consommations - Consommation en mode électrique

Rendement énergétique: $\boldsymbol{\eta_{elec}} = \mathbf{28}\%$

Capacité de la batterie : $\mathrm{C}_{\mathrm{capacité\_bat}} = 11{,}8\ kWh$

Puissance moteur : $\boldsymbol{P_m[t]}$

$$P_B = \frac{P_m[t]}{\mathrm{C}_{\mathrm{capacité\_bat}} \times \eta_{elec} \times 3600 \times 1000} \times 100\ (\%)$$

(en % /sec)

Remarque : **le constructeur annonce que l'autonomie en mode électrique est de 60 km.**

Hypothèse simplificatrice: La batterie ne présente pas de limite de vitesse de recharge

3. Modèle de consommation final :

Route exemple :

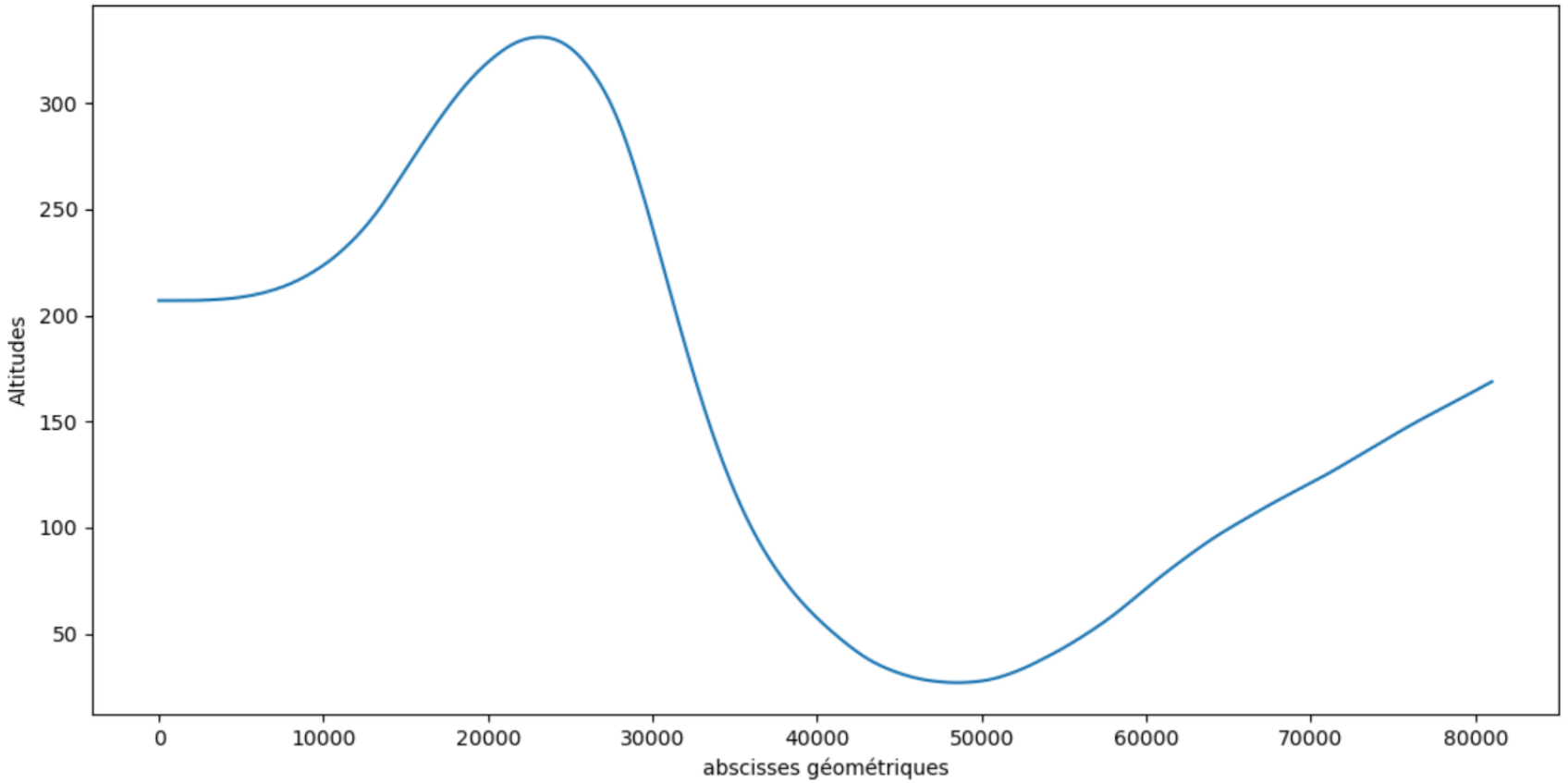


Figure 17: Courbe fictive de route (test)

