# rapport jumeau numérique eVECTORS

Clément FOUQUET

March 2025

## Table des matières

# 1 Introduction

# Notations

| Symbole | Description | nom variable/fonction | Unité |
|---|---|---|---|
| $F_{\text{totale}}$ | Force totale appliquée au système | Force().F_totale() | N |
| $F_{utile}$ | Somme des forces contre le déplacement | Force().F_usefull() | N |
| $F_{aero}$ | Force de traînée aérodynamique | Force().F_aero() | N |
| $F_{gravite}$ | Force gravitationnelle | Force().F_slope() | N |
| $F_{roulement}$ | Force de résistance au roulement | Force().F_rolling() | N |
| $F_{traction}$ | Force de traction | Force().F_traction() | N |
| $N_F$ | Réaction normale du sol | | N |
| $F$ | Force générique | | N |
| $S_{aero}$ | Surface frontale du véhicule | eV_properties.S | $\mathrm{m}^2$ |
| $\rho$ | Densité de l'air | config.rho_air | $\mathrm{kg\,m^{-3}}$ |
| $c_x$ | Coefficient de traînée aérodynamique | eV_properties.Cx | 1 |
| $R$ | Rayon des roues | eV_properties.rayon | m |
| $\mu_r$ | résistance au roulement | eV_properties.mu | m |
| $C_r$ | résistance au roulement | eV_properties.Cr | 1 |
| $m$ | Masse du véhicule | Force().mu | kg |
| $g$ | Accélération gravitationnelle | config.gravity | $\mathrm{m\,s^{-2}}$ |
| $\theta$ | Angle d'inclinaison de la route | angle | rad |
| $T$ | Couple moteur | couple | N m |
| $t$ | Temps | t | s |
| $v$ | Vitesse du véhicule | v | $\mathrm{m\,s^{-1}}$ |
| $P$ | Puissance | | W |
| $E$ | Énergie | | J |
| $dl$ | Déplacement élémentaire | | m |
| $dt$ | Temps élémentaire | dt | s |
| $\eta_{batterie}$ | Rendement de la batterie | | 1 |
| $\eta_{ondulateur}$ | Rendement de l'onduleur | | 1 |
| $\eta_{moteur}$ | Rendement du moteur | eVECTORS.eta() | 1 |
| $\eta_{reducteur}$ | Rendement du réducteur | | 1 |
| $\eta_{transmssion}$ | Rendement de la transmission | | 1 |
| $\eta_{total}$ | Rendement total | | 1 |
| $\eta_{max}$ | Rendement maximal du moteur | eV_properties.eta_max | 1 |
| $\omega$ | Vitesse de rotation du moteur | omega | $\mathrm{rad\,s^{-1}}$ |
| $\omega_{opt}$ | Vitesse de rotation optimale du moteur | eV_properties.W0 | $\mathrm{rad\,s^{-1}}$ |
| $\omega_{moteur}$ | Vitesse de rotation réelle du moteur | | $\mathrm{rad\,s^{-1}}$ |
| $\omega_{roues}$ | Vitesse de rotation des roues | | $\mathrm{rad\,s^{-1}}$ |
| $r_{ratio}$ | Rapport de transmission | | 1 |
| $T_{opt}$ | Couple moteur optimal | eV_properties.T0 | N m |
| $a_{moteur}$ | Constante d'ajustement | eV_properties.coef_A | $\mathrm{s^2\,rad^{-2}}$ |
| $b_{moteur}$ | Constante d'ajustement | eV_properties.coef_B | $\mathrm{N^{-2}\,m^{-1}}$ |
| $P_{phare}$ | Puissance des phares | eV_properties.P_headlight | W |
| $T_{ext}$ | Température extérieure | T_ext | K |
| $T_{\text{habitacle}}$ | Température intérieure de l'habitacle | equipement().T_transi( | K |
| $c_p$ | Capacité thermique massique de l'air | config.Cp_air | $\mathrm{J\,kg^{-1}\,K^{-1}}$ |
| $k_{thermique}$ | Coefficient de pertes thermiques | eV_properties.k_therm | $\mathrm{W\,K^{-1}\,m^{-2}}$ |
| $S_{tot}$ | Surface totale extérieure | eV_properties.S_tot | $\mathrm{m}^2$ |
| $\dot{m}_{air}$ | débit d'air du chauffage | eV_properties.dmdt | $\mathrm{kg\,s^{-1}}$ |
| $P_{\text{chauffage}}$ | Puissance thermique du chauffage | equippement.P_chauffage() | W |
| $P_{\text{rayonnement}}$ | Puissance thermique due au rayonnement solaire | equippement.P_rayonnement() | W |
| $I_{soleil}$ | Intensité du rayonnement solaire | config.I_soleil | $\mathrm{W\,m^{-2}}$ |

# 2 Lecture des fichiers

Pour être utilisé, le logiciel a besoin de fichiers d'entrée. On utilise ici des fichiers ".yaml".

## 2.1 Fichier véhicule

Le fichier véhicule est construit de la façon suivante :

```yaml
vehicle:
  masse: 600 # kg
  radius: 0.3 # m
  Cx: 0.3 # coefficient de trane
  Cr: 0.01 # coefficient de rsistance au roulement
  S: 2 # m^2
  mu: 0.0068 # coefficient mu
  nbr: 3
  V_eVECTORS: 4 # m^3
  S_tot: 5 # m^2

  P_headlight: 80 # 2 x 40W
  dmdt: 0.05 # kg/s
  ktherm: 10 # W

  I_soleil: 1000 # W/m^2
  S_vitre: 2 # m^2
  a_vitre: 0.6 # sans unit

  n_moteur: 1
  W0: 500 # rad/s
  T0: 100 # Nm
  eta_max: 0.95
  eta_min: 0.15
  coef_A: 1
  coef_B: 1

  rapport: 9
  eta_batterie: 0.95
  eta_reducteur: 0.96
  eta_transmission: 0.96
  eta_onduleur: 0.96
  R_int: 0.08 # Ohms
```

Listing 1 – Exemple de fichier de configuration YAML pour le véhicule

Le fichier est lu puis crée une instance de la classe eVECTORS, avec les propriétés du fichier grâce à la fonction"from_yaml" de la classe eVECTORS.

## 2.2 fichier test

Pour lancer le test souhaité, il faut remplir un fichier ".yaml" avec cette forme :

```yaml
calcul : test

valeur:
  fichier: test_Force
  classe: test_energie
  fct: affichage
  arguements:
    nfct: 1
    nv: 1
    ntheta: 1
    tf: 120
    dt: 1
    v0: 20
    angle: 0
```

Listing 2 – Exemple de fichier de configuration YAML pour les tests

- fichier doit être le nom du fichier de test
- classe doit être la classe à appeler
- fct est la fonction dans la classe que l'on souhaite appeler
- arguments sont les arguments pris par la fonction, se référer à la fonction en question dans la section dédié.

## 2.3 Interface graphique

Pour faciliter l'utilisation, une interface graphique est disponible. L'appel depuis l'invite de commande est possible de cette façon :

```
python main.py -i y
```

Elle propose :

- analyse énergétique trajet, pour calculer la dépense d'énergie sur un trajet
- test général, pour vérifier les différentes fonctions de calculs en comparaison de solution analytique
- lecture de fichier, pour lire des fichiers

### 2.3.1 analyse énergétique trajet

Cette option permet le calcul de l'énergie dépensée sur un trajet pour un véhicule donné. Il faut donc fournir un fichier vehicule.yaml et un fichier trajet.gpx. On a ensuite le choix de l'unité ainsi que de sauvegarder le résultat. Il faut ensuite lancer le calcul. On peut facilement changer les unités, il suffit de

relancer le calcul. Une fois sauvegardé, le fichier résultats peut être lu avec l'option lecture de fichier, on peut aussi changer l'unité des graphiques simplement, il suffit de cliquer sur "change unite" il n'est pas nécessaire de relancer le script.

### 2.3.2 test général

Cette option permet d'afficher le résultat de toutes les fonctions de calcul dans des conditions de test. Est affiché le nom de la fonction et son erreur en pourcentage. Voir le chapitre adéquat pour plus d'informations sur le test.

## 2.4 Lancement de calcul

On peut lancer des calculs par l'invite de commande.

```
python main.py -f [vehicle_file.yaml] [route_calcul.yaml]
```

Le fichier [vehicle_file.yaml] doit être au format fichier véhicule (sous-section 2.1). Le fichier [route_calcul.yaml]

```
calcul : trajet

trajet : fichier_test/coordonnees/mon_trajet.gpx

units :
  Energy: J
  Power: W
  Distance: km


save: True

file name: test_results
```

avec :

- trajet : le fichier trajet en .gpx (voir : section 7)
- units : les unités de calcul
  - Energy :
    - J pour Joule
    - kWh pour kiloWatt.heure
  - Power :
    - W pour Watt
    - kW pour kiloWatt
    - Horsepower
  - Distance :
    - m pour mètre
    - km pour kilomètre
    - mile
    - time

— save : True or False pour sauvegarder le fichier
— file : nom du fichier si save == True

## 2.5 Calcul batch

On peut facilement lancer plusieurs calculs depuis l'invite de commande de la même manière qu'en sous-section 2.4 avec un fichier de cette forme :

```
calcul : batch

trajet :
  file: workspace/route
  route:
  - mon_trajet.gpx
vehicle :
  file: workspace/vehicle
  vehicle:
  - vehicle.yaml
  - vehicle_2.yaml
  - vehicle_3.yaml


units :
  Energy: J
  Power: W
  Distance: km

save: True
filedire: workspace/results
fileprefix: result
```

avec :
— file : le dossier contenant les fichiers vehicule ou route
— route : la liste des noms des fichiers .GPX (doit être une liste ou "all")
— vehicle : la liste des fichier véhicule (doit être une liste ou "all")
— filedire : le dossier d'arrivé (il sera crée si non existant)
— fileprefix : le debut du nom de fichier résultats
Les fichiers résultats sont de la forme

```
    nom_fichier = yaml_dict["fileprefix"]+f"_{vehicle_name}_{route_name}"
```

Il est possible de choisir tout un dossier, la syntaxe du fichier est la suivante :

```
calcul : batch

trajet :
  file: workspace/route
  route: all
vehicle :
  file: workspace/vehicle
```

```
  vehicle: all


units :
  Energy: J
  Power: W
  Distance: km

save: True
filedire: workspace/results/test
fileprefix: result
```

### 2.5.1 Création de véhicule batch

Il est possible de créer rapidement des fichiers véhicule pour faire varier certains paramètres. Le fichier a ce format :

```
calcul: batch vehicle
# This file is used to generate multiple vehicle configurations for batch
    processing
nbr_file: 5
variable:
  S:
    lower bound: 1.6
    upper bound: 2.4
    probability law: normal
  masse:
    lower bound: 240.0
    upper bound: 500.0
    probability law: uniform
filedire: workspace/vehicle_test
fileprefix: vehicle_batch_test
vehicle:
  Crr: 0.01
  Cx: 0.3
  I_soleil: 1000
  P_headlight: 80
  R_int: 0.08
  S: 2
  S_tot: 5
  S_vitre: 2
  V_eVECTORS: 4
  a_vitre: 0.6
  dmdt: 0.05
  eta_batterie: 0.95
  eta_onduleur: 0.96
  eta_reducteur: 0.96
  eta_transmission: 0.96
  ktherm: 10
```

```
  masse: 300
  n_moteur: 1
  nbr: 1
  radius: 0.3
  self.efficiency_map: "fichier_test\efficiency_map_motor_generator"
  rapport: 9
```

avec :

- nbr_file : le nombre total de fichier différent
- variable : le nom de la variable à modifier
- lower bound : la valeur minimal de cette variable
- upper bound : la valeur maximal de cette variable
- probability law : la loi de probabilité suivi par cette variable,soit "uniform" soit "normal" (pour la loi normal les limites fixent la zone ou 99% des valeurs sont comprise)
- filedire : le dossier d'arrivé (il sera crée si non existant)
- fileprefix : le debut du nom de fichier résultats
- vehicle les valeurs par défaut prise si la valeur n'est pas présente dans "variable"

Il est possible de créer ce fichier facilement depuis l'interface graphique.
On peut lancer des calculs par l'invite de commande. Le fichier véhicule, bien que nécessaire pour lancer le calcul, sera ignoré.

```
python main.py -f [vehicle_file.yaml] [batch_vehicle.yaml]
```

### 2.5.2 Affichage batch

Le module "module_display" permet d'afficher certaines données. Pour cela, on appelle le programme :

```
python main.py -f [batch_affichage.yaml]
```

Le fichier a cette construction :

```
calcul : display

affichage choice: energy bar # pie, energy bar, energy power, mean, ,
    energy, trajectory

results :
  directory: workspace/results/Trajet
  file: all

units :
  Energy: kWh
  Power: kW
  Distance: km
```

- `calcul` : `display`
  Type de calcul demandé. Ici, il s'agit uniquement d'un affichage.
- `affichage choice` : `energy bar`
  Type d'affichage sélectionné. Les options possibles incluent : `pie`, `energy bar`, `energy power`, `mean`, `energy`, `trajectory`.
- `results` :
  - `directory` : `workspace/results/Trajet`
    Répertoire où les résultats de l'affichage sont enregistrés. Il est créé s'il n'existe pas.
  - `file` : `all`
    Nom du fichier dans lequel les résultats sont enregistrés.
- `units` : unités utilisées pour les grandeurs physiques :
  - `Energy` : `kWh`
  - `Power` : `kW`
  - `Distance` : `km`

## 2.6 aide

Une aide des entrées disponibles est accessible avec :

```
python main.py -h
```

# 3 Véhicule

Le véhicule possède des propriétés qu'il faudra déterminer par mesure directe ou indirecte. Le programme doit donc pouvoir se modifier simplement.

