Projet Optimisation.

Meziane Boucherrab

January 11, 2024

Master 2 Ingénierie Mathématique, MPE, Université Pierre et Marie Curie

Plan de Travail

1. Présentation du projet
2. Algorithme SQP
   - Présentation de l'algorithme SQP
   - Mise en place de l'algorithme SQP
     - Approximation du gradient
     - Calcul du hessien
     - Matrice définie positive
     - Résolution du problème quadratique
     - Recherche linéaire
   - Validation de l'algorithme
     - Test 1 : MHW4D
     - Test 2 : Ariane1
   - Solution théorique du problème d'étagement
     - Définition et reformulation du problème
     - Conditions d'optimalité d'ordre 1
     - Méthode de la sécante
3. Simulation de trajectoire

1 Présentation du projet

La tâche fondamentale d'un lanceur spatial consiste à placer un satellite en orbite, définie par des paramètres tels que l'altitude et la vitesse à atteindre. L'objectif de ce projet est de déterminer le lanceur le plus léger capable de mettre en orbite un satellite d'une masse donnée sur une orbite spécifique. De plus, nous cherchons à déterminer la vitesse de propulsion nécessaire pour atteindre cette orbite.

Dans la première phase de notre travail, nous mettrons en œuvre l'algorithme SQP (Sequential Quadratic Programming), qui est utilisé pour résoudre des problèmes d'optimisation sous contraintes.

Ensuite, nous aborderons le problème de l'étagement, qui consiste à déterminer, à une vitesse propulsive donnée, les masses d'ergols (carburant) permettant d'optimiser la masse de la fusée au moment du décollage.

Les masses d'ergols ainsi obtenues seront utilisées pour simuler les données de vol du lanceur, en prenant en compte les angles de poussée paramétrés. Nous tenterons ensuite de trouver des configurations d'angles permettant de respecter les conditions requises pour la mise en orbite.

2 Algorithme SQP

2.1 Présentation de l'algorithme SQP

L'algorithme SQP (Sequential Quadratic Programming) émerge comme un outil puissant dans l'arsenal des méthodes d'optimisation, particulièrement adapté aux problèmes complexes de lancement spatial. En intégrant SQP dans le processus de conception et d'optimisation des lanceurs spatiaux, avec cette algorithme ilest possible d'affiner les trajectoires, ajuster les paramètres des étages de propulsion et optimiser la consommation d'ergol pour atteindre des performances optimales du lanceur.

2.2 Mise en place de l'algorithme SQP

2.2.1 Approximation du gradient

Pour calculer le gradient $\nabla f(x)$, on utilise la méthode des différences finies. On note h_i le i-ème élément de la base canonique, multiplié par un scalaire suffisamment petit noté h. Pour tout $i \in I$, l'approximation partielle de la dérivée de $f(x)$ par rapport à x_i est donnée par :

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+h_i) - f(x)}{h}$$

2.2.2 Calcul du Hessien

On dispose de deux méthodes itératives, "BFGS" et "SR1", qui permettent une approximation moins coûteuse du hessien lors du calcul.
-BFGS est une méthode d'optimisation itérative qui estime l'inverse du hessien en ajustant une approximation de la matrice hessienne à chaque itération, en se basant sur les gradients de la fonction à optimiser.

$$\begin{cases} H_k = H_{k-1} + \frac{y_{k-1}y_{k-1}^T}{y_{k-1}^T d_{k-1}} - \frac{H_{k-1}d_{k-1}d_{k-1}^T H_{k-1}}{d_{k-1}^T H_{k-1} d_{k-1}}, & \text{si } y_{k-1}^T d_{k-1} > 0 \\ H_k = H_{k-1}, & \text{sinon} \end{cases}$$

-SR1 est une méthode itérative qui actualise une approximation de la matrice hessienne de manière symétrique. Contrairement à BFGS, elle utilise une modification basée sur le produit externe symétrique pour guider l'optimisation en utilisant des informations sur les gradients.

$$\begin{cases} H_k = H_{k-1} + \frac{(y_k - H_{k-1}d_k)(y_k - H_{k-1}d_k)^T}{(y_k - H_{k-1}d_k)^T d_k}, & \text{si } (y_k - H_{k-1}d_k)^T d_k \neq 0 \\ H_k = H_{k-1}, & \text{sinon} \end{cases}$$