3.1 Modèle de consommation final : Sur un trajet quelconque:

Résultat pour:
bat[0] = 100 %
Res[0] = 44.0 L

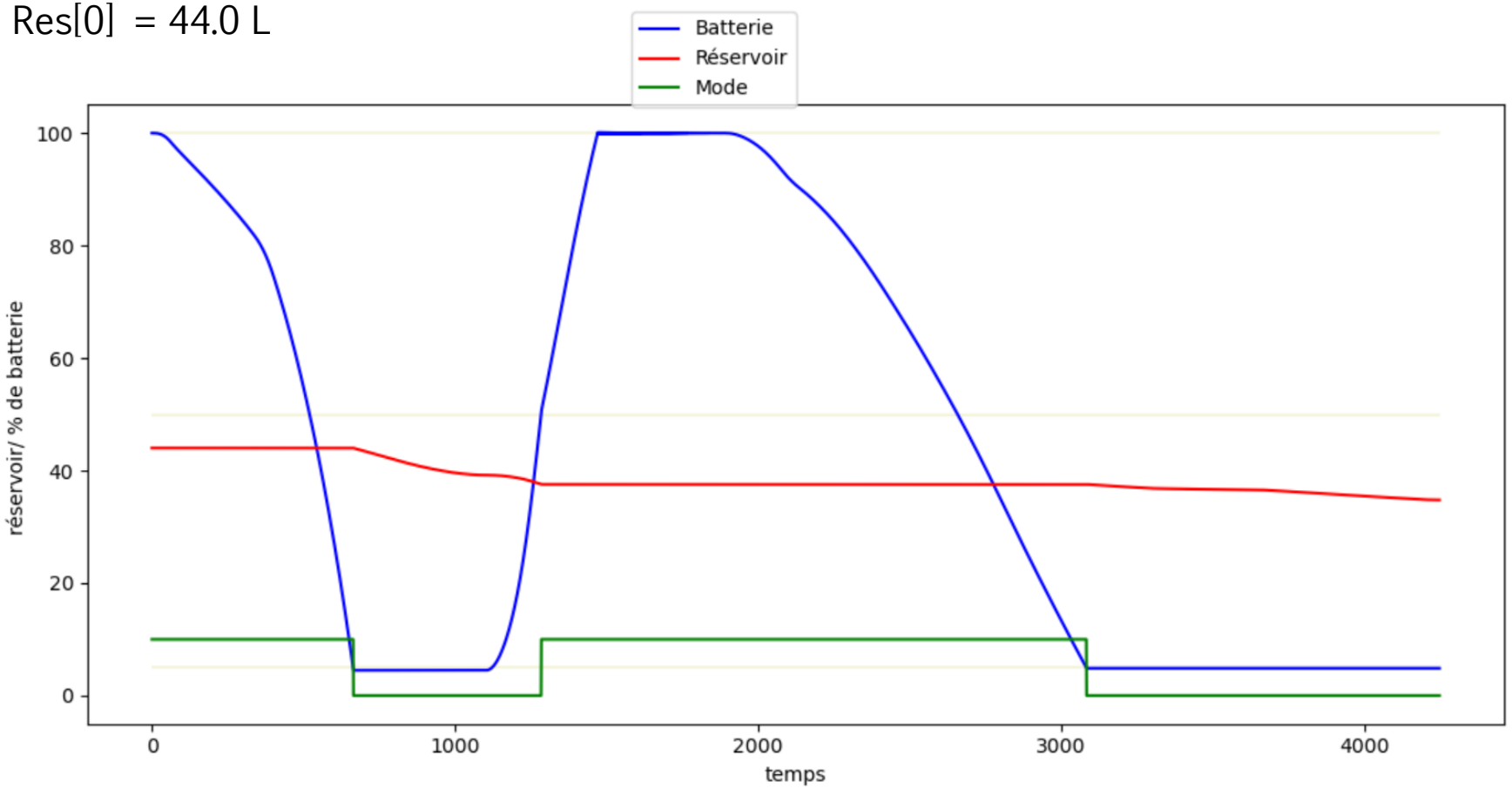


Figure 18: Résultats et explication de la boucle primaire sur la courbe fictive de route (test)