## 3.1 Valeurs d'intérêt

On liste les valeurs comprises dans eV_properties.py :

| variable | Description |
|---|---|
| masse | Masse totale du véhicule (kg) |
| radius | Rayon des roues (m) |
| Cx | Coefficient de traînée aérodynamique (sans dimension) |
| Cr | Coefficient de résistance au roulement (sans dimension) |
| S | Surface frontale du véhicule ($m^2$) |
| mu | Coefficient de frottement de roulement (m) |
| nbr | Nombre de roues |
| V_eVECTORS | Volume intérieur du véhicule ($m^3$) |
| P_headlight | Puissance des phares (W) |
| dmdt | Débit massique d'air à travers les bouches d'aération (kg/s) |
| ktherm | Coefficient de pertes thermiques (W) |
| S_tot | Surface totale du véhicule ($m^2$) |
| I_soleil | Intensité du rayonnement solaire ($W\,m^{-2}$) |
| S_vitre | Surface des vitres ($m^2$) |
| a_vitre | Absorbance des vitres |

TABLE 1 – variable et description des paramètres du véhicule

## 3.2 utilisation

Afin de pouvoir tester facilement plusieurs valeurs pour une caractéristique, il faut créer une instance de eVECTORS, puis l'envoyer dans le module adéquat. Exemple modifier la masse et le coefficient de frottement de l'air :

```
1 mon_vehicule = eV_properties.eVECTORS()
2 mon_vehicule.masse = 1000
3 mon_vehicule.Cx = 0.2
```

On peut à tout moment afficher tous les paramètres simplement :

```
1 mon_vehicule = eV_properties.eVECTORS()
2 print(mon_vehicule)
```

# 4 Force extérieur

L'étude des forces extérieures appliquées à un véhicule est essentielle pour comprendre son comportement dynamique et estimer sa consommation énergétique. Dans ce cadre, nous nous intéressons aux différentes forces influençant le mouvement, telles que la traînée aérodynamique, la résistance au roulement, la force de traction, la force gravitationnelle et les forces d'inertie. L'objectif principal est de modéliser ces contributions afin de quantifier la puissance et l'énergie nécessaires au déplacement du véhicule dans diverses conditions.

## 4.1 Référentiel

On définit le référentiel de base $\mathcal{R}$ avec ces axes orientés vers l'Est le Nord et la verticale. Le GPS mesurant la vitesse dans ce référentiel, aucun effet inertiel du véhicule ne doit être pris en compte. On négligera les effets inertiels dus à la rotation de la Terre.

### 4.1.1 Définition des angles

On définit $\lambda$ l'angle de la pente donc l'angle entre $\vec{e}_v$ et le plan $(x, y)$. On définit la matrice de passage associée.

$$R_\lambda = \begin{bmatrix} \cos\lambda & 0 & \sin\lambda \\ 0 & 1 & 0 \\ -\sin\lambda & 0 & \cos\lambda \end{bmatrix} \tag{1}$$

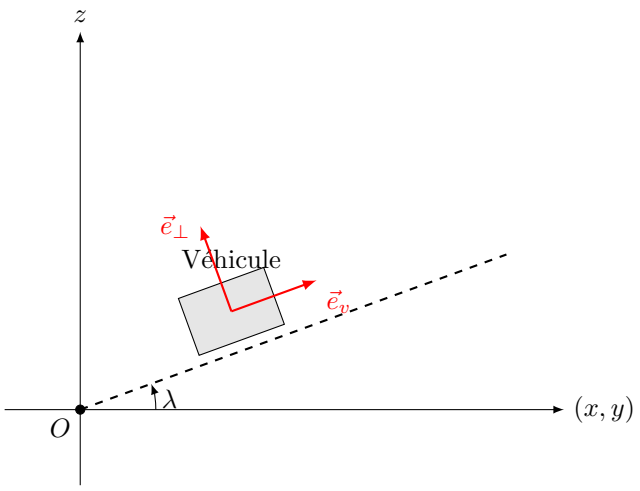



FIGURE 1 – Schéma du véhicule sur une pente inclinée d'un angle $\lambda$ dans le plan $((x, y), z)$

## 4.2 Principe fondamentale de la dynamique (PFD)

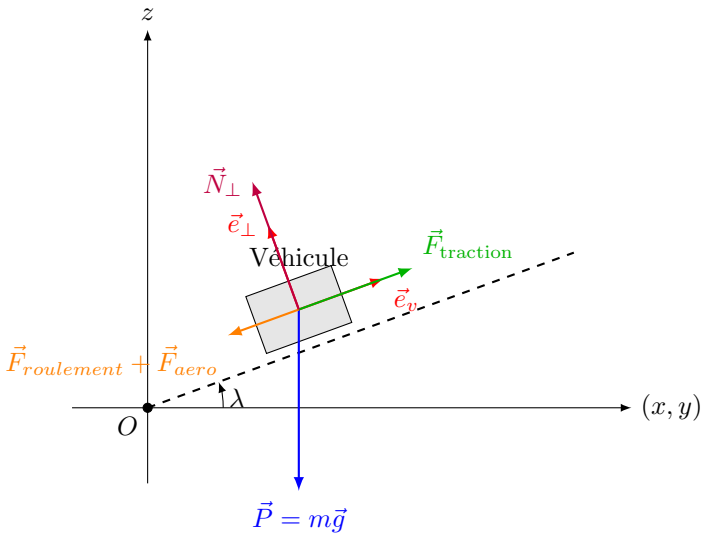



FIGURE 2 – Projection des forces extérieures dans le référentiel du véhicule sur une pente inclinée d'un angle $\lambda$

Soit $\vec{v}_v$ le vecteur vitesse du véhicule dans le référentiel galiléen. On définit :

— Le vecteur unitaire tangent à la trajectoire du véhicule :

$$\vec{e}_v = \frac{\vec{v}_v}{\|\vec{v}_v\|}$$

— Le vecteur unitaire orthogonal à $\vec{e}_v$ dans le plan horizontal (plan $(x, y)$) :

$$\vec{e}_{v_\perp} = \vec{e}_\perp \wedge \vec{e}_v$$

— la base $(\vec{e}_v, \vec{e}_{v_\perp})$ est un repère de Frenet ce qui facilite la projection des forces d'inertie

Ainsi, la base locale associée au référentiel du véhicule $\mathcal{R}'$ est définie par le triplet de vecteurs orthonormés :

$$(\vec{e}_v, \vec{e}_{v_\perp}, \vec{e}_\perp)$$

On projette dans le référentiel du véhicule R″ avec comme vecteur de base $(\vec{e}_v, \vec{e}_{v_\perp}, \vec{e}_\perp)$.

On s'intéresse à l'accélération subie par le véhicule donc :

$$m\vec{a} = \sum \vec{F}_{ext} \tag{2}$$

avec :

$$\sum \vec{F}_{ext} = -\frac{1}{2}\rho S c_x \|\vec{v}_v\| \vec{v}_v - C_{rr}\|\vec{N}\|\vec{e}_v + F_{traction}\vec{e}_v + N_{\perp}\vec{e}_{\perp} + N_{v_\perp}\vec{e}_{v_\perp} - mg\vec{e}_z \tag{3}$$

Ou $\vec{e_z}$ est le vecteur unitaire orthogonal au plan $(x, y)$.

On projette $m\vec{g}$ dans le référentiel du véhicule :

$$R_\lambda \vec{g} = \begin{bmatrix} \cos\lambda & 0 & \sin\lambda \\ 0 & 1 & 0 \\ -\sin\lambda & 0 & \cos\lambda \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = -g \begin{bmatrix} \sin\lambda \\ 0 \\ \cos\lambda \end{bmatrix} \tag{4}$$

Qui donne avec nos vecteurs : $\vec{g} = -g(\sin\lambda\vec{e}_v + \cos\lambda\vec{e}_\perp)$.

En négligeant les brusques variations de pente $\dot{\lambda} << 1$ et en supposant que les virages sont dans le plan $(x, y)$, on trouve :

$$\begin{cases} ma_v(t) = -\frac{1}{2}\rho S c_x v_v^2 - C_{rr}N + F_{traction} - mg\sin\lambda \\ ma_{v_\perp}(t) = -m\frac{\|v_v\|^2}{R_{v_\perp}} + N_{v_\perp} \\ ma_\perp(t) = N_\perp - mg\cos\lambda \end{cases} \tag{5}$$

Avec :

- $a_{v_\perp} = 0$, pas de glissement latérale
- $a_\perp = 0$, la voiture reste au sol
- $R_{v_\perp}$ Le rayon de courbure dans le plan $(x, y)$

$$F_{aero} = -\frac{1}{2}\rho S_{aero} c_x v^2 \tag{6}$$

$$F_{roulement} = -C_{rr}\|\vec{N}_\perp\| = -C_{rr}mg\cos(\lambda) \tag{7}$$

$$F_{traction} = \frac{T}{R} \tag{8}$$

$$F_{gravite} = -mg\sin(\lambda) \tag{9}$$

$$F_{inertie} = ma_v \tag{10}$$

Le but étant de simuler la dépense énergétique et non le mouvement, on isole la force de traction qui est la dépense énergétique du véhicule :

$$F_{traction} = ma_v - (F_{aero} + F_{roulement} + F_{gravite}) \tag{11}$$

On calcule l'erreur en pourcentage avec la norme $L^2$ de la façon suivante :

$$Erreur_{\%} = \frac{\sqrt{\int (u_{analytique} - u_{calcule})^2 \, dt}}{\|u_{analytique}\|_{L^2}} * 100 \approx \frac{\sqrt{\sum_i \left(u^i_{analytique} - u^i_{calcule}\right)^2 dt^i}}{\|u_{analytique}\|_{L^2}} * 100 \tag{12}$$

*i est un indice et non une puissance.

### 4.2.1 Force de roulement

La solution analytique du système est donc :

$$\begin{cases} x(t) = v_0 t - C_r g cos(\theta)\dfrac{t^2}{2} \\ \dot{x}(t) = v_0 - C_r g cos(\theta) t \end{cases} \tag{13}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse initiale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— coefficient de résistance au roulement : $C_r = 0.01$
— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$
— angle de la pente : $\theta = 0$ rad

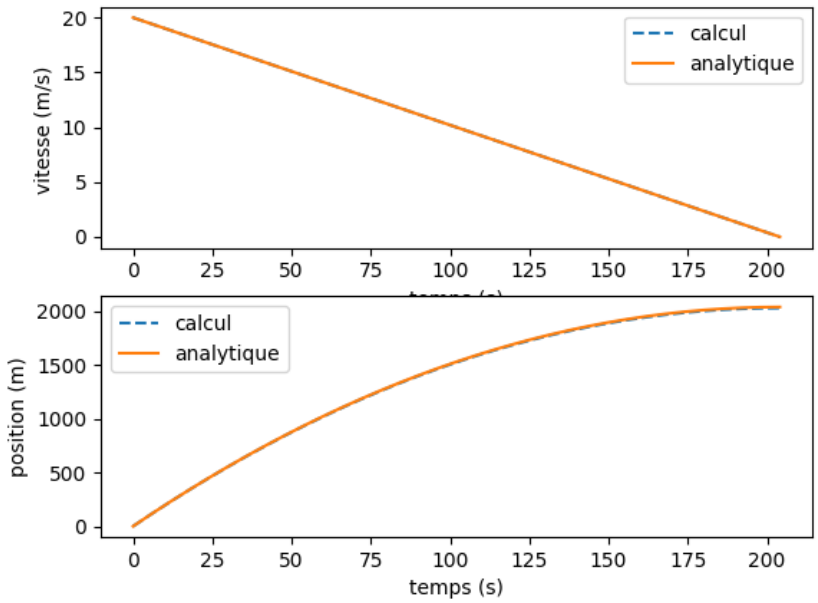



FIGURE 3 – Test de la force de roulement

On mesure ensuite l'erreur en fonction de la précision.