2.2.3 Matrice definie Positive

Cette section traite de la procédure de modification de la matrice hessienne lors de la résolution de problèmes d'optimisation.
La fonction ajuste une matrice hessienne H pour garantir qu'elle reste définie positive, assurant ainsi la stabilité de l'algorithme d'optimisation. Si la plus petite valeur propre de H (θ) est déjà positive, la matrice est renvoyée telle quelle. Sinon, la matrice est ajustée en soustrayant un multiple de la matrice identité I. L'ajout de 0.1 à θ assure que toutes les valeurs propres de la matrice ajustée soient strictement positives.

$$H' = \begin{cases} H & \text{si } \theta > 0 \\ H - (\theta + 0.1)I & \text{sinon} \end{cases}$$

2.2.4 Résolution du probléme quadratique

Pour résoudre le problème, nous choisissons de simplifier le problème quadratique

$$\longrightarrow \begin{cases} \min_{d_{\mathrm{QP}}} & \frac{1}{2} {d_{\mathrm{QP}}}^T \nabla^2 \mathcal{L} d_{\mathrm{QP}} + \nabla \mathcal{L}^T d_{\mathrm{QP}} \\ \text{sous contraintes} & \nabla c(x_k)^T d_{\mathrm{QP}} + c(x_k) = 0 \end{cases}$$

par le problème simplifié

$$\longrightarrow \begin{cases} \min_{d_{\mathrm{QP}} \in R^n} & \frac{1}{2} d_{\mathrm{QP}}^T Q d_{\mathrm{QP}} + g^T d_{\mathrm{QP}} \\ \text{sous contraintes} & \nabla c(x_k)^T d_{\mathrm{QP}} + c(x_k) = 0 \end{cases}$$

qui a pour solution

$$\begin{aligned} (d_{\mathrm{QP}}, \lambda_{\mathrm{QP}}) \longrightarrow & \begin{cases} d_x & = d_{\mathrm{QP}} \\ d_\lambda & = \lambda_{\mathrm{QP}} - \lambda_k \end{cases} \\ \longrightarrow & \begin{cases} x_{k+1} & = x_k + d_{\mathrm{QP}} \\ \lambda_{k+1} & = \lambda_{\mathrm{QP}} \end{cases} \end{aligned}$$

en posant:

$$\begin{aligned} Q &= \nabla^2 L(x_k, \lambda_k) \\ A &= \nabla C(x_k)^T \\ g &= \nabla f(x_k) \\ c &= -c(x_k) \end{aligned} \tag{1}$$

et la solution dans ce cas sera (programmée dans MAJ.m):

$$\begin{cases} \lambda_{\mathrm{qp}} & = -\left(AQ^{-1}A^T\right)^{-1}\left(AQ^{-1}g + b\right) \\ d_{\mathrm{qp}} & = -Q^{-1}(A^T\lambda_{\mathrm{qp}} + g) \end{cases} \tag{2}$$

Puis, on effectue le déplacement suivant :

$$\begin{cases} x_{k+1} & = x_k + d_{\mathrm{qp}} \\ \lambda_{k+1} & = \lambda_{\mathrm{qp}} \end{cases} \tag{3}$$

2.2.5 Recherche linéaire

Étant donné que la méthode SQP n'est pas toujours robuste, il est impératif de vérifier la qualité de la solution en introduisant la fonction de mérite. Supposons que la direction de Newton à l'étape k soit connue. Cependant, il est important de noter qu'il n'y a aucune garantie que cette direction minimise la fonction f sous les contraintes c, car le nouveau point résultant ne fait qu'entraîner une diminution du gradient du Lagrangien du problème initial. C'est pourquoi nous introduisons une fonction de mérite pour évaluer la pertinence de cette direction et, par extension, du nouveau point.

Dans cette partie, nous allons procéder de la sorte : à partir du point x_k, nous allons tenter de trouver une direction à l'aide de la recherche linéaire vérifiant la condition d'Armijo. Cette condition consiste à vérifier :

$F(x_k + sd) \leq F(x_k) + c_1 s F'_d(x_k)$
avec :

$F'_d(x_k) = \nabla f(x_k)^T d - \rho \|c(x_k)\|_1$

Où F représente la fonction objectif,$F_d(x_k)$ est la dérivée directionnelle de F, et $c_1$est un paramètre de contrôle. Cette condition garantit une décroissance suffisante de la fonction objectif après le déplacement dans la direction d, proportionnellement à la distance parcourue.