3.2 Retour au problème de départ :

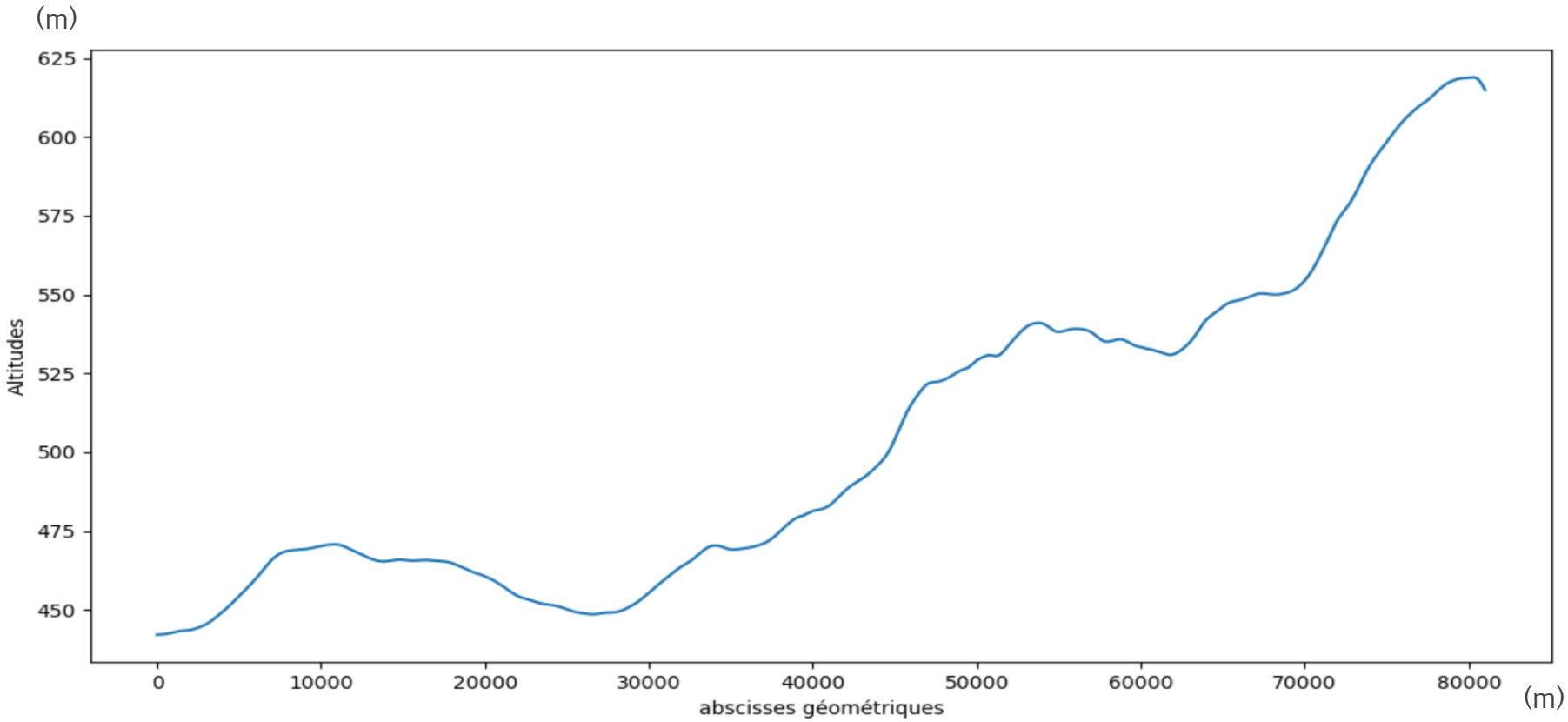


Figure 5: Courbe de la route – Altitudes en fonction de l'abscice géométrique

3.2 Retour au problème de départ :

Résultat pour:
Res[0] = 44.0 L

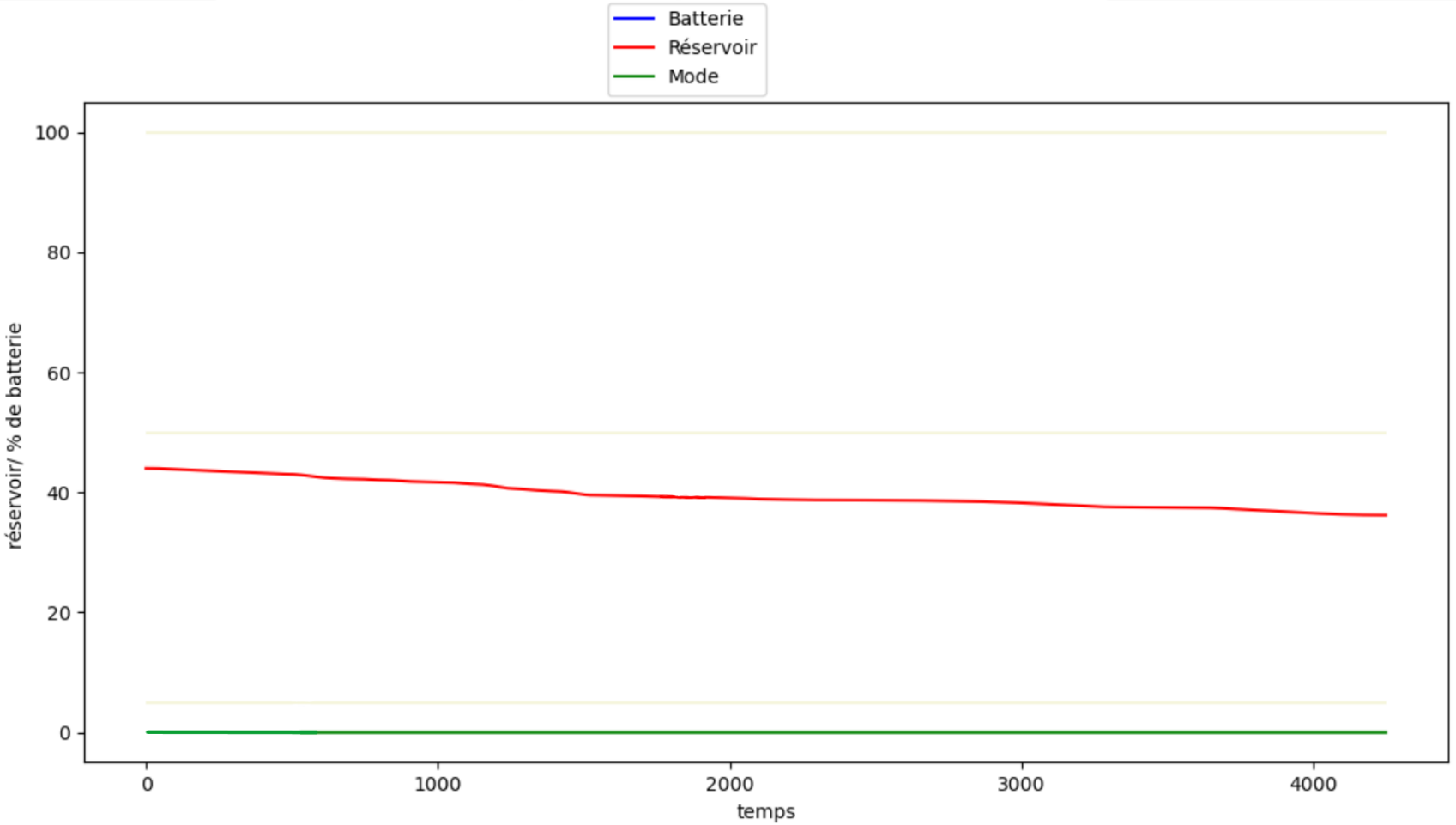


Figure 22: Résultats en termes de consommation pour un véhicule thermique uniquement, sur la route réelle

3.2 Retour au problème de départ :

Rappel 1 : Prix du kWh domestique : = 0,163 €/kWh
Rappel 2 : Prix du litre de SP 95 : = 1.549 €/L

Résultat pour:
bat[0] = 5 %
Res[0] = 44.0 L

<u>Coût du trajet :</u>

Consommation essence :

$$44{,}0\ L\ - 33{,}0\ L = 11{,}0\ L \Rightarrow co\hat{u}t : C = 17{,}0\ €$$

$$C_{tot} \cong 17{,}0\ €$$

3.2 Retour au problème de départ :

Résultat pour:
bat[0] = 100 %
Res[0] = 44.0 L

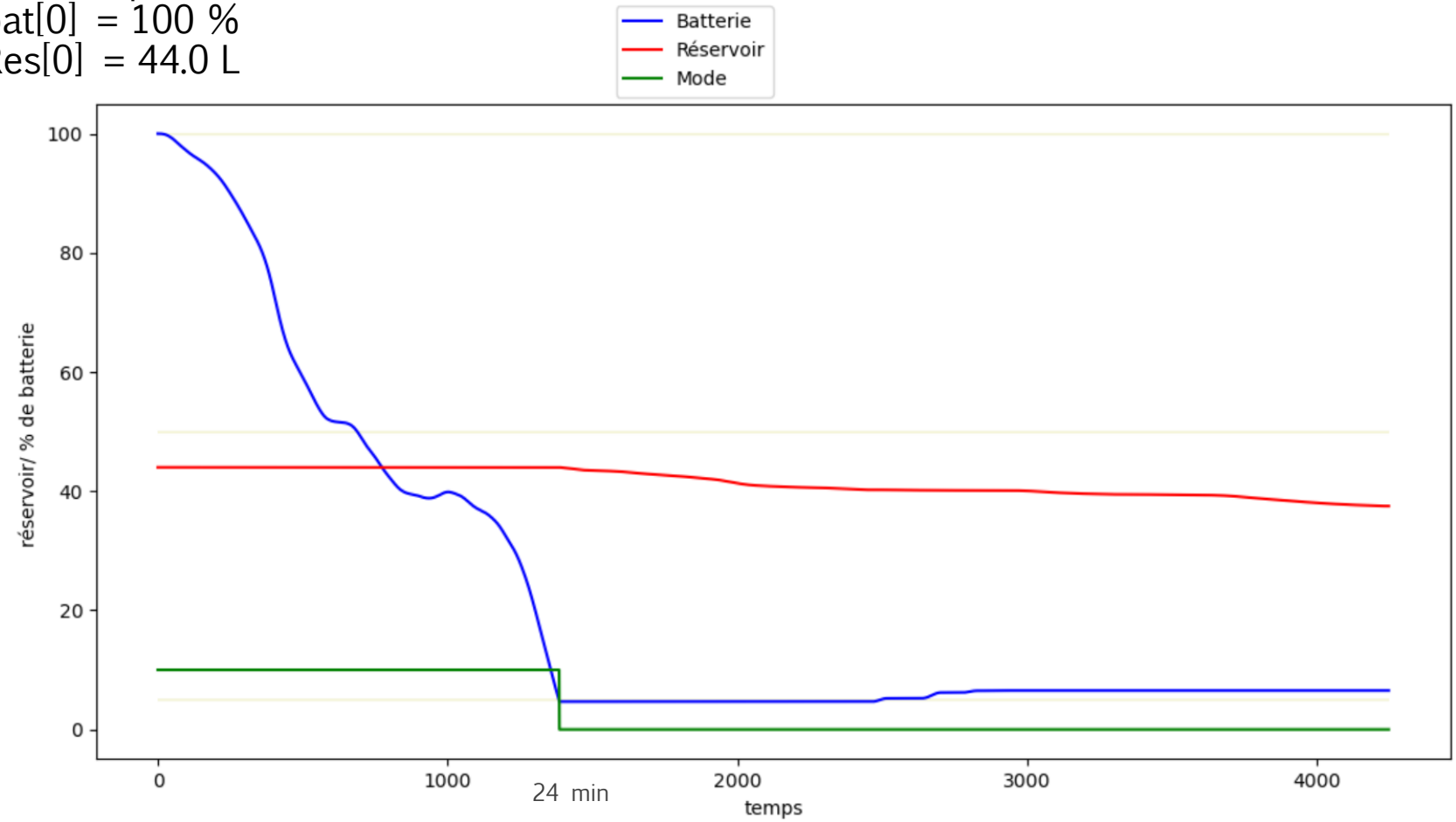


Figure 20: Résultats en termes de consommation sur la route réelle cas n°1 batterie à 100%

3.2 Retour au problème de départ :

Rappel 1 : Prix du kWh domestique : = 0,163 €/kWh
Rappel 2 : Prix du litre de SP 95 : = 1.549 €/L

Résultat pour:
bat[0] = 100 %
Res[0] = 44.0 L

<u>Coût du trajet :</u>

Consommation électrique :

$11{,}8\ kW$ *pour la charge complète de la batterie* $\Rightarrow$ *coût*: $C_1 = 1{,}92€$