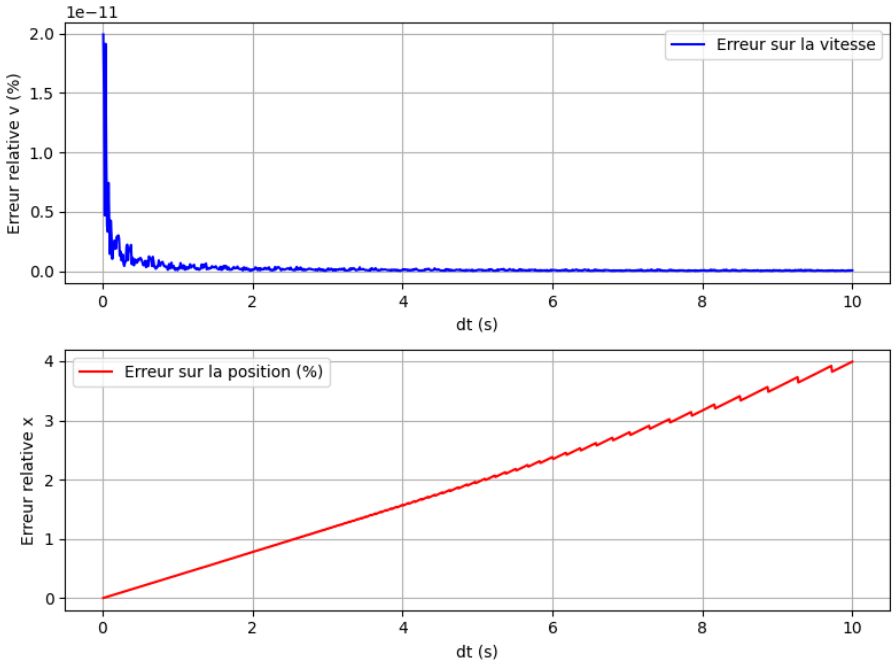



FIGURE 4 – Erreur de la force de roulement en fonction de la précision dt

### 4.2.2 Force de gravitation

La solution analytique du système est donc :

$$\begin{cases} x(t) = v_0 t - g sin(\theta) \dfrac{t^2}{2} \\ \dot{x}(t) = v_0 - g sin(\theta) t \end{cases} \tag{14}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse initiale : $v_0 = 200\ \text{m}\,\text{s}^{-1}$
— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$
— angle de la pente : $\theta = \frac{\pi}{2}$ rad

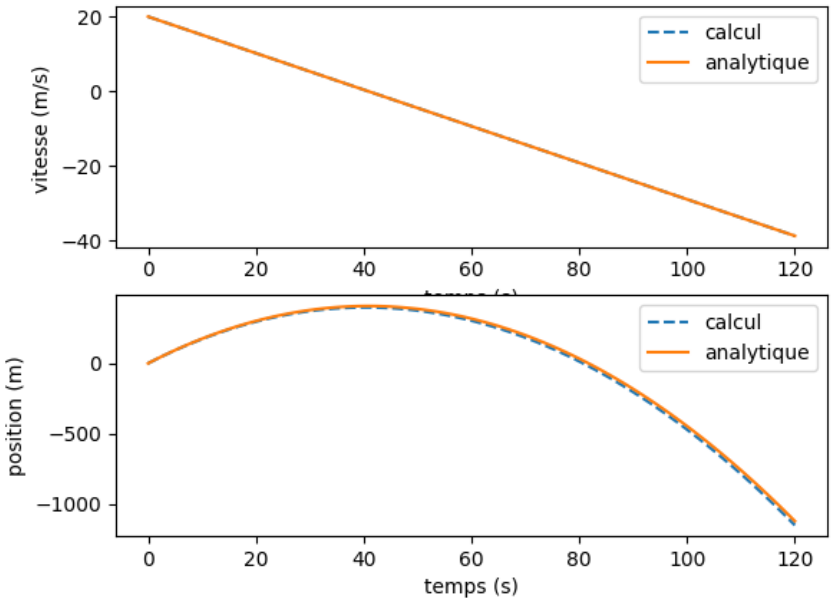



FIGURE 5 – Test de la force de gravitation

On mesure ensuite l'erreur en fonction de la précision.

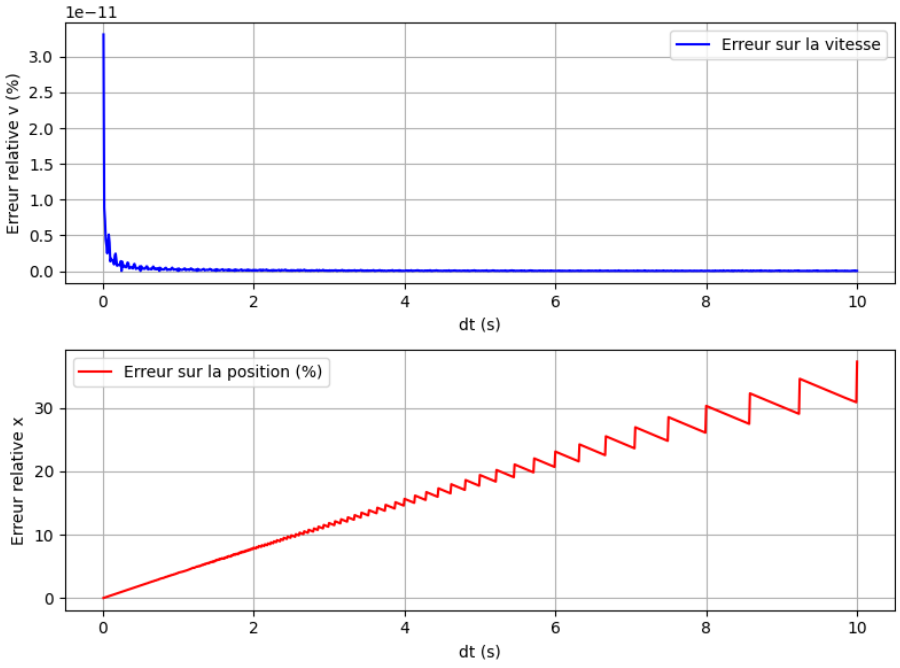



FIGURE 6 – Erreur de la force de gravitation en fonction de la précision dt

### 4.2.3 Force aérodynamique

La solution analytique du système est donc :

$$\begin{cases} x(t) = \dfrac{2m}{\rho_{air} S_{aero} c_x} ln\left(1 + \dfrac{\rho_{air} S_{aero} c_x v_0 t}{2m}\right) \\ \dot{x}(t) = \dfrac{v_0}{1 + \frac{\rho_{air} S_{aero} c_x}{2m} v_0 t} \end{cases} \tag{15}$$

avec ces valeurs et $dt = 1\text{s}$ :

— vitesse initiale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— Surface frontale : $S = 2\ \text{m}^2$
— densité de l'air : $\rho_{air} = 1.293\ \text{kg}\,\text{m}^{-3}$
— coefficient de frottement fluide : $c_x = 0.3$
— masse du véhicule : $m = 500\text{kg}$

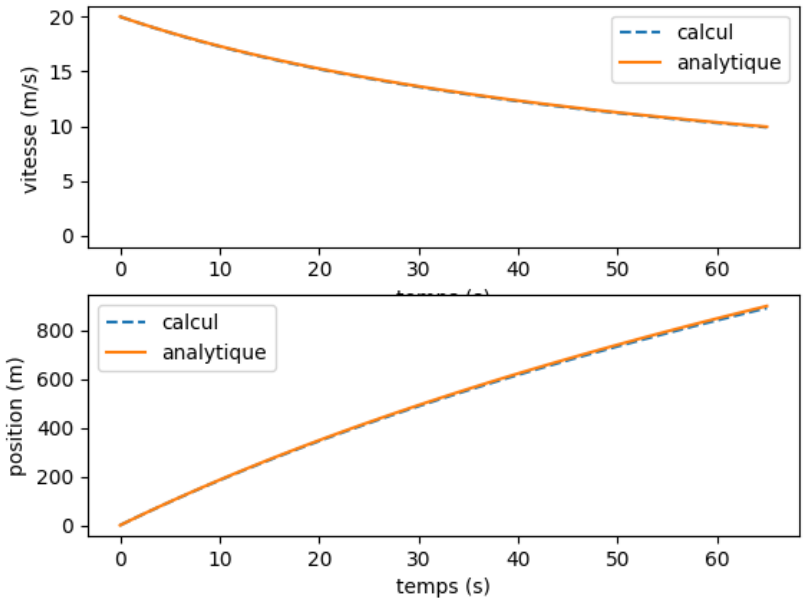



FIGURE 7 – Test de la force de frottement fluide

On mesure ensuite l'erreur en fonction de la précision.

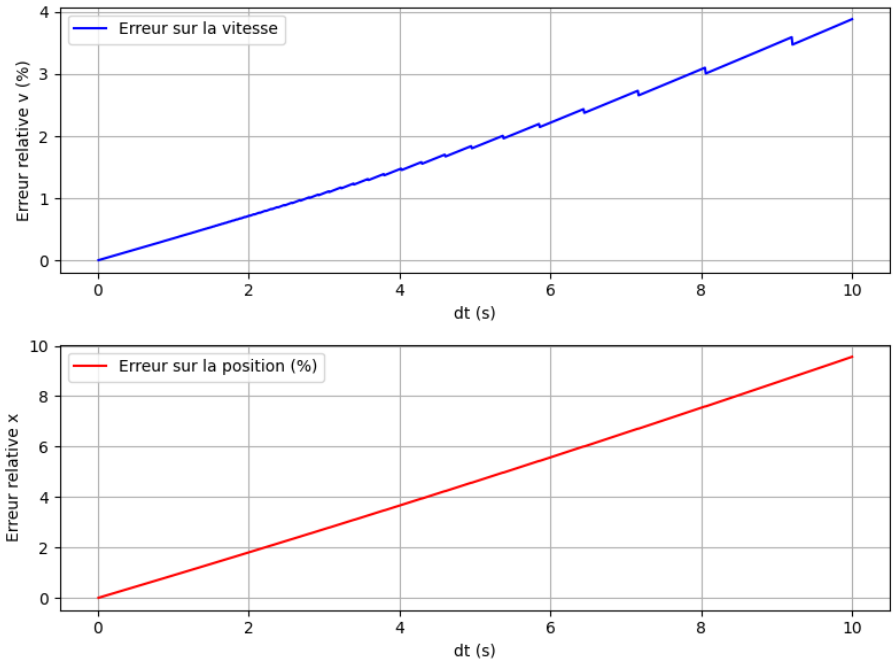



FIGURE 8 – Erreur de la force de frottement fluide en fonction de la précision dt

#### 4.2.4 Force d'inertie

La solution analytique du système est donc :

$$\begin{cases} x(t) = v_0 t \\ \dot{x}(t) = v_0 \end{cases} \tag{16}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse initiale : $v_0 = 200\ \text{m}\,\text{s}^{-1}$

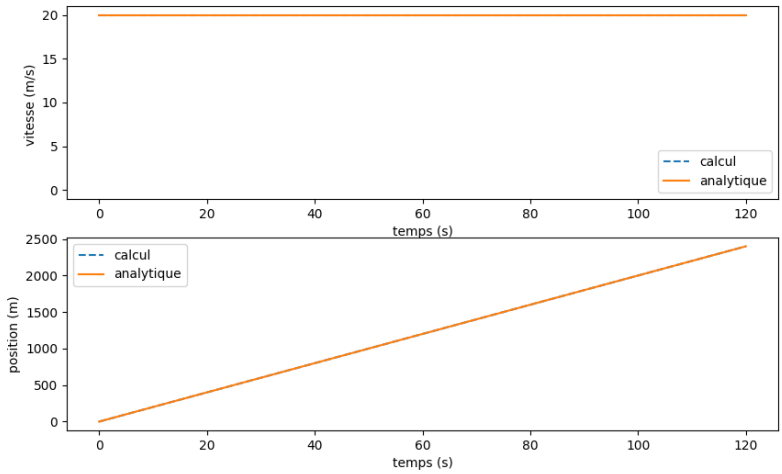



FIGURE 9 – Test de la force d'inertie

On mesure ensuite l'erreur en fonction de la précision.