2.3 Validation de l'algorithme

pour la validation de l'algorithme on procéde a l'execution de l'algorithme sqp sur 2 cas teste:

2.3.1 test 1 : MHW4D

le test MHW4D consiste a minimiser

$f(x_1, x_2, x_3, x_4, x_5) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - 5)^4$

sous les contrainte:

$$\begin{cases} c_1(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2^2 + x_3^2 - 3\sqrt{2} - 2 \\ c_2(x_1, x_2, x_3, x_4, x_5) = x_2 - x_3^2 + x_4 - 2\sqrt{2} + 2 \\ c_3(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot x_5 - 2 \end{cases} \tag{4}$$

Avec l'initialisation : $(x_1, x_2, x_3, x_4, x_5) = (-1; 2; 1; -2; -2)$

Analyse du resultat : l'algorithme SQP ne manifeste pas une stabilité optimale, montrant une tendance à s'éloigner initialement dans une direction sous-optimale lors de la première itération. Cette divergence initiale est suivie d'oscillations prononcées avant d'atteindre une convergence approximative après environ 10 itérations. Pendant ce processus, les contraintes semblent être considérablement annulées. En résumé, bien que l'algorithme converge finalement, le parcours n'est pas aussi direct et efficace qu'on pourrait l'espérer. Des améliorations pourraient être envisagées pour renforcer la stabilité et optimiser l'algorithme.

iter:k	fcon	xk	f(xk)	c(xk)	lambda
1	5	-0.861,1.643,1.680,-0.225,-2.277	95.000000	-2.243,-1.828,0.000	-12.190,51.195,-8.876
2	4	-1.320,1.791,1.701,-0.163,-1.068	40.657813	-1.582,-2.234,-0.038	-5.673,12.689,-9.791
3	3	-1.331,1.728,1.824,0.580,-1.115	27.812455	-1.461,-2.095,-0.591	-1.989,-0.731,-9.894
4	1	-0.506,2.305,1.224,-1.261,-2.000	25.432184	-1.259,-1.848,-0.516	-2.621,2.166,-10.783
5	2	-0.852,2.283,1.401,-0.487,-1.119	49.908893	0.063,-1.284,-0.988	-2.260,0.653,-10.772
6	2	-0.522,2.175,1.452,0.013,-1.859	26.794296	0.079,-0.994,-1.046	-2.587,3.004,-11.620
7	2	-0.853,2.285,1.394,-0.077,-1.171	26.548043	0.073,-0.748,-1.030	-1.897,0.665,-11.020
8	0	-1.351,2.431,1.309,0.104,-1.660	20.111396	0.070,-0.564,-1.001	-1.942,0.290,-10.385
9	0	-1.289,2.462,1.216,-0.163,-1.557	33.045566	0.028,-0.007,0.244	-2.211,0.759,-10.116
10	1	-1.201,2.462,1.178,-0.253,-1.661	28.634631	0.010,-0.009,0.006	-2.537,0.375,-8.981
11	0	-1.271,2.461,1.208,-0.175,-1.569	28.496748	0.006,-0.006,-0.006	-2.511,0.143,-8.902
12	0	-1.236,2.462,1.191,-0.215,-1.616	28.478422	0.001,-0.001,-0.006	-2.510,0.133,-8.903
13	0	-1.237,2.462,1.191,-0.216,-1.617	28.495079	0.000,-0.000,-0.002	-2.512,0.124,-8.897
14	0	-1.237,2.462,1.191,-0.216,-1.617	28.508728	0.000,-0.000,0.000	-2.512,0.125,-8.896
15	0	-1.237,2.462,1.191,-0.216,-1.617	28.508725	0.000,-0.000,-0.000	-2.512,0.125,-8.896
16	0	-1.237,2.462,1.191,-0.216,-1.617	28.508725	0.000,-0.000,-0.000	-2.512,0.125,-8.896
17	0	-1.237,2.462,1.191,-0.216,-1.617	28.508725	-0.000,0.000,0.000	-2.512,0.125,-8.896
18	12	-1.237,2.462,1.191,-0.216,-1.617	28.508725	-0.000,0.000,-0.000	-2.512,0.125,-8.896
19	5	-1.237,2.462,1.191,-0.216,-1.617	28.508725	-0.000,0.000,-0.000	-2.512,0.125,-8.896
20	5	-1.237,2.462,1.191,-0.216,-1.617	28.508725	-0.000,0.000,-0.000	-2.512,0.125,-8.896