Consommation essence :

$$44{,}0\ L\ - 37{,}1\ L = 6.9\ L \Rightarrow \text{coût}: C_2 = 10.7\ €$$

$$C_{tot} = C_1 + C_2\ \cong 12.6\ €$$

Comparaison entre essence pure et Hybride : *économie de presque* 5€

3.2 Retour au problème de départ :

Que se passe-t-il sur le chemin retour

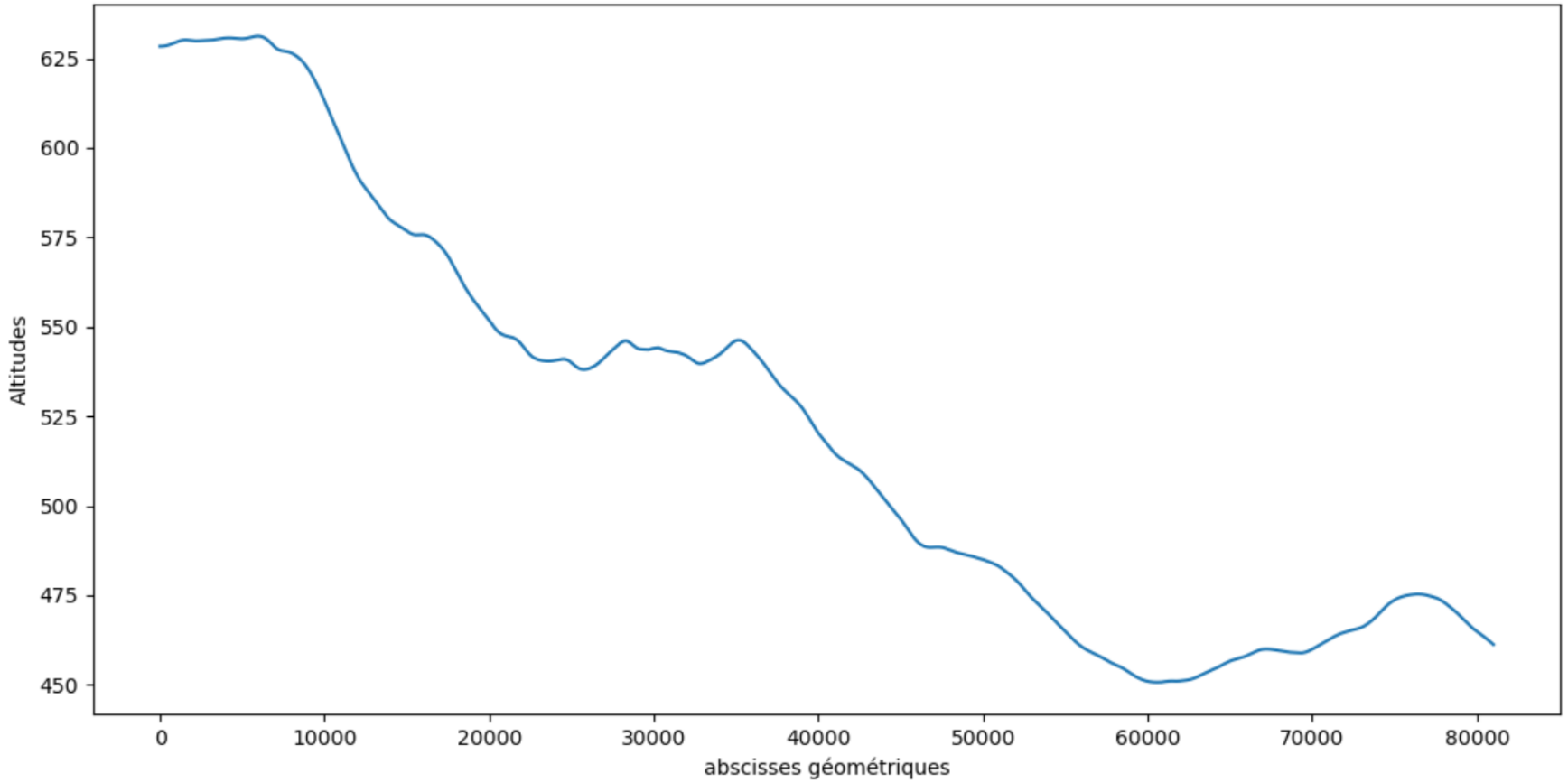


Figure 21: Route réelle – Trajet Retour

3.2 Retour au problème de départ :

Résultat pour:
Res[0] = 44.0 L

Route retour :

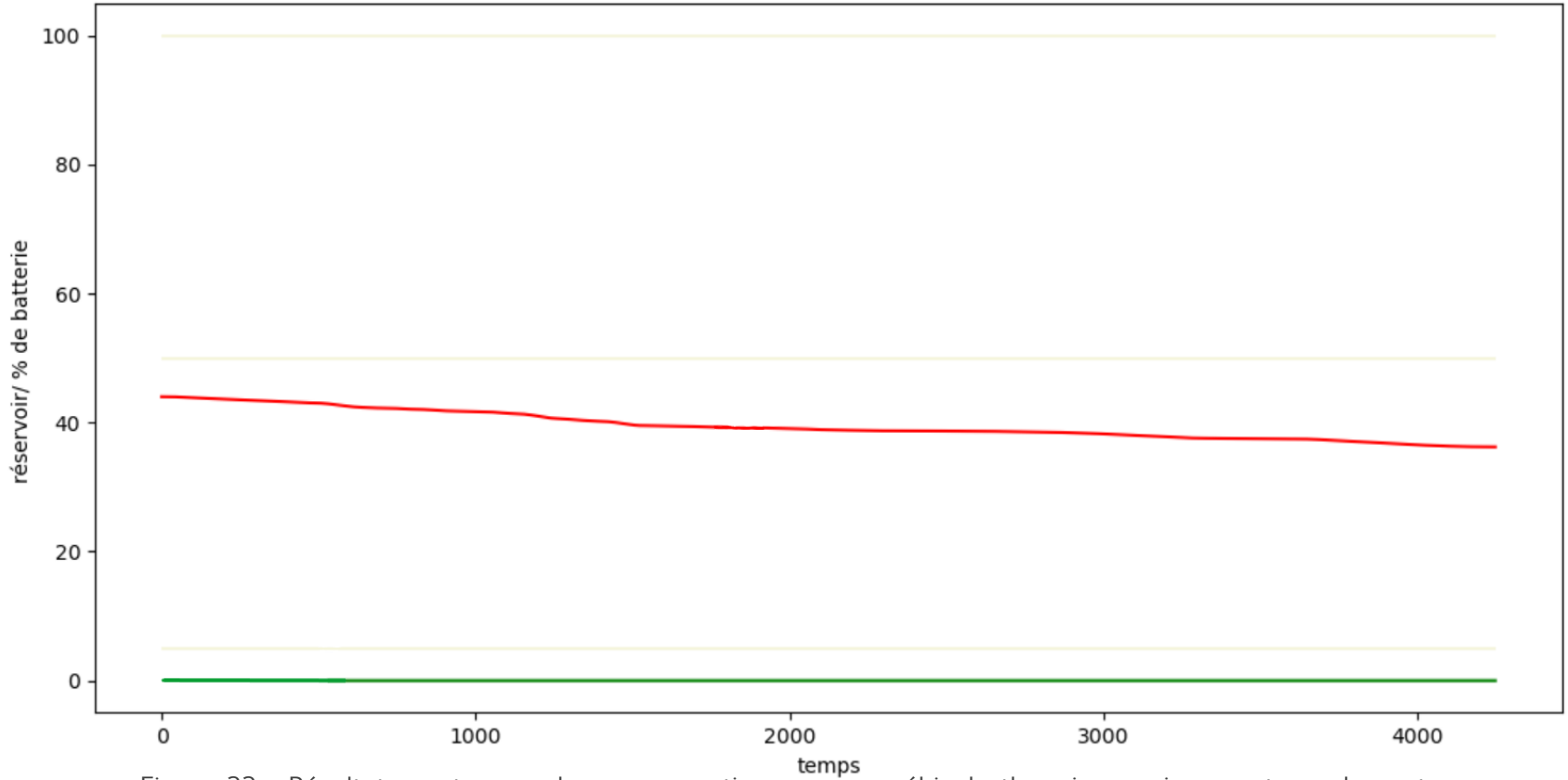


Figure 22: Résultats en termes de consommation pour un véhicule thermique uniquement, sur la route réelle