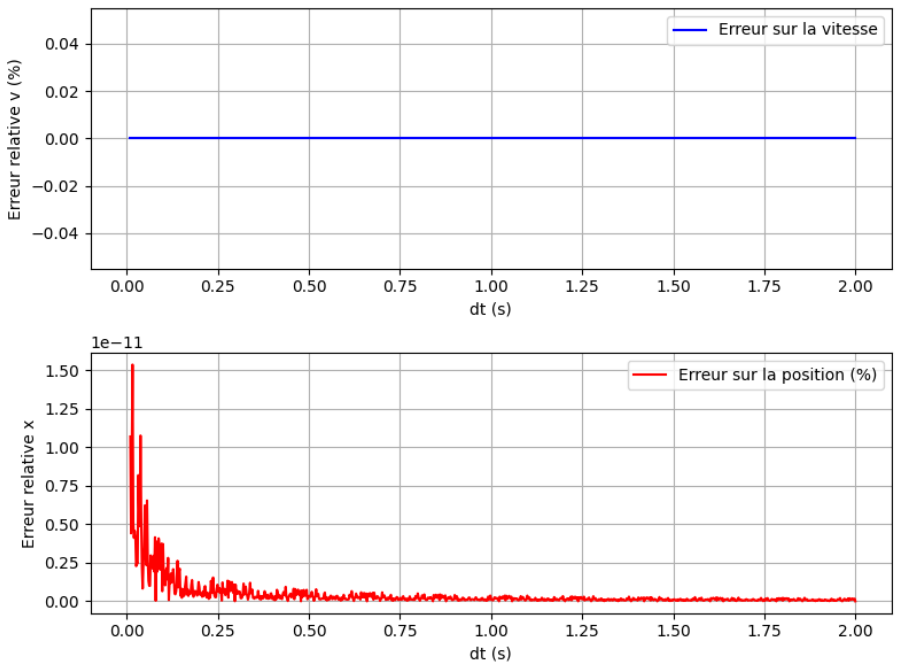



FIGURE 10 – Erreur de la force d'inertie en fonction de la précision dt

### 4.2.5 Force utile

La solution analytique du système est donc :

$$\begin{cases} x(t) = \frac{1}{\alpha} ln \left( \frac{cosh\left(t\sqrt{\alpha g(sin(\theta) - C_r cos(\theta))} - arctanh\left(v_0 \sqrt{\frac{\alpha}{g(sin(\theta) - C_r cos(\theta))}}\right)\right)}{cosh\left(arctanh\left(v_0 \sqrt{\frac{\alpha}{g(sin(\theta) - C_r cos(\theta))}}\right)\right)} \right) \\ \dot{x}(t) = \sqrt{\frac{g(sin(\theta) - C_r cos(\theta))}{\alpha}} tanh\left(t\sqrt{\alpha g(sin(\theta) - C_r cos(\theta))} - arctanh\left(v_0 \sqrt{\frac{\alpha}{g(sin(\theta) - C_r cos(\theta))}}\right)\right) \\ \alpha = \frac{\rho_{air} S_{aero} c_x}{2m} \end{cases} \tag{17}$$

avec ces valeurs et $dt = 1$s :
— vitesse initiale : $v_0 = 0$ m s$^{-1}$
— Surface frontale : $S = 2$ m$^2$
— densité de l'air : $\rho_{air} = 1.293$ kg m$^{-3}$
— coefficient de frottement fluide : $c_x = 0.3$
— masse du véhicule : $m = 500$kg
— accélération terrestre : $g = 9.81$ m s$^{-2}$
— angle de la pente (pente de 5%) : $\theta = \frac{2.86\pi}{180}$ rad
— $(sin(\theta) - C_r cos(\theta)) > 0$

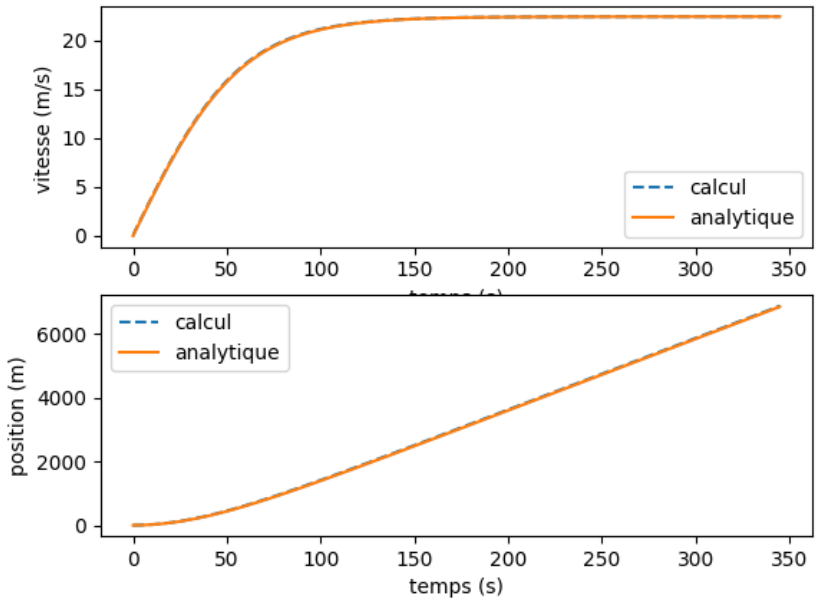



FIGURE 11 – Test avec toute les forces

On mesure ensuite l'erreur en fonction de la précision.

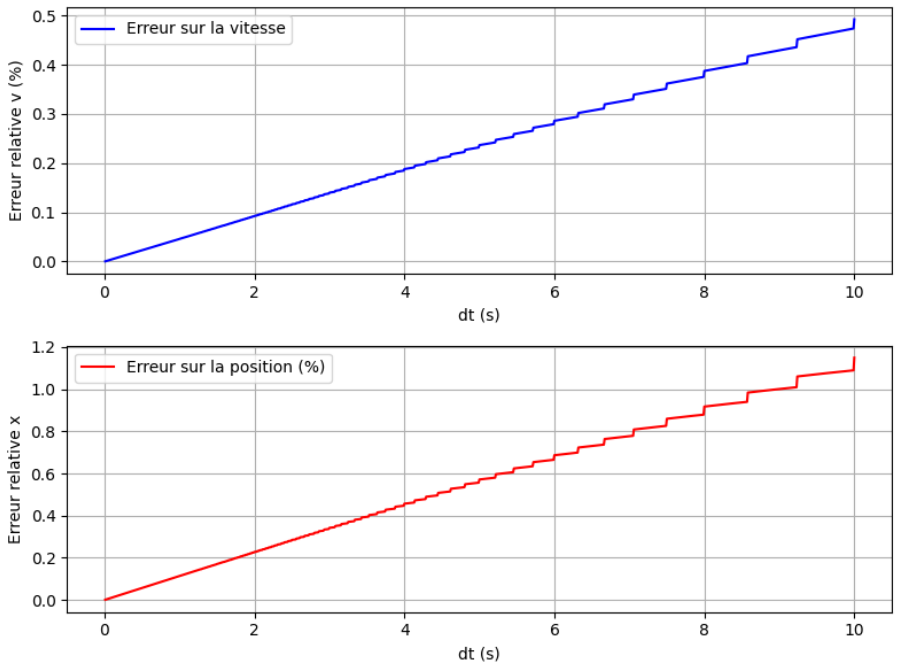



FIGURE 12 – Erreur de la force totale en fonction de la précision dt

#### 4.2.6 Force utile (cas particulier)

La solution analytique du système à l'équilibre est donc :

$$\begin{cases} x(t) = t\sqrt{\frac{g}{\alpha}(sin(\theta) - C_r cos(\theta))} \\ \dot{x}(t) = \sqrt{\frac{g}{\alpha}(sin(\theta) - C_r cos(\theta))} \end{cases} \tag{18}$$

avec ces valeurs et $dt = 1$s :
- vitesse initiale : $v_0 \approx 22.5\ \mathrm{m\,s^{-1}}$
- Surface frontale : $S = 2\ \mathrm{m^2}$
- densité de l'air : $\rho_{air} = 1.293\ \mathrm{kg\,m^{-3}}$
- coefficient de frottement fluide : $c_x = 0.3$
- masse du véhicule : $m = 500$kg
- accélération terrestre : $g = 9.81\ \mathrm{m\,s^{-2}}$
- angle de la pente (pente de 5%) : $\theta = \frac{2.86\pi}{180}$ rad
- $(sin(\theta) - C_r cos(\theta)) > 0$

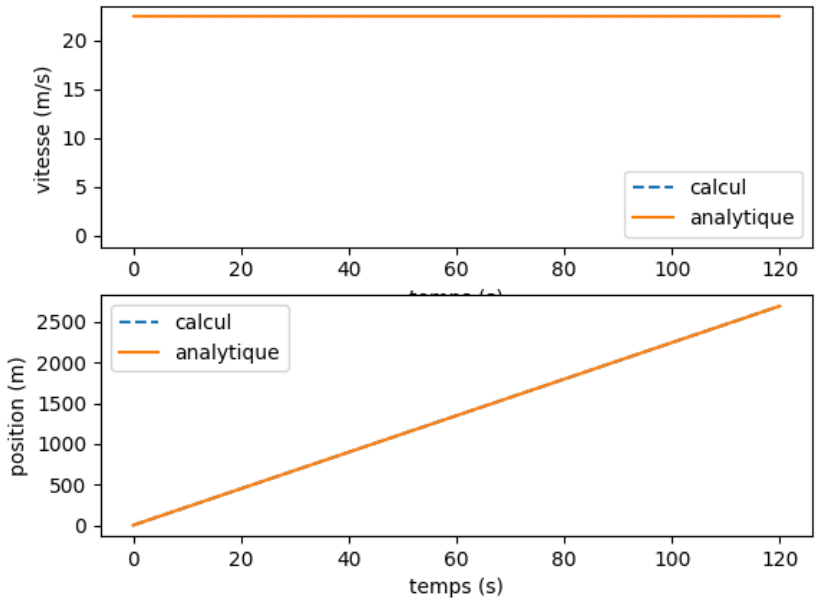



FIGURE 13 – Test de force utile

On mesure ensuite l'erreur en fonction de la précision.

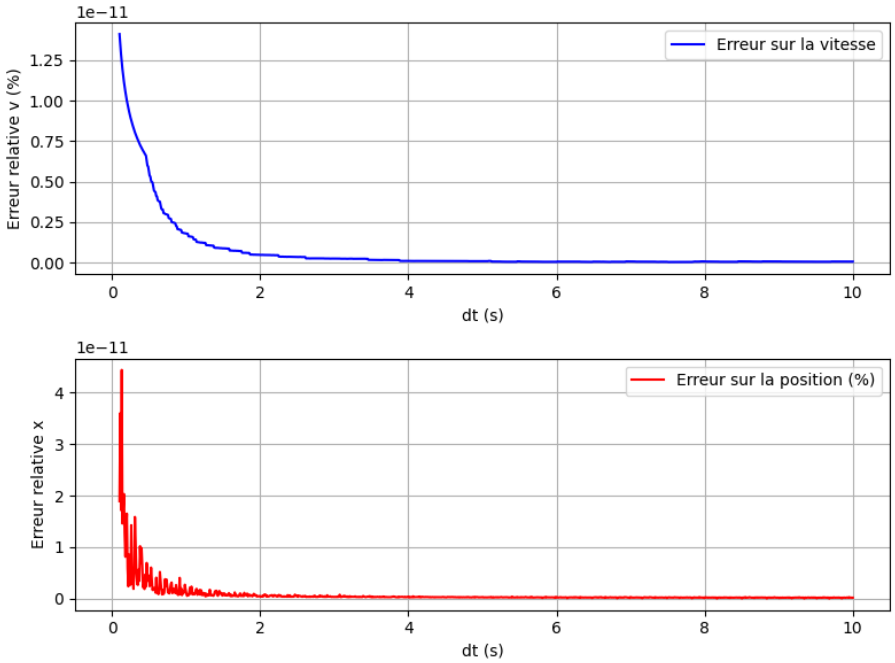
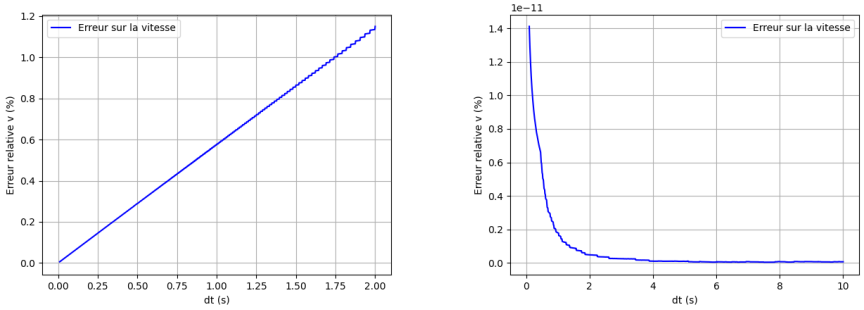



FIGURE 14 – Erreur des forces utiles en fonction de la précision dt

## 4.3 Conclusion

La plus grande source d'erreur provient du terme $F_{aero}$. Erreur qui disparaît lorsque les forces sont à l'équilibre. L'erreur est inférieure à 1%




(a) Test force utile

(b) Test force utile cas particulier

FIGURE 15 – Comparaison des tests de la force utile

## 4.4 Annexe : Utilisation des tests

### 1. Initialisation

— Créer une instance de la classe :

```
mon_test = test_force()
```

— Définir un angle theta de pente en rad.
— Définir la vitesse initiale $v_0$ en $\mathrm{m\,s^{-1}}$.