2.3.2 test 1 : Ariane1

Le test Ariane1 représente une étape cruciale dans le développement des lanceurs spatiaux, visant à optimiser la répartition des masses d'ergol au temps $t0$pour obtenir un lancement optimal. L'objectif est de minimiser la masse totale $M0$ sous la contrainte que la somme des changements de vitesse ΔV doit être égale au changement de vitesse requis $\Delta v_{\text{requise}}$
le probléme a minimiser est:

$$\min_{(m_{e1},m_{e2},m_{e3})} M_0 \quad \text{sous les contraintes } \Delta v = \Delta v_{\text{requise}} \tag{5}$$

Pour résoudre ce problème d'optimisation, il est essentiel de calculer les masses d'ergol optimales pour chaque étage du lanceur. Les coefficients K_i et les vitessesV_i spécifiés pour chaque étage sont des paramètres cruciaux dans ce calcul, reflétant les propriétés de consommation spécifique d'ergol.

i	k_i	v_i (m/s)
1	0.1101	2647.2
2	0.1532	2922.4
3	0.2154	4344.3

$$\Delta v_{\text{requise}} = 11527 m/s$$
$$\mathrm{M}_u = 1700 Kg$$

la solution souhaité est : $(M_{e1}, M_{e2}, M_{e3}) = (145349kg; 31215kg; 7933kg)$

Cette optimisation permet d'assurer un lancement efficace du lanceur Ariane1 en ajustant les masses d'ergol de manière à atteindre les performances requises tout en respectant les contraintes opérationnelles. L'utilisation d'algorithmes d'optimisation avancés, tels que l'algorithme SQP, offre une approche mathématique robuste pour résoudre ce défi complexe d'ingénierie spatiale.

2.4 Solution théorique du problème d'étagement

2.4.1 Définition et reformulation du probléme

On cherche le lanceur le plus léger possible permettant d'atteindre une vitesse propulsive donnée V_p pour un satellite de masse m_u. Par conséquent, le problème principal que nous essayons de résoudre est formulé comme suit :

$$\min_{m_{e,1},m_{e,2},m_{e,3}} \quad J = \frac{m_u}{M_0} \quad \text{sous} \quad V = V_p \tag{6}$$

Nous voulons réécrire ce problème de la manière suivante

$$\min_{x_1,x_2,x_3} \quad f(x_1,x_2,x_3) \quad \text{sous} \quad c(x_1,x_2,x_3) = 0 \tag{7}$$

$$\text{avec} \begin{cases} f(x_1,x_2,x_3) = -(\frac{1+k_1}{x_1} - k_1)(\frac{1+k_2}{x_2} - k_2)(\frac{1+k_3}{x_3} - k_3) \\ c(x_1,x_2,x_3) = v_{e_1} \ln x_1 + v_{e_2} \ln x_2 + v_{e_3} \ln x_3 - V_p \end{cases}$$

Pour réécrire nous définirons de nouvelles variables. Soit

$$x_j = \frac{M_{i,j}}{M_{f,j}} \tag{8}$$

$$J = \frac{m_u}{M_0} = \frac{M_{i,4}}{M_{i,1}} = \frac{M_{i,4}}{M_{i,3}} \times \frac{M_{i,3}}{M_{i,2}} \times \frac{M_{i,2}}{M_{i,1}} \tag{9}$$

Nous pouvons réécrire $\frac{M_{i,j}}{M_{i,j-1}}$ comme:

$$\frac{M_{i,j}}{M_{i,j-1}} = \frac{M_{f,j} - m_{s,j}}{M_{i,j}} \tag{10}$$

$$= \frac{M_{f,j}}{M_{i,j}}(1 + (-\frac{m_{s,j}}{M_{f,j}})) \tag{11}$$

Nous nous rendons compte que

$$k_i = \frac{m_{s,j}}{M_{f,j} - M_{i,j}} = \frac{m_{s,j}}{m_{e,j}}$$

On obtient donc la forme que l'on veut. Pour voir que les conditions aux limites sont les mêmes il suffit juste de se rappeler la formule de V