3.2 Retour au problème de départ :

Résultat pour:
bat[0] = 100 %
Res[0] = 44.0 L

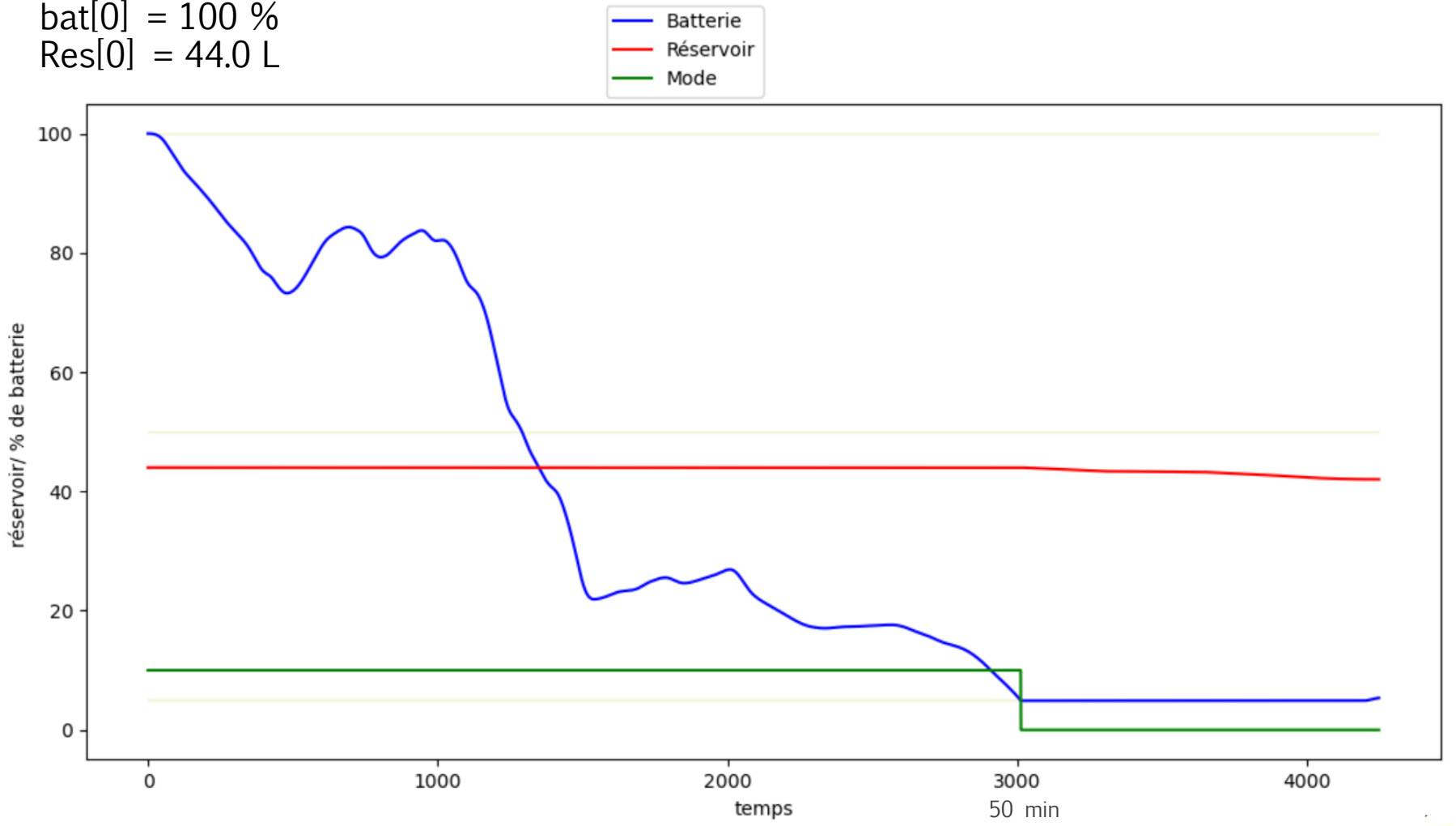


Figure 23: Résultats en termes de consommation pour un véhicule hybride, sur la route retour

3.2 Retour au problème de départ :

Rappel 1 : Prix du kWh domestique : = 0,163 €/kWh
Rappel 2 : Prix du litre de SP 95 : = 1.549 €/L

Résultat pour:
Res[0] = 44.0 L

Coût du trajet :

Consommation essence :
$44{,}0\ L - 36{,}0\ L = 8{,}0\ L$

$\Rightarrow co\hat{u}t:\ C = 12{,}4\ €$

Résultat pour:
Bat[0] =100 %
Res[0] = 44.0 L

Coût du trajet :

Consommation électrique :
$11{,}8\ kW$ *pour la charge complète de la batterie*
$\Rightarrow co\hat{u}t:\ C_1 = 1{,}92€$

Consommation essence :
$44{,}0\ L - 42{,}0\ L = 2{,}0\ L \Rightarrow co\hat{u}t: C_2 = 3{,}1\ €$

$C_{tot} = C_1 + C_2 \cong 5{,}2\ €$

Comparaison entre essence pure et Hybride : *économie de presque* 7,2€

CONCLUSION :

- Economies prédictibles de consommation;
- Limites de la simulation :
  - Hypothèses simplificatrices ;
  - Le moteur thermique ne recharge pas directement la batterie ;
- Ouverture - Possibilité d'application :
  - Déterminer un optimum d'alternance électrique et thermique consommation
  - Un retour 'temps réel' au conducteur.

FIN

Estimation de la valeur de K :

K est une constante homogène à une force (en Newton)

K correspondant à la valeur qu'aurait la force de frottements (si cette force était modélisée comme une force solide constante.)
regroupant :

- Les frottements dû à la déformation des pneus
- Les frottements internes à la liaison pivot (roues-essieux)

Celle-ci est non négligeable à faible vitesse donc sera considérée majoritaire

pour : $||\overrightarrow{v[t]}|| \leq 55\ km.h^{-1}$

$$v(t) = \frac{0 - v_0}{tf - 0}t + v_0$$

$$TEC : \quad -\frac{1}{2}mv_0^2 = W$$

$$et : \mathrm{W} = \int_0^f \vec{F}\vec{v}[t]\,\mathrm{d}t = \int_0^f -K\left(\frac{-v_0 t}{\Delta t} + v_0\right)dt$$