### 2. Obtenir l'erreur

— Utiliser la fonction :

```
mon_test.test_erreurs(mon_test.test_F_[nom de la force], v0=v_0
    , angle=theta)
```

— Remplacer `[nom de la force]` par la force à tester.
— (Optionnel) Définir le pas de temps minimal et maximal pour le calcul d'erreur :

```
dt_min=<valeur> # Pas de temps minimum en secondes
dt_max=<valeur> # Pas de temps maximum en secondes
```

### 3. Obtenir la vitesse et la trajectoire

— Utiliser la fonction :

```
mon_test.test_affichage(mon_test.test_F_[nom de la force], v0=
    v_0, angle=theta)
```

— (Optionnel) Définir le pas de temps `dt` en s :

```
dt = <valeur>
```

### 4. Exemple d'utilisation

```
mon_vehicule = ppt.eVECTORS()
test = test_force(mon-vehicule)
angle = 2.86 / 180 * np.pi # Conversion en radians
v0 = 20 # Vitesse initiale en m/s

test.test_erreurs(test.test_F_utile, v0=v0, dt_max=10, angle=angle)
test.affichage(test.test_F_utile, dt=1, v0=v0, angle=angle)
```

# 5 Puissance et énergie

L'une des sources de consommation majeure d'énergie du véhicule est de contrer les forces extérieures. Pour cela, on dispose des relevés GPS, de la vitesse instantanée et du temps détaillé (h :min :sec). On pourra donc se concentrer sur le calcul de l'énergie dépensée et donc en déduire l'énergie nécessaire.

## 5.1 équation générale

La puissance à partir de la force :

$$P = \vec{F} \cdot \vec{v} \tag{19}$$

On trouve :

$$\sum \vec{F}_{ext} \cdot \vec{v} = -\frac{1}{2}\rho S_{aero} c_x v^3 - C_{rr} mg\cos(\lambda)v - mg\sin(\lambda)v \tag{20}$$

$$P_{aero} = -\frac{1}{2}\rho S_{aero} c_x v^3 \tag{21}$$

$$P_{roulement} = -C_r mg\cos(\lambda)v \tag{22}$$

$$P_{gravite} = -mg\sin(\lambda)v \tag{23}$$

$$P_{kinetic} = \frac{1}{2} m \frac{d}{dt} v^2 \tag{24}$$

Le but étant de simuler la dépense énergétique et non le mouvement, on calculera la force utile :

$$P_{utile} = P_{kinetic} - (P_{aero} + P_{roulement} + P_{gravite}) \tag{25}$$

et l'énergie :

$$E = \int P dt \tag{26}$$

L'énergie peut aussi s'écrire en fonction du déplacement élémentaire. Pour les forces conservatives :

$$E = \int \vec{F} \cdot \vec{dl} \tag{27}$$

La force de frottement fluide n'étant pas conservative, on passe par l'intégrale de la puissance pour le calcul.

## 5.2 Forme du calcul

Les schémas d'intégration numérique introduisent des erreurs numériques, il est important d'être certain de leur nécessité.

Calculons l'énergie dissipée par la force de frottement fluide des deux manières.

$$\tilde{E}_{aero} = \vec{F_{aero}} \cdot \vec{x} = \frac{1}{2}\rho S_{aero} c_x v^2 x \tag{28}$$

$$E_{aero} = \int \vec{F}_{aero} \cdot \vec{v} dt = \int \frac{1}{2}\rho S_{aero} c_x v^3 dt \tag{29}$$

En choisissant $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$, il vient $x(t) = \frac{4v_0 t^2}{t_f}\left(\frac{1}{2} - \frac{t}{3t_f}\right)$.

On obtient :

$$\tilde{E}_{aero} = \frac{1}{2}\rho S_{aero} c_x \left(\frac{4v_0}{t_f}\right)^3 t^4 \left(1 - \frac{t}{t_f}\right)^2 \left(\frac{1}{2} - \frac{t}{3t_f}\right) \tag{30}$$

$$E_{aero} = \frac{1}{2}\rho S_{aero} c_x \left(\frac{4v_0}{t_f}\right)^3 t^4 \left(\frac{1}{4} - \frac{3t}{5t_f} + \frac{3t^2}{6t_f^2} - \frac{t^3}{7t_f^3}\right) \tag{31}$$

On constate $\tilde{E}_{aero} \neq E_{aero}$.
On doit donc en passer par le calcul de l'intégrale.

## 5.3 test et validation du programme des forces (module_energie.Power())

Les forces étant des forces résistantes, on testera ce système :

$$\begin{cases} E(t) = \int P(v(t), \theta(t), t) dt \\ E(0) = 0 \end{cases} \tag{32}$$

Comme précédemment, on utilisera un schéma d'Euler pour les calculs numériques.

$$\begin{cases} E_{n+1} = E_n + P_n dt \\ E_0 = E(0) \end{cases} \tag{33}$$

## 5.4 Modèle pour la vitesse

On testera 3 modèles différents de vitesse pour valider les calculs :
— vitesse constante $v(t) = v_0$
— vitesse dépendant du temps au carré $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$
— vitesse obtenue dans l'équation (17)

## 5.5 Modèle pour $\theta(t)$

On testera 2 modèles différents d'angle pour valider les calculs :
— Angle constant : $\theta(t) = \theta_0$
— Angle linéaire en temps $\theta(t) = \theta \frac{t}{t_f}$

## 5.6 Énergie dû à la pente

On calcule avec $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$ et $\theta(t) = \omega t$. Ce qui aboutit à :

$$E_{pente} = -mgv_0 \cdot \frac{4t_f}{\omega^3}\left(\left(\left(\frac{\omega}{t_f}\right)^2 t(t - t_f) - 2\right)\cos\left(\frac{\omega t}{t_f}\right) + \frac{\omega}{t_f}(t_f - 2t)\sin\left(\frac{\omega t}{t_f}\right) + 2\right) \tag{34}$$

avec ces valeurs et $dt = 1\text{s}$ :

— vitesse maximale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$

— masse du véhicule : $m = 500\text{kg}$

— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$

— angle de la pente : $\theta = \frac{\pi}{6}$ rad

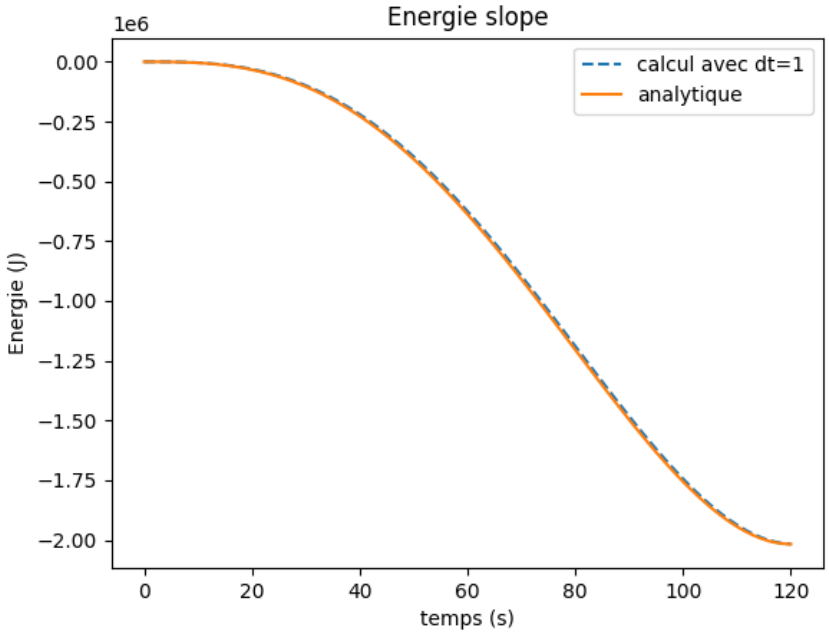



FIGURE 16 – Énergie dû à la pente

Avec l'erreur :

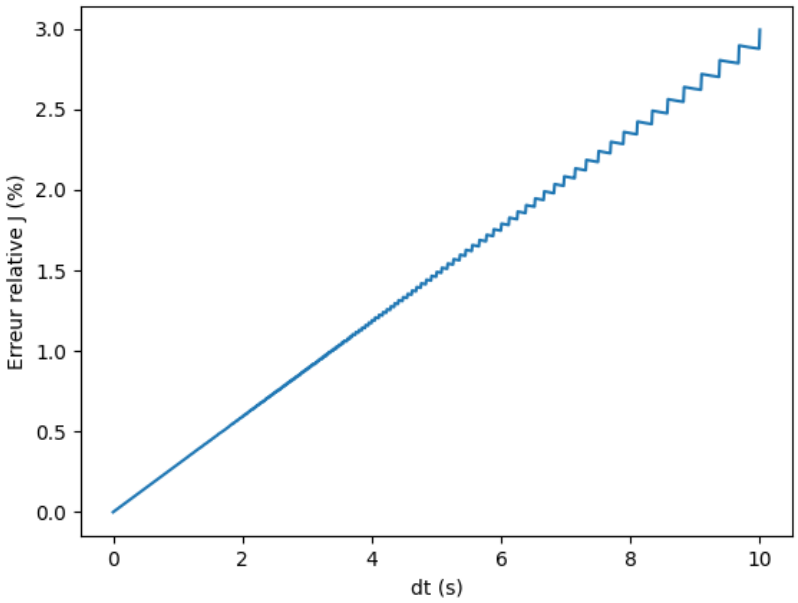



FIGURE 17 – erreur de l'énergie dû au roulement

## 5.7 Énergie dû au roulement

On calcule avec $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$ et $\theta(t) = \omega t$. Ce qui aboutit à :

$$E_{roulement} = -C_r m g v_0 \cdot \frac{4t_f}{\omega^3}\left(\left(\left(\frac{\omega}{t_f}\right)^2 t(t_f - t) + 2\right)\sin\left(\frac{\omega t}{t_f}\right) + \frac{\omega}{t_f}(t_f - 2t)\cos\left(\frac{\omega t}{t_f}\right) - \omega\right) \tag{35}$$

avec ces valeurs et $dt = 1\text{s}$ :

— vitesse maximale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— coefficient de résistance au roulement : $C_r = 0.01$
— masse du véhicule : $m = 500\text{kg}$
— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$
— angle de la pente : $\theta = \frac{\pi}{6}$ rad

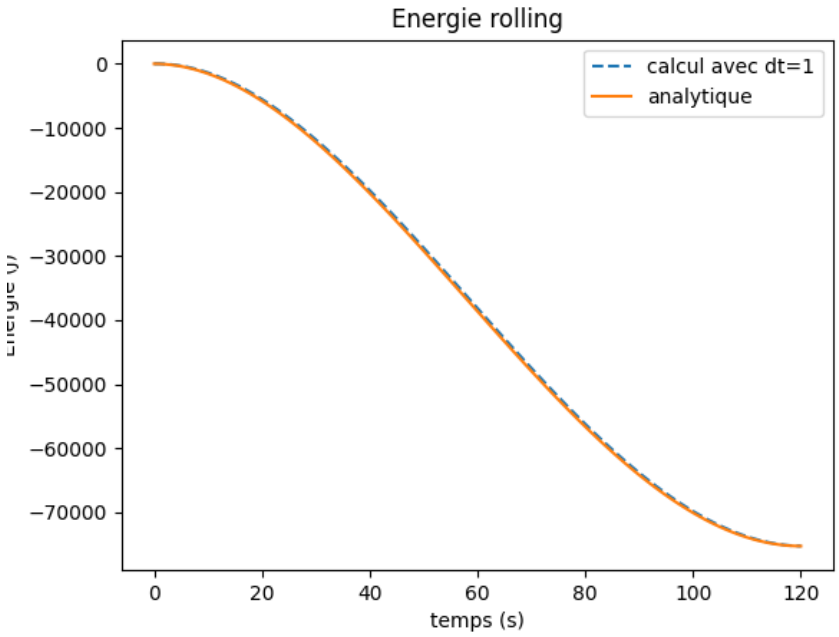



FIGURE 18 – Énergie dû au roulement

Avec l'erreur :

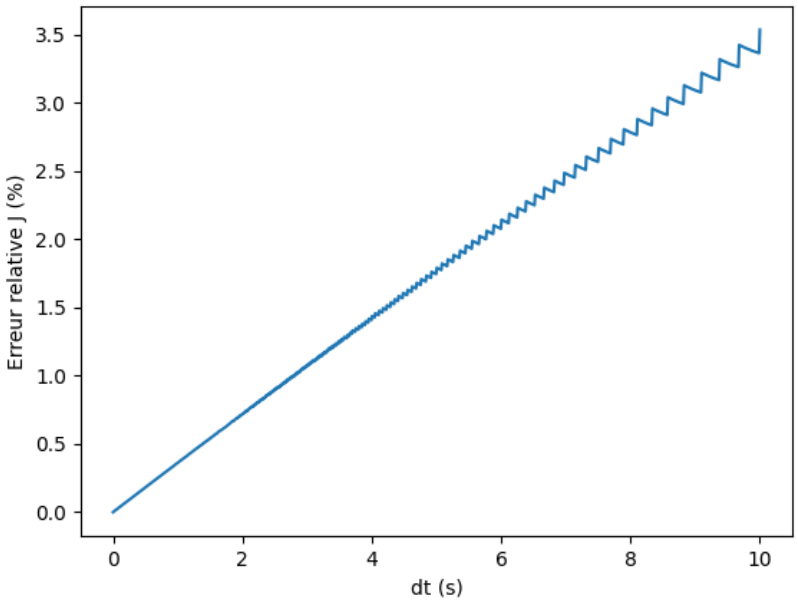



FIGURE 19 – erreur de l'énergie dû au roulement

## 5.8 Énergie dû au frottement visqueux

On calcule avec $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$ et $\theta(t) = \omega t$. Ce qui aboutit à :

$$E_{aero} = \frac{\rho_{air} S_{aero} c_x}{2} \frac{64 v_0^3 t^4}{t_f^3}\left(\frac{1}{4} - \frac{3t}{5t_f} + \frac{t^2}{2t_f^2} - \frac{t^3}{7t_f^3}\right) \tag{36}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse maximale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— Surface frontale : $S = 2\ \text{m}^2$
— densité de l'air : $\rho_{air} = 1.293\ \text{kg}\,\text{m}^{-3}$
— coefficient de frottement fluide : $c_x = 0.3$
— masse du véhicule : $m = 500\text{kg}$