2.4.2 Conditions d'optimalité d'ordre 1

En écrivant les conditions KKT , on obtient le système d'équation suivant

$$\begin{cases} c(x_1,x_2,x_3) = 0 \\ \nabla f(x_1,x_2,x_3) + \nabla c(x_1,x_2,x_3)^T \lambda = 0 \end{cases}$$

si nous élargissons les termes.

$$\begin{cases} v_{e_1} \ln x_1 + v_{e_2} \ln x_2 + v_{e_3} \ln x_3 - V_p = 0 \\ (\frac{1+k_2}{x_2} - k_2)(\frac{1+k_3}{x_3} - k_3)(\frac{1+k_1}{v_{e_1} x_1}) = \lambda \\ (\frac{1+k_1}{x_1} - k_1)(\frac{1+k_3}{x_3} - k_3)(\frac{1+k_2}{v_{e_2} x_2}) = \lambda \\ (\frac{1+k_1}{x_1} - k_1)(\frac{1+k_2}{x_2} - k_2)(\frac{1+k_3}{v_{e_3} x_3}) = \lambda \end{cases}$$

Soit

$$y_j = \frac{1+k_j}{x_j} - k_j \tag{12}$$

$$\omega_j = \frac{k_j}{1+k_j} \tag{13}$$

$$\psi = y_1 \cdot y_2 \cdot y_3 \tag{14}$$

si on applique un changement de variables

$$\begin{cases} \psi \cdot (\frac{1+k_1}{v_{e_1} x_1}) = \lambda \cdot y_1 \\ \psi \cdot (\frac{1+k_2}{v_{e_2} x_2}) = \lambda \cdot y_2 \\ \psi \cdot (\frac{1+k_3}{v_{e_3} x_3}) = \lambda \cdot y_3 \end{cases}$$

Nous le simplifions donc davantage et on obtient

$$\forall j \in \{1,2,3\}, v_{e_j}(1-\omega_i x_i) = \frac{\lambda}{\psi} = cte$$

A partir de l'expression précedent, on remarque que l'on peut exprimer x_1 et x_2 en fonction de x_3 et réinjecter leurs expressions dans l'équation de la contrainte. On obtient ainsi une équation à une seule inconnue x_3

$$v_{e_1} \ln(\frac{1}{\omega_1}(1 - \frac{v_{e_3}}{v_{e_1}}(1-\omega_3 x_3)) + v_{e_2} \ln(\frac{1}{\omega_2}(1 - \frac{v_{e_3}}{v_{e_2}}(1-\omega_3 x_3)) + v_{e_3} \ln(x_3) - V_p$$

2.4.3 Méthode de la sécante

La prochaine étape consiste à mettre en œuvre une méthode de Newton pour résoudre l'équation ci-dessus: $g(x) = 0$. Bien que nous puissions calculer la dérivée de $g(x)$, il y a une forte probabilité qu'elle devienne nulle à certains points. Par conséquent , nous privilégierons une méthode de Newton approchée, la méthode sécante, qui approxime la dérivée de $g(x)$ à l'aide d'une différence finie du premier ordre.

Données de l'algorithme sécante pour $m_u = 1700$:

M_{e1}	M_{e2}	M_{e3}
145400.851	31238.540	7935.512

Les données obtenues sont approximativement les mêmes que celles de la solution exacte.

3 Simulation de trajectoire

La vitesse réelle V_r peut être calculée en utilisant un simulateur de trajectoire, en tenant compte du problème de trajectoire.

Le fonctionnement des étages propulsifs est décomposé en trois séquences correspondant à la trajectoire du lanceur. Lorsque tous les ergols d'un étage ont pris feu, il est nécessaire de le séparer avant l'allumage de l'étage suivant. Il est supposé que la séparation se fasse instantanément.

Voici le scénario de vol pour le lanceur pour chaque étage j :

1. Le décollage du lanceur est effectué, avec une poussée orientée θ_j
2. Il utilise une quantité m_{e_j} d'ergols.
3. Il évacue l'étage j qui est vide, avec une masse sèche de $k_1 m_{e_1}$
4. Puis, Il évacue le dernier étage. Si la mission est un succès, le satellite sera maintenant en orbite.