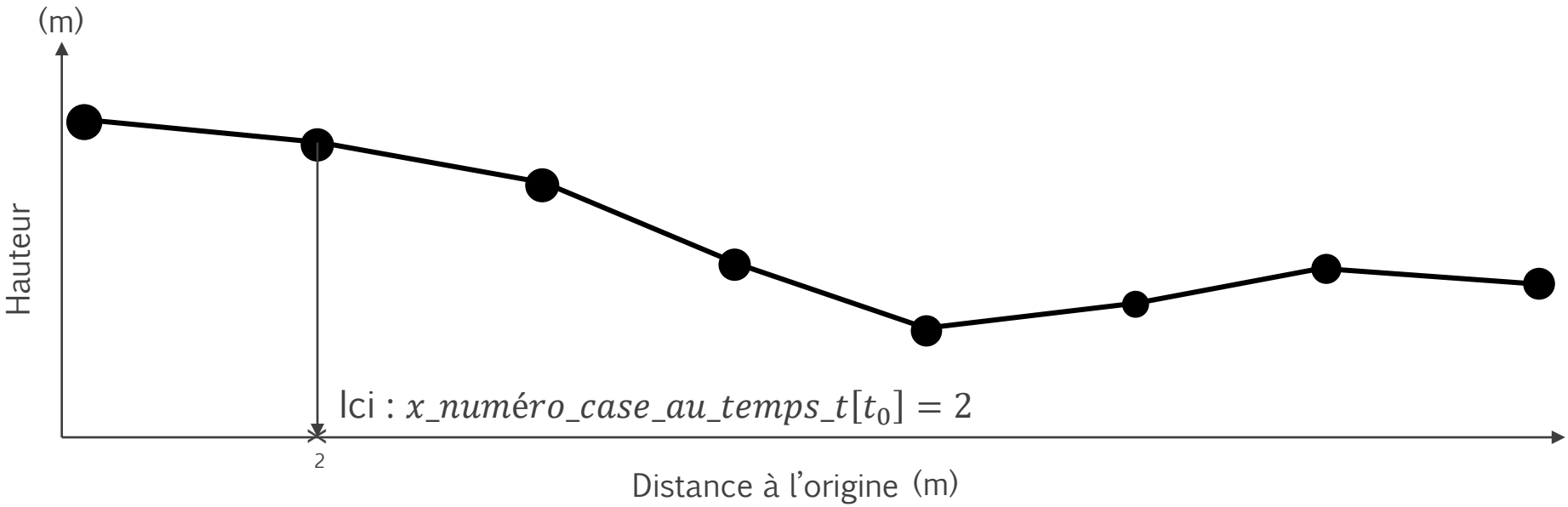
$$\mathrm{W} = \int_0^f \frac{K}{\Delta t}v_0 t\,\mathrm{d}t - \int_0^f v_0 \mathrm{K}\,\mathrm{d}t = -\frac{1}{2}Kv_0\Delta t$$

$$\begin{array}{c} \Delta t \approx 30s\ ; \\ \& \quad m \approx 1535\ kg\ ; \\ v_0 \approx 30\ km.h^{-1} \end{array} \quad \Rightarrow \quad \mathrm{K} = \frac{mv_0}{\Delta t} \quad \Rightarrow \quad \boxed{\mathrm{K} \approx 430N}$$

Liste des altitudes de la route : Pour aller plus loin – x_numéro_case_au_temps_t

Enfin on crée la liste: **x_numéro_case_au_temps_t**

Celle-ci permet d'identifier l'abscisses de l'ordonnée (altitude) atteinte à l'instant t ;



La liste **x_numéro_case_au_temps_t** (taille : 4200 valeurs) contenant les **abscisses géométriques** au temps t

Evaluation instantanée de la consommation : Consommation en essence

Vérifions :

$$V_{moy} = \frac{114.10^{-3} \times 20000\ W}{0.28\ \times -5507.10^{3}\ \times 0.72} \cong 0.00205\ L.s^{-1}$$

$$V_{moyen}(L/100km) = \frac{V_{moy}}{vit_{moyenne}}$$

En moyenne: ici j'obtiens que mon véhicule consomme:
$\boldsymbol{V_{moyen}} \cong \mathbf{7\ L/100km}$; $vit_{moyenne} \cong 100\ km/h$

Et je sais que Mon Véhicule consomme en moyenne:
$\boldsymbol{V = 5,4\ L/100km}$

L'ordre de grandeur est donc respecté. ✓

2.4 : Calcul des Consommations - Consommation en essence

Vérifications :

En moyenne:
Pour $vit_{moyenne} \cong 100\ km/h$

$$V_{moyen}(L/100km) \cong 7\ L/100km$$

Ordre de grandeur vérifié:
Mon Véhicule consomme en moyenne: $\boldsymbol{V = 5,4\ L/100km}$

Vérifications :

$En\ moyenne$:
$Pour\ vit_{moyenne} \cong\ 100\ km/h$

$$\boldsymbol{P_B \cong\ 0.058\ \%\ (de\ \text{charge/sec})}$$

La distance maximale est ainsi: $\mathbf{D_{max} \cong 50\ km}$ (sans compter les possibles regains d'énergie)

$\mathbf{D_{max\_annoncée} = 60\ km}$ (donc vérifié)

Vue d'ensemble sur les véhicules hybrides sur le marché :

- **L'Hybride Non rechargeable :**
  - Une poignée de kilomètre d'autonomie et ne se recharge qu'avec une conduite de ville (stop & go nombreux)

- **L'hybrides rechargeable :**
  - Autonomie améliorée jusqu'à près de 100 km en 100 % électrique pour les meilleures

4. Programmes – Importations et variables

```
## Variable modifiables

init_batterie= 100. #pourcents
init_reservoire= 44. # litres
volume_réservoir_total =44 # Litres
init_mode =1


## Variables strictes au programme

N=4250 #secondes 1H11min
g = 9.81 #m.s-2
taille_course = 81000 #mètres projeté sur x
Masse_ensemble= 1403. #kg, à vide #réservoir vide la 208 hybride n'existe pas donc produit en croix : 1720 kg (508 Hybride) ; 1420 kg (508 thermique) ; 1158 kg (208 thermique)
Rayon_roue = 0.191 #m (=15")
n_global= 28/100 #thermique
Cx = 0.32 #constructeur
rho_air = 1.225 #kg/m^3 à 15°C et Pext = 1 atm
rho_essence = 0.702 #kg/L
Rayon_modélisation_sphérique = 0.799 #mètre,
viscosity_of_air= 1.81e-5 #kg/(ms) à 15°c
hauteur_véhicule= 1.460
largeur_véhicule= 1.739
ArH_essence = -5507*10**3 #J.mol-1.
n_electrique = 80/100 #en moyenne car la valeur varie pour un moteur à courant alternatif
capacité_batterie = 11.8 #kWh
M = 114.2*10**-3 #masse molaire de l'octane
```

4. Programmes:

```
## Variable induites


SEUIL_DE_BASCULEMENT_THERM_VERS_BATTERIE=50. #seuil Variable
SEUIL_DE_BASCULEMENT_BATTERIE_VERS_THERM = 5. #min Batterie seuil Variable
SEUIL_bat_max =100. #seuil Fixe


# On pose pour le moment la liste des temps de 1 à N et donc de 'pas'=1 seconde

bat=np.zeros((N))
bat[0]=init_batterie
res=np.zeros((N))
res[0]=init_reservoire
mode=np.zeros((N))
mode[0]= init_mode   # électrique 10 essence 0
```

4. Programmes:

```
### I- LA COURBE DE ROUTE

image = plt.imread("file:///C:/Users/hecqu/OneDrive/Bureau/Route étudiée elevation profile utilisé.PNG")

def seuil(x,h):
if x < h:
return 0.
return 1.

def seuil_image_1(image,h):
        img = copy.deepcopy(image)
        a,b,c = np.shape(img)
        deriv_img=np.zeros((a-1,b))
        for j in range(b):
                for i in range(a-1):
                        deriv_img[i,j]=abs(img[i+1,j,0]-img[i,j,0])
        liste_courbe_route0=[]
        for j in range(b):
                ordonne=a-1;
                        for i in range(a-2,0,-1):
                        val_provisoire = seuil(deriv_img[i,j],h)
                        if val_provisoire==1. :
                                ordonne=i;
                                liste_courbe_route0.append(ordonne)
        return liste_courbe_route0,a,b


liste_courbe_route,nb_lignes , nb_col = seuil_image_1(image,0.005)
```