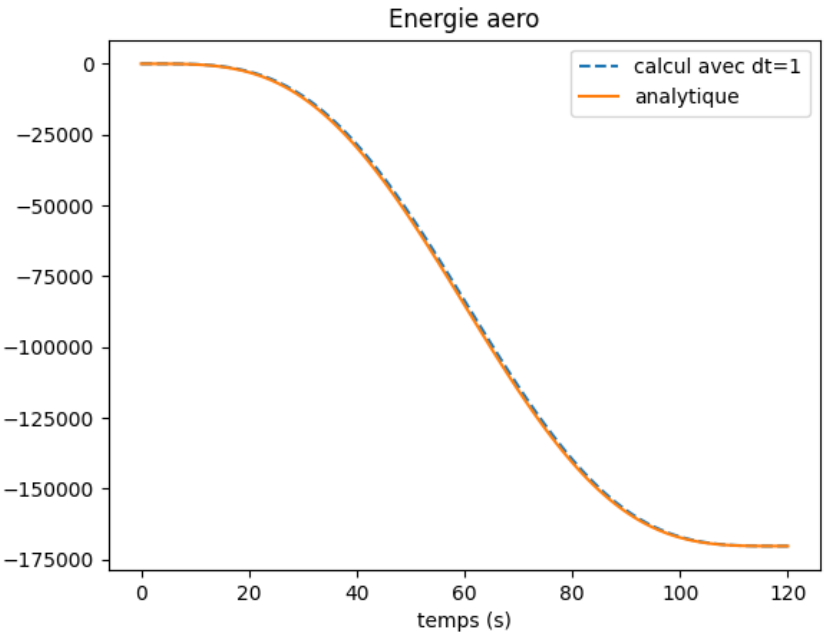



FIGURE 20 – Énergie dû au frottement visqueux

Avec l'erreur :

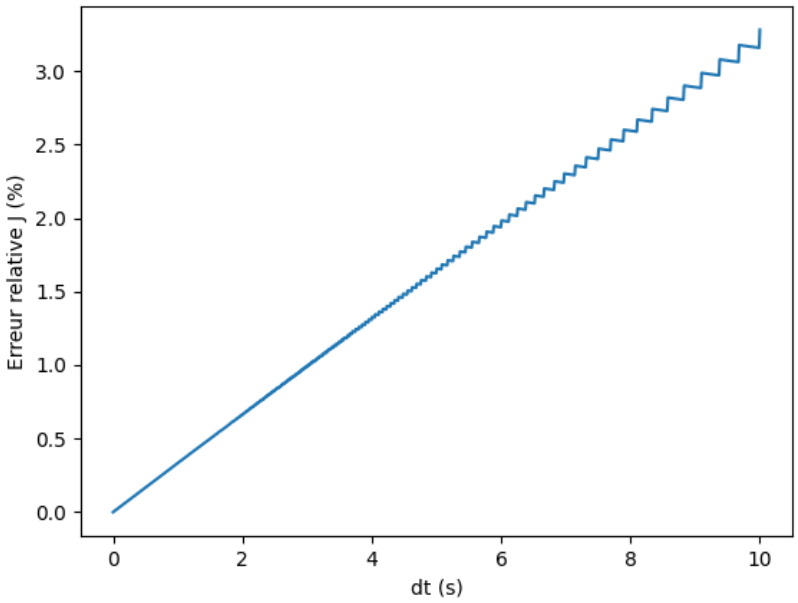


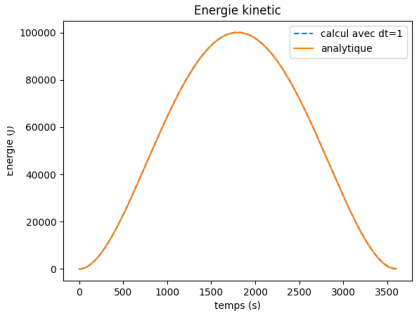

FIGURE 21 – erreur de l'énergie dû au frottement visqueux

## 5.9 Énergie dû à l'inertie

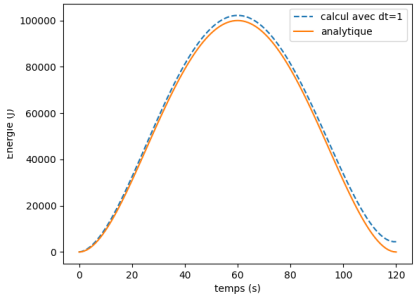
On calcule avec $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$ et $\theta(t) = \omega t$. Ce qui aboutit à :

$$E_{kinetic} = \frac{1}{2}mv^2 = 8v_0^2\frac{t^2}{t_f^2}\left(1 - \frac{t}{t_f} + \frac{t^2}{t_f^2}\right) \tag{37}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse maximale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— masse du véhicule : $m = 500\text{kg}$
— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$
— angle de la pente : $\theta = \frac{\pi}{6}$ rad



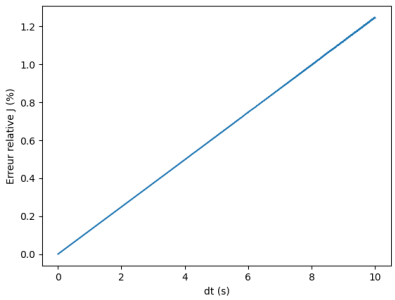

(a) Énergie dû à l'inertie



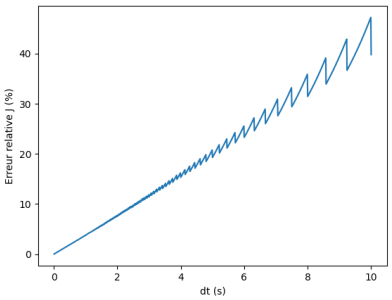

(b) Énergie dû à l'inertie

FIGURE 22 – Comparaison de l'énergie dû à l'inertie

Avec l'erreur :




(a) erreur de l'énergie dû à l'inertie




(b) erreur de l'énergie dû à l'inertie

FIGURE 23 – Comparaison de l'énergie dû à l'inertie

On observe que l'erreur diminue avec le temps. C'est une erreur d'interprétation, l'erreur augmente avec la variation de la fonction. La fonction variant fortement sur l'intervalle de temps [0 ;120] l'erreur est plus importante.

## 5.10 Énergie totale

On calcule avec $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$ et $\theta(t) = \omega t$. Ce qui aboutit à :

$$E_{utile} = E_{kinetic} - E_{aero} - E_{pente} - E_{roulement} \tag{38}$$

avec ces valeurs et $dt = 1\text{s}$ :
— vitesse maximale : $v_0 = 20\ \text{m}\,\text{s}^{-1}$
— Surface frontale : $S = 2\ \text{m}^2$
— densité de l'air : $\rho_{air} = 1.293\ \text{kg}\,\text{m}^{-3}$

— coefficient de frottement fluide : $c_x = 0.3$
— masse du véhicule : $m = 500\text{kg}$
— coefficient de résistance au roulement : $C_r = 0.01$
— accélération terrestre : $g = 9.81\ \text{m}\,\text{s}^{-2}$
— angle de la pente : $\theta = \frac{\pi}{6}$ rad

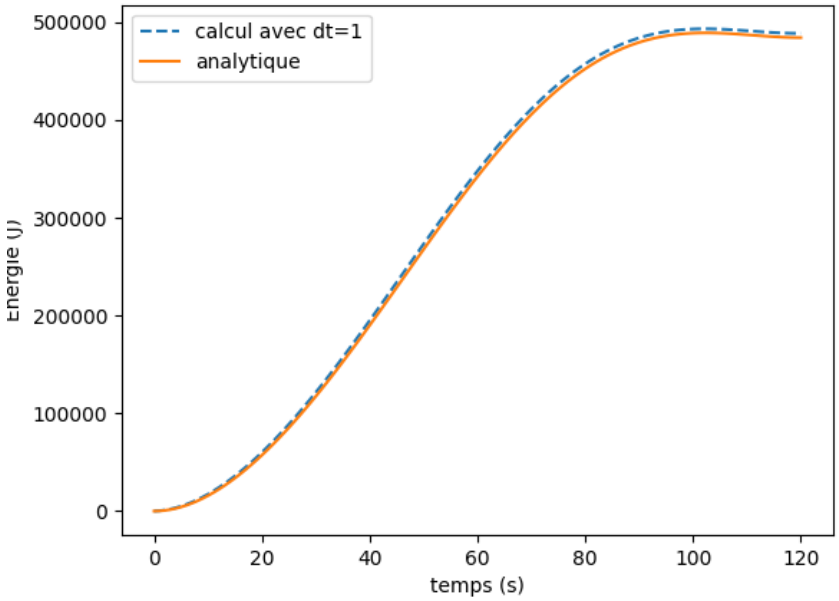



FIGURE 24 – Énergie totale

Avec l'erreur :

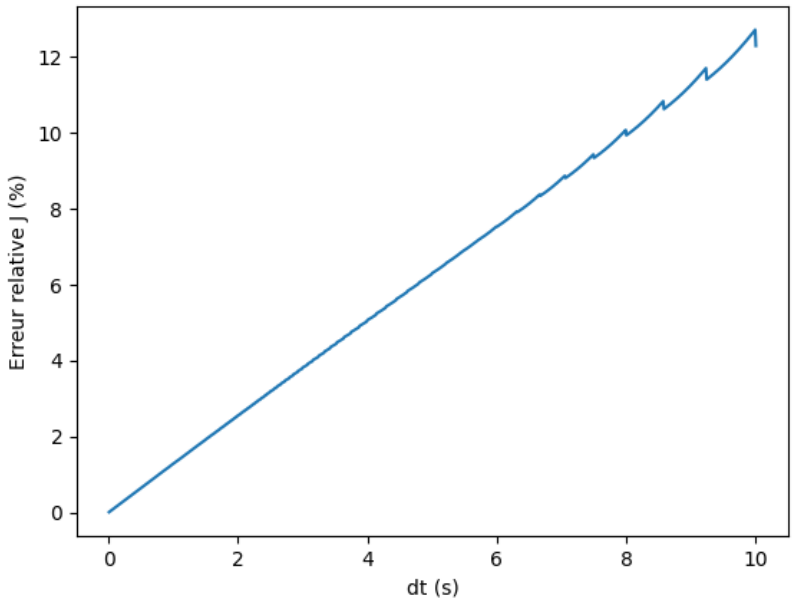



FIGURE 25 – erreur de l'énergie totale

## 5.11 Conclusion

L'erreur est linéaire et inférieure à 2% pour $dt = 1$.