4. Programmes:

```
def lissage_de_liste_à_5_pas(pt):
        moyenne = np.zeros((len(pt)))
        moyenne[0], moyenne[1], moyenne[2],moyenne[3], moyenne[4], moyenne[5] , moyenne[-1],moyenne[-2],moyenne[-3],
moyenne[-4],moyenne[-5] = pt[0], pt[1],pt[2],pt[3],pt[4],pt[5] , pt[-1], pt[-2], pt[-3], pt[-4], pt[-5]
        for i in range(5, len(pt)-5):
                moyenne[i] = (pt[i-5]+ pt[i-4] + pt[i-3]+ pt[i-2]+ pt[i-1] + pt[i] + pt[i+1] + pt[i+2]+ pt[i+3]+ pt[i+4]+
pt[i+5])/11.0
        return moyenne


##Lissage Passe Bas

# Fréquence de coupure
fc = 1/30000 # Hz
tau = 1/(2*np.pi*fc)

# Période d'échantillonnage
Te = 1 # s

# Préparation de la liste de sortie
listef_T_N = []
listef_T_N.append(listef_T_N1[0])


# Application du filtre
for i in range(1, len(listef_T_N1)):
        listef_T_N.append(listef_T_N[i-1]+Te/tau*(listef_T_N1[i-1]-listef_T_N[i-1]))
```

4. Programmes:

```
### II- Courbes vitesses et Vectoriasation

def dérivée(liste, t1):
if t1>0:
return liste[t1]-liste[t1-1]
elif t1==0:
return 0

# mainteant on crée une fonction courbe de vitesse précise mais suffisement simple : [90, 130,110,80,30,80] km/h

delt=60
vit1=np.zeros((N))
v1=90/3.6
v2=130/3.6
v3=110/3.6
v4=80/3.6
v5=30/3.6
v6=80/3.6

a_0 = 60 #sec (temps passage de 0 à90km/h)
a_1= int(N*8/100) #(5min)
a_2= int(N* 35/100)
a_3= int(N *49/100)
a_4= int(N* 77/100)
a_5= int(N * 85/100)

for t in range(a_0):
vit1[t]= v1/delt * t
for t in range(a_0,a_1):
vit1[t]= v1
for t in range(a_1, a_1+delt):
vit1[t] = (v2-v1)*(t-a_1)/(a_1+delt-a_1) + v1
for t in range(a_1 + delt, a_2):
vit1[t]= v2
for t in range(a_2, a_2+delt):
vit1[t] = (v3-v2)*(t-a_2)/(a_2+delt-a_2) + v2
for t in range(a_2 + delt, a_3):
vit1[t]= v3
for t in range(a_3, a_3+delt):
vit1[t] = (v4-v3)*(t-a_3)/(a_3+delt-a_3) + v3
for t in range(a_3 + delt, a_4):
vit1[t]= v4
for t in range(a_4, a_4+delt):
vit1[t] = (v5-v4)*(t-a_4)/(a_4+delt-a_4) + v4
for t in range(a_4+delt, a_5):
vit1[t] = v5
for t in range(a_5,a_5+delt):
vit1[t] = (v6-v5)*(t-a_5)/(a_5+delt-a_5) + v5
for t in range(a_5+delt, N-delt):
vit1[t]=v6
for t in range(N-delt, N):
vit1[t]= (0.-v6)*(t-(N-delt))/(N-(N-delt))+ v6


vit = lissage_de_liste_à_5_pas( vit1 )
```

4. Programmes:

```
### Calcul des VECTEURS vitesse au cours du temps

# étape 1: Transformer la liste des altitudes obtenues géométriquement en liste fonction du temps

#u00 distance entre deux abscisses spaciales 1,4 m (réels)

u00=1.4
n_max=len(listef_T_N)

distance_lineique=np.zeros((n_max))
distance_lineique[0]=0

for i in range(n_max-1):
        distance_lineique[i+1]=distance_lineique[i]+np.sqrt((listef_T_N[i+1]-listef_T_N[i])**2+u00**2)


L = np.zeros((N))

for t in range(1,N-1):
        L[t+1]=L[t]+vit[t]*1. #1sec

#u00 * x_numero_case_au_temps_t[t]=x(t)

x_numero_case_au_temps_t=np.zeros((N))
x_numero_case_au_temps_t[0]=0
n=0

## Détermination position géographique réelle au cours du temps:

for t in range(1,N):
        while (distance_lineique[n]<L[t] and n+1<=n_max-1):
                n=n+1
        x_numero_case_au_temps_t[t]=n
```

4. Programmes:

```
## étape 2: Détermination de l'angle thêta que fait la route avec notre repère xyz galiléen


theta_au_temps_t1= np.zeros((N))

for t in range(N-1):
                if int(x_numero_case_au_temps_t[t+1])==int(x_numero_case_au_temps_t[t]) and t>=1:
                                provisoire=theta_au_temps_t1[t]
                else:
                                provisoire=np.arctan((listef_T_N[int(x_numero_case_au_temps_t[t+1])] - listef_T_N[
int(x_numero_case_au_temps_t[t])] )/ (u00))

theta_au_temps_t1[0]=theta_au_temps_t1[1]

theta_au_temps_t = lissage_de_liste_à_5_pas( theta_au_temps_t1 )
```

4. Programmes:

```
##étape 3 vectoriser la vitesse


vitesse_x_au_temps_t=np.zeros((N))

vitesse_z_au_temps_t=np.zeros((N))

vit_vec1 = []

for t in range(N):
                vitesse_x_au_temps_t[t]=vit[t]*m.cos(theta_au_temps_t[t])
                vitesse_z_au_temps_t[t]=vit[t]*m.sin(theta_au_temps_t[t])
                vit_vec1.append(np.array([vitesse_x_au_temps_t[t],0,vitesse_z_au_temps_t[t]]))

vit_vec = np.array( vit_vec1 )


## enfin fonction route_fontion_temps utile aussi de taille N:

route_fonction_temps1 = np.zeros((N))
for i in range(N):
                route_fonction_temps1[i]= listef_T_N[int(x_numero_case_au_temps_t[i])]

route_fonction_temps = lissage_de_liste_à_5_pas( route_fonction_temps1 )
```

4. Programmes:

```
## III- Mécanique

#repère galiléen x abscises route ( -> ), z altitude ( Î ), y rentrant (x)

## Vectorisation des forces

g_v = g* np.array([0.,0.,-1.])

# On négligera la varitation de masse

masse_essence=np.zeros((N))
for t in range(N): #N-1
        masse_essence[t] = rho_essence * res[0]


#masse volumique Comprise 720,0 kg/m3
masse_totale = Masse_ensemble + masse_essence

#Poids

Poids1 = []
for k in range(N):
        Poids1.append(masse_totale[k]* g_v )
Poids = np.array(Poids1)



# La force de frottements fluide est de la forme : F = -1/2*Cx*µ*Sv^2
#S = Largeur * hauteur


Force_frottements1= []
# 1) frottements fluides avec l'air
# 2) frottements solides des roues sur leur essieu
# 3) résistance au roulement

# Je néglige 2 et regroupe 1 et 3 en une seule force qui dépend régime (en gros qui devient (1) à grande vitesse et (3) à faible vitesse).
# Et on décide de modéliser 3 comme une force simple, inspirée du frottement solide par exemple.

# Par exemple, on peut estimer le temps que met une voiture à s'arrêter en passant de 30km/h à 0, et évaluer ce que doit valoir une force constante pour réaliser ce temps.

# valeur approximative : K = masse/delta_t * vo donc on peut estimer K ~= 1535 * 8.333 /30 = 430 #N (kg.m.s-2)

k = 430 #N (kg.m.s-2)

for t in range(N):
        Force_frottements1.append((-0.5 * Cx * rho_air * hauteur_véhicule * largeur_véhicule *vit[t] *vit_vec[t] + np.array([-k*m.cos(theta_au_temps_t[t]),0,-k*m.sin(theta_au_temps_t[t])])))

Force_frottements = np.array( Force_frottements1 )
```

4. Programmes:

```
## Calcul des puissances des forces et utilisation du produit scalaire

# Vecteur acélération
acc_vec1 = []
acc_x =[0]
acc_z= [0]

for k in range(N):
        if k>0:
            acc_x.append(dérivée(vitesse_x_au_temps_t, k))
            acc_z.append(dérivée(vitesse_z_au_temps_t, k))
            acc_vec1.append( np.array([acc_x[k], 0, acc_z[k]]))

acc_vec = np.array(acc_vec1)

##puissance forces

def puissance_de_force(force, vecteur_vitesse):
        L = np.zeros((N))
        for k in range (N):
            L[k] = np.vdot(force[k],vecteur_vitesse[k])
        return L

P_Poids = puissance_de_force(Poids, vit_vec)
P_frottements = puissance_de_force(Force_frottements, vit_vec)


## Théorème de l'énergie cinétique
## m*a* v = Pm + Pf + Ppoids

Pm1 = []
for t in range (N):
            Pm1.append(masse_totale[t]*
np.vdot(acc_vec[t], vit_vec[t]) - P_frottements[t] - P_Poids[t])


# Lissage de la courbe

fc1 = 1/110 # Hz
tau1 = 1/(2*np.pi*fc1)

# Période d'échantillonnage
Te = 1 # s

# Préparation de la liste de sortie
Pm2 = []
Pm2.append(Pm1[0])

# Application du filtre
for i in range(1, len(Pm1)):
            Pm2.append(Pm2[i-1]+Te/tau1*(Pm1[i-1]-Pm2[i-
1]))

Pm = np.array(Pm2)
```

4. Programmes:

```
## Consommation

def consommation_fonction_de_la_puissance_moteur(puissance_t,mode_t, t1):
        V1 = 0
        V2 = 0
        V3 = 0
        if t1==0 :
                return V1,V2,V3

        else :
                if mode_t[t1]==1 and puissance_t[t1] > 0 :
                        V1 = puissance_t[t1]*100 / (n_electrique * capacité_batterie*3600000)
                elif mode_t[t1]==1 and puissance_t[t1] <=0 :
                        V1 = puissance_t[t1]*100 / (2*n_electrique * capacité_batterie*3600000)
                elif mode_t[t1]==0: #and puissance_t[t1]> 0:
        # exercice 4 thermochimie:
        # essence assimilée à l'octane
        # Quelle est, exprimée en litres aux 100 km, la consommation d'essence d'une voiture roulant à vit[t] ms-1 en développant une puissance de Pm (environ 40 ch???) (1 ch = 736 W) J/s.
        # Nous assimilerons l'essence à de l'octane, de masse volumique p = 0,72 kg L-1. Le moteur rendement énergétique de 28 %. Données : C3H18(g) +25/202(g) → 8CO2(g) + 9H,O avec ArH° = -5507 kJ mol-1.

                        V2 = puissance_t[t1] * M / (n_global*ArH_essence *rho_essence) #en L/s

                if puissance_t[t1]<=0 and bat[t1]<= SEUIL_bat_max :
                # # regain d'énergie
                        V3 = puissance_t[t1]*100 / (n_electrique* capacité_batterie*3600* 1000)
                return V1, V2, V3
```

4. Programmes:

```
## boucle Mode(thermique électrique) - niveau réservoir - charge batterie ...

histeresis = np.zeros((N)) ##condition d'histérésis = SEUIL_DE_BASCULEMENT_THERM_VERS_BATTERIE

# initialisation histérésis
if init_batterie > SEUIL_DE_BASCULEMENT_BATTERIE_VERS_THERM:
        histeresis[0] = 1
else :
        histeresis[0] = 0

def choix_histeresis(t1):

        if histeresis[t1]==1 and bat[t1] > SEUIL_DE_BASCULEMENT_BATTERIE_VERS_THERM:
                histeresis[t1]=1
        elif histeresis[t1] ==1 and bat[t1] <= SEUIL_DE_BASCULEMENT_BATTERIE_VERS_THERM:
                histeresis[t1] =0
        elif histeresis[t1]==0 and bat[t1] <= SEUIL_DE_BASCULEMENT_THERM_VERS_BATTERIE:
                histeresis[t1]=0
        if histeresis[t1]==0 and bat[t1] > SEUIL_DE_BASCULEMENT_THERM_VERS_BATTERIE and bat[t1]<=SEUIL_bat_max:
                histeresis[t1]= 1
```

4. Programmes:

```
def choix_thermique_electrique(t1):

        if mode[t1] ==0:
            if histeresis[t1] ==0:
                return 0 #thermique reste thermique
            if histeresis[t1] ==1:
                return 1 #thermique devient électrique
        if mode[t1]==1:
            if histeresis[t1] ==1 :
                return 1 #électrique reste électrique
            if histeresis[t1] ==0:
                return 0 #électrique devient thermique

def mise_jour_batterie_reservoire(t1):
        bat[t1]= bat[t1]-abs(decharge) + abs(recharge_de_la_batterie_en_décélération)
        res[t1]= res[t1] -abs(vider_essence)

##Recharger par thermique est devenu trop compliqué à mettre en oeuvre dans nos calculs donc les seuls gains d'énergie seront dû à la décélération et pente négative (Pm <=0)

def obtenir_decharge_recharge_vider_essence(Puissance_moteur, liste,t1):
        recharge_de_la_batterie_en_décélération =0
        recharge_par_thermique=0
        decharge1=0
        Vider_essence1=0
        V1 , V2 , V3 = consommation_fonction_de_la_puissance_moteur(Pm, mode, t1)
        decharge1=V1
        vider_essence1=V2
        recharge_de_la_batterie_en_décélération = V3
        return decharge1, vider_essence1, recharge_de_la_batterie_en_décélération,recharge_par_thermique
```

4. Programmes:

```
for t in range(N):

#à l'instant t (seconde du trajet) on effectue une mise à jour :
#debut du round

        decharge, vider_essence, recharge_de_la_batterie_en_décélération, recharge_par_thermique =obtenir_decharge_
recharge_vider_essence(Pm, mode, t)

        mode[t]=choix_thermique_electrique(t)

        mise_jour_batterie_reservoire(t)

        choix_histeresis(t)

        if t+1<=N-1:

                mode[t+1]=mode[t]

                bat[t+1]=bat[t]

                res[t+1]=res[t]

                histeresis[t+1] = histeresis[t]

#fin du round
```

FIN

3.2 Retour au problème de départ :

CONCLUSION :

- Economies de consommation vérifiées.
- Limites de la simulation :
  - Courbe d'altitudes n'est pas exacte ;
  - Vitesse simple fut imposée ;
  - Simplifications nombreuses au niveau du bilan des efforts ;
  - Simplifications des phénomènes en jeu pour les consommations ;
  - Le phénomène de recharge de la batterie via le moteur thermique a été abandonné ;
- Ouverture : Possibilité d'application des prévisions de consommation en retour 'temps réel' au conducteur.
- Aujourd'hui la plupart des véhicules hybrides réalisent un calcul en temps réel qui détermine le mix électrique/thermique Mais c'est un calcul instantané 'originalité de mon TIPE est de considérer la consommation du parcours entier. Dès lors on pourrait faire un calcul d'optimisation avant le départ et la suivre pendant le parcours.