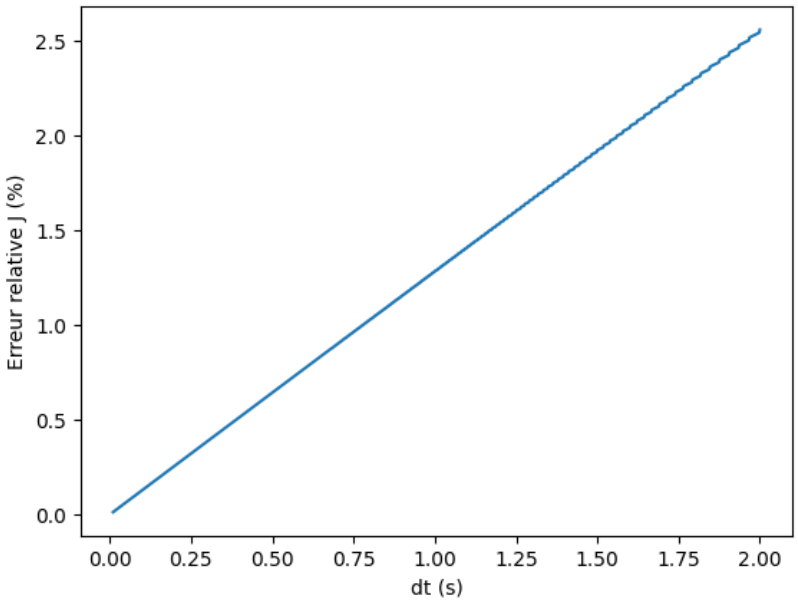



FIGURE 26 – erreur de l'énergie utile

## 5.12 Annexe : Utilisation des tests

### 1. Initialisation

— Créer une instance de la classe :

```
mon_test = test_energie()
```

— Définir les constantes du système :

```
angle = np.pi/6
tf = 120
dt = 1
v0 = 20
```

— Définir quelle fonction de vitesse $nv$ choisir :
  — $v(t) = v_0$, $nv = 1$
  — $v(t) = \frac{4v_0 t}{t_f}\left(1 - \frac{t}{t_f}\right)$, $nv = 2$

— Définir quelle fonction d'angle $ntheta$ choisir :
  — $\theta(t) = \theta_0$, $ntheta = 1$
  — $\theta(t) = \begin{cases} \theta_0, & \text{si } |t - t_f| \leq \frac{t_f}{6} \\ 0, & \text{sinon} \end{cases}$, $ntheta = 2$
  — $\theta(t) = \theta_0 \frac{t}{t_f}$

— Définir quelle puissance $nfct$ calculer :
  — $P_{aero}$, $nfct = 1$
  — $P_{roulement}$, $nfct = 2$
  — $P_{pente}$, $nfct = 3$
  — $P_{utile}$, $nfct = 4$

**2. Obtenir l'erreur**

— Utiliser la fonction :

```
mon_test.test_erreur(nv,ntheta,nfct,v0=v0,angle=angle,tf=tf,
    dt_min=0.001, dt_max=2)
```

— (Optionnel) Définir le pas de temps minimal et maximal pour le calcul d'erreur :

```
dt_min=<valeur> # Pas de temps minimum en secondes
dt_max=<valeur> # Pas de temps maximum en secondes
```

**3. Obtenir l'énergie en fonction du temps**

— Utiliser la fonction :

```
mon_test.affichage_energie(nv,ntheta,tf=tf,dt=dt,v0=v0,angle=
    angle)
plt.show()
```

— ou bien pour choisir quelle fonction afficher :

```
test.affichage(nv,ntheta,nfct,tf=300,dt=1,v0=20,angle=angle)
plt.show()
```

**4. Exemple d'utilisation**

```
# exemple test energie
mon_vehicule = ppt.eVECTORS()
test = test_energie(mon_vehicule)
# nv,ntheta,nfct
nv,ntheta,nfct = 2,3,4
angle = np.pi/6
tf = 120
dt = 1
v0 = 20
test.affichage_energie(nv,ntheta,tf=tf,dt=dt,v0=v0,angle=angle)
plt.show()

for nfct in [1,2,3,4]:
    test.test_erreur(nv,ntheta,nfct,v0=v0,angle=angle,tf=tf,dt_min=0.001,
        dt_max=2)
```

```
15      print(f"fonction␣{nfct}␣finie␣sur␣4")
16      plt.xlabel("dt␣(s)")
17      plt.ylabel("Erreur␣relative␣J␣(%)")
18      plt.show()
```

# 6 Température et chauffage

La consommation des systèmes auxiliaires n'est pas négligeable. Il faut donc les simuler de façon réaliste.
Pour la puissance nécessaire au chauffage, on considère la puissance dissipée vers l'extérieur et la puissance de rayonnement reçue.
on explore plusieurs modélisations pour en garder une réaliste.
on modélise l'évolution de la température dans l'habitacle par :

$$\begin{cases} m_{air}c_p \dfrac{dT_{\text{habitacle}}(t)}{dt} = P_{perdu} + P_{\text{rayonnement}} + P_{\text{chauffage}} \\ T(0) = T_{\text{ext}} \end{cases} \tag{39}$$

avec :

$$P_{perdu} = -k_{thermique}(T_{\text{habitacle}} - T_{ext}) \tag{40}$$

$$P_{\text{rayonnement}} = I_{soleil} S_{vitre} \alpha_{vitre} \tag{41}$$

## 6.1 Modèle 1 : Puissance constante

On considère $P_{\text{chauffage}}$ constante, ce qui amène au résultat :

$$T_{\text{habitacle}}(t) = T_{\text{ext}} + \left(\frac{P_{\text{rayonnement}} + P_{\text{chauffage}}}{k_{thermique}}\right)\left(1 - exp\left(-\frac{k_{thermique}}{m_{air}c_p}t\right)\right) \tag{42}$$

La puissance nécessaire pour atteindre une température $T_{cible}$ est :

$$P_{necessaire} = k_{thermique} S_{tot}(T_{cible} - T_{\text{ext}}) - P_{\text{rayonnement}} \tag{43}$$

## 6.2 Modèle 2 : Puissance dépendant de la température

On considère $P_{\text{chauffage}}$ comme dépendant de la température , comme suit :

$$P_{\text{chauffage}} = \dot{m}_{air} c_p (T_{air} - T_{\text{habitacle}}) \tag{44}$$

On calcule la solution stationnaire :

$$P_{stationnaire} = \frac{\dot{m}_{air}c_p}{k_{thermique}S_{tot} + \dot{m}_{air}c_p}\left(k_{thermique}S_{tot}(T_{air} - T_{\text{ext}}) - P_{\text{rayonnement}}\right) \tag{45}$$

Puis la solution instationnaire :

$$T_{\text{habitacle}}(t) = T_{air} - \frac{P_{stationnaire}}{\dot{m}_{air}c_p} + \left(T_{\text{ext}} - T_{air} + \frac{P_{stationnaire}}{\dot{m}_{air}c_p}\right) exp\left(\frac{-(k_{thermique} + \dot{m}_{air}c_p)}{m_{air}c_p}t\right) \tag{46}$$

Avec (44) et en multipliant par le temps d'utilisation, on en déduit l'énergie utilisée :

$$E_{chauffage}(t) = \dot{m}_{air}c_p\left(\frac{k_{thermique}(T_{air} - T_{\text{ext}}) - P_{\text{rayonnement}}}{k_{thermique} + \dot{m}_{air}c_p}\right)t \tag{47}$$

La formule est aussi valable pour $T_{air} < T_{\text{habitacle}}$ donc exprime la puissance de la climatisation.

## 6.3 Comparaison

En prenant des grandeurs de la littérature afin de pouvoir comparer ces deux modèles, on utilise les valeurs suivantes :

— Volume de l'habitacle : $V_{\text{voiture}} = 2\ \text{m}^3$
— Masse volumique de l'air : $\rho_{\text{air}} = 1.293\ \text{kg}\,\text{m}^{-3}$
— Débit massique d'air : $\dot{m} = 0.05\ \text{kg}\,\text{s}^{-1}$
— Capacité thermique massique de l'air : $C_p = 1000\ \text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$
— Coefficient de pertes thermiques : $k_{thermique} = 10\ \text{W}\,\text{K}^{-1}$
— Surface totale d'échange thermique : $S_{tot} = 5\ \text{m}^2$
— Intensité du rayonnement solaire : $I_{soleil} = 0\ \text{W}\,\text{m}^{-2}$
— Surface des vitres : $S_{\text{vitre}} = 2\ \text{m}^2$
— Facteur d'absorption des vitres : $\alpha_{\text{vitre}} = 0.6$
— Température extérieur : $T_{ext} = 273\ \text{K}$

On peut alors comparer ces deux modèles en traçant la figure suivante :

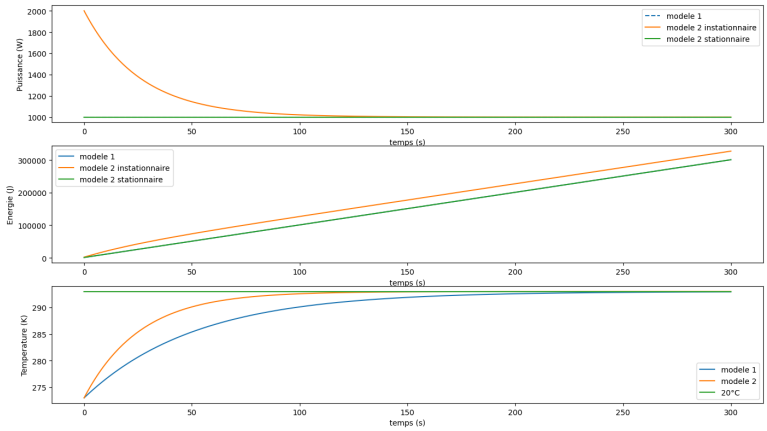



FIGURE 27 – Comparaison des modèles

# 7 Données GPS et traitement

Afin de calculer la dépense énergétique du véhicule, on récupère les données GPS, la vitesse et le temps du trajet à différents points. Dans un premier temps, ces données sont récupérées au format fichier ".GPX" qui contient la position GPS, la vitesse en $\mathrm{m\,s^{-1}}$ et le temps au format h :min :sec.

## 7.1 traitement fichier GPS

On veut calculer la distance parcourue et la pente. Pour cela, on doit convertir les données GPS en position xyz. On utilise la formule de Haversine pour obtenir la distance horizontale :

$$d_{xy} = 2R_{Terre} \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \tag{48}$$

avec :

- latitude du point 1 et 2 : $\varphi_1$, $\varphi_2$ en rad
- longitude du point 1 et 2 : $\lambda_1$, $\lambda_2$ en rad
- rayon Terrestre : $R_{Terre}$ en m
- distance horizontale : $d_{xy}$ en m

On calcule ensuite la pente :

$$\theta = arctan\left(\frac{\Delta h}{d_{xy}}\right) \tag{49}$$

avec :

- angle de la pente : $\theta$ en rad
- distance horizontale : $d_{xy}$ en m
- variation d'altitude : $\Delta h$ en m

la distance totale est la somme des $d_{xy}$

## 7.2 Transformation des coordonnées GPS en repère local ENU

La fonction `get_ENUt_coordinates` permet de convertir des coordonnées GPS exprimées en latitude, longitude et altitude (système géodésique WGS84) en un repère cartésien local **ENU** (East-North-Up), centré sur le premier point mesuré du trajet.

- La première étape consiste à définir un point de référence, ici choisi comme le premier point de la trajectoire, noté $(\phi_0, \lambda_0, h_0)$.
- On utilise ensuite une transformation de coordonnées fournie par la bibliothèque `pyproj` pour convertir toutes les coordonnées GPS en coordonnées cartésiennes géocentriques (ECEF : Earth-Centered Earth-Fixed).

— Les coordonnées du point de référence sont également converties en ECEF, puis soustraites à toutes les autres coordonnées afin d'obtenir des vecteurs position relatifs.
— Enfin, ces vecteurs relatifs sont projetés dans le repère local ENU grâce à la matrice de rotation classique définie par la latitude et la longitude du point de référence. Les composantes calculées sont :
  — $x$ : direction Est (East),
  — $y$ : direction Nord (North),
  — $z$ : direction haut (Up).
— La fonction retourne les coordonnées locales $(x, y, z)$ ainsi que le temps associé à chaque point.

Cette transformation n'est valable que pour de petits trajets en raison de la forme elliptique et non sphérique de la Terre.

## 7.3 pas de temps

On s'intéresse au pas de temps pour pouvoir correctement intégrer ces données. Le temps est donné au format hh :mm :ss. On récupère donc le pas de temps en faisant la différence des données converties.

$$dt_i = heure_{i+1} * 3600 + minute_{i+1} * 60 + seconde_{i+1} - (heure_i * 3600 + minute_i * 60 + seconde_i) \tag{50}$$

## 7.4 Structure des données

Le script ""module_traitement_trajet.py" prend un fichier GPX contenant les coordonnées GPS, la vitesse et le temps. Le script renvoie un dictionnaire avec ces clés et valeurs (attention à l'orthographe et aux majuscules) :

| key | value | type |
|---|---|---|
| Angles | give the angle of the slope | np.array |
| Distance totale | total distances for time | np.array |
| Vitesse | speed at each time | np.array |
| Altitude | heigh at each time | np.array |
| dt | time step for each time | np.array |
| Latitude | Latitude | np.array |
| Longitude | Longitude | np.array |
| Altitude | Altitude | np.array |
| x | position selon l'axe x initialisé pour $x_0 = 0$ | np.array |
| y | position selon l'axe x initialisé pour $y_0 = 0$ | np.array |
| z | position selon l'axe z | np.array |
| Time | temps du trajet | np.array |

TABLE 2 – key and values

## 7.5 Données trajet

Une fois traitées, les données peuvent être affichées. Ce qui donne pour un trajet test :

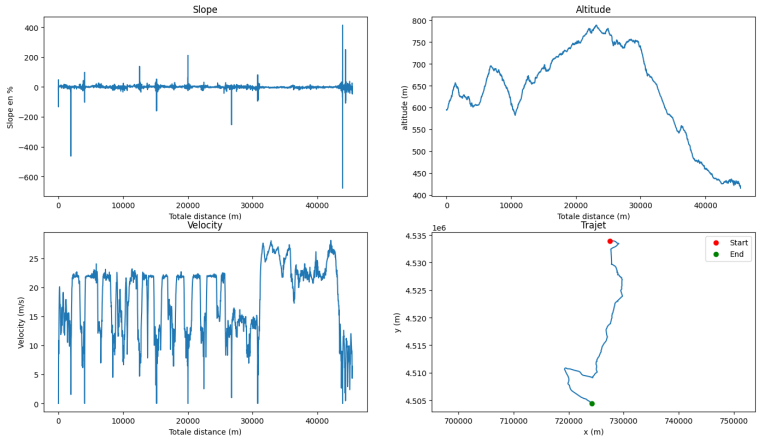



FIGURE 28 – exemple de trajet

Il est aussi possible d'afficher le trajet en 3 dimensions.

## 7.6 Énergie sur un trajet

Grâce au traitement de trajet et au module d'énergie, on peut calculer la dépense sur un trajet. Avec le même trajet que précédemment :

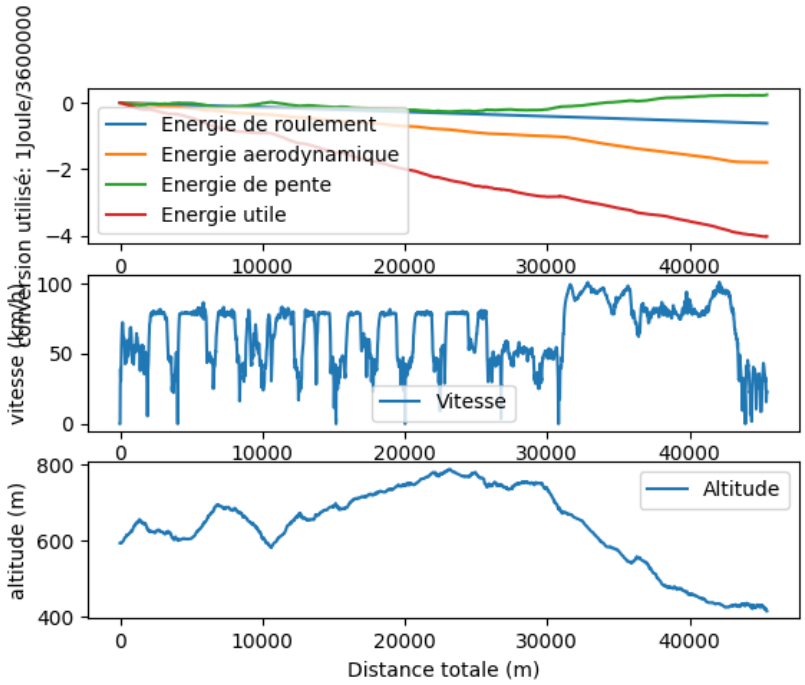



FIGURE 29 – Énergie sur le trajet

L'énergie ici est en kWh, on ne considère que l'énergie utile.

## 7.7 Annexe : Utilisation du module

Récupération des données traitées :

```
fichier = 'mon_trajet.gpx'
test = trt_xyz(fichier)
test.traitement_final()
donnees = test.donnees
```

ou bien :

```
fichier = 'mon_trajet.gpx'
test = trt_xyz(fichier)
donnees = test.traitement_final()
```

## 7.8 Annexe : Utilisation du test

L'utilisation est très simple : définir le fichier trajet, choisir (ou non) les caractéristiques du véhicule, choisir une conversion d'énergie. Exemple :

```
fichier = 'mon_trajet.gpx'
eVECTORS = ppt.eVECTORS()
# option :
eVECTORS.masse = 500
eVECTORS.Cx = 0.3
eVECTORS.Cr = 0.01
eVECTORS.S = 2
eVECTORS.radius = 0.3
eVECTORS.nbr = 4
# fin option
mon_test = test_GPS_Energie(fichier,eVECTORS)
mon_test.affichage(cv.joule_kWh)
plt.show()
```

# 8 Moteur

## 8.1 Rendement

Les systèmes n'étant pas parfaits, on introduit des rendements afin de simuler ces pertes. Chaque composant de la chaîne énergétique, de la batterie jusqu'à la roue, a un rendement propre que nous détaillons ci-dessous.

### 8.1.1 Batterie

Le rendement de la batterie, noté $\eta_{batterie}$.La modélisation de la batterie peut se faire relativement simplement avec un modèle de résistance interne, avec des pertes par effet Joule proportionnelles à $R_{batterie}I^2_{batterie}$ avec $R_{batterie}$ la résistance interne et $I_{batterie}$ la tension fournie par le moteur.
En première approche, un rendement moyen de 95% peut être pris en compte tant pour la recharge que pour la décharge. (issu du PV Revue d'avancement eVECTORS 3)
On calcule $I$ avec la puissance du moteur et la tension de la batterie :

$$I_{batterie}(P_{moteur}, U_{batterie}) = \frac{P_{moteur}}{U_{batterie}} = \frac{T_{moteur}\omega_{moteur}}{U_{batterie}} \tag{51}$$

Et le rendement de la batterie :

$$\eta_{batterie}(P_{moteur}, U_{batterie}) = \frac{P_{moteur}}{P_{moteur} + R_{batterie}I^2_{batterie}} \tag{52}$$

On calcule la puissance du moteur comme suit :

$$P_{moteur} = T_{moteur}\omega_{moteur} \tag{53}$$

Finalement :

$$\eta_{batterie}(T_{moteur}, \omega_{moteur}, U_{batterie}) = \frac{U^2_{batterie}}{U^2_{batterie} + R_{batterie}T_{moteur}\omega_{moteur}} \tag{54}$$

### 8.1.2 Onduleur

Le rendement de l'onduleur, $\eta_{ondulateur}$, est très élevé et quasiment constant."Pour l'efficacité de l'onduleur, une efficacité moyenne de 96% est suggérée par Anthony" (issu du PV Revue d'avancement eVECTORS 3).

### 8.1.3 Moteur

Le rendement d'un moteur électrique dépend du couple appliqué $T$ et de sa vitesse de rotation $\omega$. Il varie significativement selon ces deux paramètres.

On utilise les valeurs fournies par le constructeur. On interpole les données manquantes en fonction de $\omega_{moteur}$ et $T_{moteur}$.

$$\begin{cases} f_\eta(T,\omega) = aT + b\omega + c \\ \qquad a = \dfrac{\eta(T_1,\omega_0) - \eta(T_0,\omega_0)}{T_1 - T_0} \\ \qquad b = \dfrac{\eta(T_0,\omega_1) - \eta(T_0,\omega_0)}{\omega_1 - \omega_0} \\ \qquad c = \eta(T_0,\omega_0) - aT_0 - b\omega_0 \end{cases} \tag{55}$$

Avec $f_\eta(T,\omega)$ la fonction d'interpolation locale.

Afin de calculer le rendement, il est donc nécessaire de calculer $\omega_{moteur}$ et $T_{moteur}$. On modélise :

$$\omega_{moteur}(v(t)) = \frac{v(t)}{R} r_{ratio} \tag{56}$$

avec $v(t)$ la vitesse du véhicule, $R$ le rayon des roues et $r_{ratio}$ le rapport de transmission.
Puis le couple :

$$T_{moteur}(F_{utile}(v(t)), v(t)) = F_{traction}\frac{R}{r_{ratio}} = -F_{utile}\frac{R}{r_{ratio}} \tag{57}$$

### 8.1.4 Réducteur

Le réducteur mécanique possède un rendement élevé, généralement supérieur à 95%. Ce rendement $\eta_{reducteur}$ est essentiellement constant, bien que de très légères pertes supplémentaires puissent apparaître à très faible couple.

### 8.1.5 Transmission

Le rendement de la transmission finale (pont, différentiel, cardans) est également élevé et très peu variable. On le modélise par une constante $\eta_{transmission}$ typiquement autour de 0,98.

### 8.1.6 Rendement moteur

Le rendement du moteur fait intervenir les éléments entre la roue et le moteur à savoir : la transmission et le réducteur. Le rendement s'écrit donc :

$$\eta(T,\omega) = \eta_{moteur}(T,\omega) \cdot \eta_{reducteur} \cdot \eta_{transmission} \tag{58}$$

### 8.1.7 Rendement total

On modélise l'énergie allant de la batterie jusqu'à la roue. Le rendement total de la chaîne énergétique est donc le produit des rendements individuels :

$$\eta_{total}(T,\omega,U) = \eta_{batterie}(T,\omega,U)\cdot\eta_{ondulateur}\cdot\eta_{moteur}(T,\omega)\cdot\eta_{reducteur}\cdot\eta_{transmission} \tag{59}$$

Ce rendement global permet de déterminer la proportion de l'énergie électrique effectivement transmise aux roues pour produire du mouvement.

## 8.2 Énergie nécessaire

L'énergie nécessaire devient donc :

$$E_{utile} = \int \frac{F_{utile} \cdot v}{\eta_{total}(T,\omega)} dt \tag{60}$$

Dans les calculs informatiques, si $\eta_{total}$ est égal à "0" ou "nan" (not a number) alors $\frac{F_{utile}\cdot v}{\eta_{total}(T,\omega)} = 0$.

# 9 Calcul et méthode numérique

## 9.1 Intégration

Le pas de temps étant imposé par l'échantillonnage du GPS, on se base sur une méthode d'Euler pour l'intégration.

$$\begin{cases} x_{n+1} = x_n + \dot{x}_n dt \\ \dot{x}_{n+1} = \dot{x}_n + \dfrac{F(x_n, \dot{x}_n, t)}{m} dt \\ x_0 = x(0) \\ \dot{x}_0 = \dot{x}(0) \end{cases} \tag{61}$$

### 9.1.1 Annexe : utilisation de la fonction

Exemple :

```
fct = Puissance.P_rolling
v = donnees["Vitesse"]
theta = donnees["Angles"]
t = donnees["Distance totale"] #la fonction ne depend pas de t
dt = donnees["dt"]
Euler_fct(fct, v, theta, t, dt=dt, CI = 0)
```

v et theta doivent être des listes ou un array, dt peut être une constante, une liste ou un array. Ces listes doivent toutes avoir la même taille.

## 9.2 Erreur

Pour le calcul d'erreur, on utilise la norme $L^2$ définie comme :

$$\|f(t)\|_{L^2} = \int_0^{t_f} f(t)^2 dt \tag{62}$$

Approximée numériquement :

$$\|f(t)\|_{L^2} = \int_0^{t_f} f(t)^2 dt \approx \sum_{i=0}^{N} f_i(t_i)^2 dt_i \tag{63}$$

Finalement, l'erreur est calculée :

$$Erreur_{\%} = \frac{\sqrt{\int (u-v)^2 \, dt}}{\|u\|_{L^2}} * 100 \approx \frac{\sqrt{\sum_{i=0}^{N} (u_i - v_i)^2 \, dt_i}}{\|u\|_{L^2}} * 100 \tag{64}$$

## 9.3 Interpolation

L'interpolation linéaire est une méthode simple et efficace pour obtenir des valeurs à partir d'un tableau de valeurs. On approxime une fonction $z(x)$ localement par une fonction linéaire $f(x)$.

En 1 dimension :

$$\begin{cases} f(x) = ax + b \\ \quad a = \dfrac{z(x_1) - z(x_0)}{x_1 - x_0} \\ \quad b = \dfrac{z(x_1)x_0 - z(x_0)x_1}{x_0 - x_1} \end{cases} \tag{65}$$

En 2 dimension :

$$\begin{cases} f(x,y) = ax + by + c \\ \quad a = \dfrac{z(x_1,y_0) - z(x_0,y_0)}{x_1 - x_0} \\ \quad b = \dfrac{z(x_0,y_1) - z(x_0,y_0)}{y_1 - y_0} \\ \quad c = z(x_0,y_0) - ax_0 - by_0 \end{cases} \tag{66}$$

### 9.3.1 Annexe : utilisation de la fonction

```
# 1 dimension:
z = mcc.interpolation([x_0, x_1], [z_0, z_1], x)

# 2 dimension:
z = mcc.interpolation_2D([x_0,x_1],[y_0,y_1],[z_00,z_01,z_10],x,y)
# z_01 = z(x_0,y_1), z_10 = z(x_1,y_0)
```

# 10 Phare

## 10.1 Modèle

Les phares sont un système On/Off. On modélise simplement l'énergie par :

$$E_{phare} = P_{phare}t \tag{67}$$

$$E_{phare} = P_{phare} * t \tag{68}$$

Avec $t$ le temps d'utilisation.

# 11 Bibliographie

https://en.wikipedia.org/wiki/Rolling_resistance
https://en.wikipedia.org/wiki/Haversine_formula
https://www.pompe-moteur.fr/blog/comment-calculer-la-puissance-d-un-moteur-electrique--n56#
~:text=Calcul%20du%20couple%20d'un%20moteur%20%C3%A9lectrique,-Le%
20couple%20d&text=On%20peut%20le%20calculer%20gr%C3%A2ce,minute%20(tr%
2Fmin).
https://ftp.idu.ac.id/wp-content/uploads/ebook/tdg/TERRAMECHANICS%
20AND%20MOBILITY/epdf.pub_vehicle-dynamics-theory-and-application.
pdf