Nous avons le plan, il faut maintenant le réécrire en expliquant davantage la dynamique de ces événements. Nous commencerons par décrire d'abord les équations du mouvement.

les donnée specifique qu'on utilisera pour la simulation sont les suivante :

Étage	**Accélération initiale α (m/s^2)**	**Masse d'ergol (kg)**	**Vitesse d'éjection (m/s)**
1	15	145349	2600
2	10	31215	3000
3	10	7933	4400

Table 1: Caractéristiques des étapes

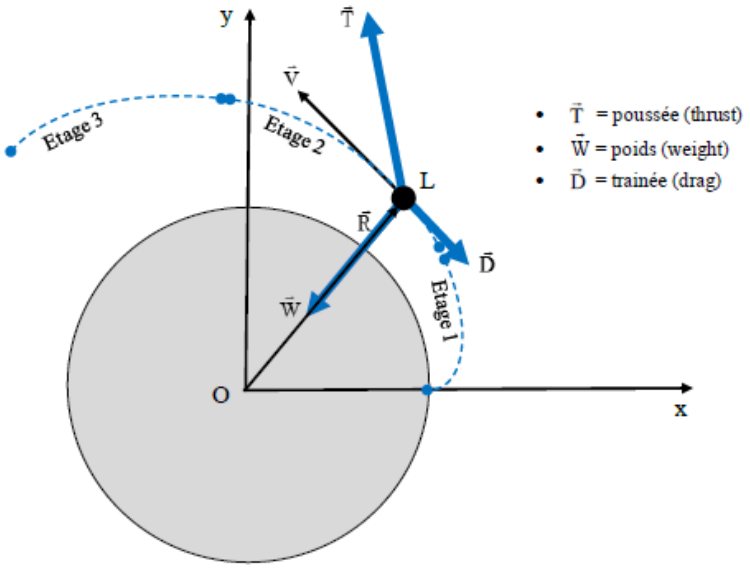
$$\begin{cases} \dot{\overrightarrow{R}} = \overrightarrow{V} \\ \dot{\overrightarrow{V}} = \dfrac{\overrightarrow{T} + \overrightarrow{W} + \overrightarrow{D}}{M} = a(t) \\ \dot{M} = -q \\ (\dot{MV}) = F \end{cases}$$

On se place dans un cadre où $M(t)$ est la masse du lanceur à un instant t, $V(t)$ sa vitesse, $R(t)$ sa position et F les forces extérieures qui s'appliquent au point.

Nos variables dans le problème sont défini comme suit:

$$W(t) = -\mu \frac{\overrightarrow{R}}{R^3} M = -G \frac{m_T M}{(|x(t)| - R_T)^2} \frac{R(t)}{|R(t)|}$$

où:
$G \approx 6.674.1011 m^3 kg^{-1} s^{-2}$ est la constante gravitationnelle.
$m_T = 5,972 \cdot 10^{24} kg$ la masse de la Terre.
$R_T = 6371.103m$ le rayon de la Terre.
R la position du lanceur au temps t.



On défini alors la traînée par la relation :

$$D(t) = -c_x \rho V \overrightarrow{V} = -c_x \rho (|R(t)| - R_T)|V(T)|V(T)$$

où ρ est défini par

$$\rho(t) = \rho_0 e^{\frac{-|R(t)|-R_T}{H}} \quad \text{et} \quad p_0 = \text{densité au sol} = 1.225 kg/m^3$$

Avant de définir la force de poussée, il faut définir ce qu'est l'axe de poussée qui est notée par $\overrightarrow{u}$. Après un calcul on trouve

$$\gamma = arcsin(\frac{R(t)}{|R(t)|}, \frac{V(t)}{|V(t)|})$$

et on obtient la direction u dans les coordonnées cartésiennes:

$$\mathbf{u} = \begin{bmatrix} cos(\gamma + \theta) + sin(\gamma + \theta) \\ sin(\gamma + \theta) - cos(\gamma + \theta) \end{bmatrix} \tag{15}$$

En raison de la discontinuité de $M(t)$, $a(t)$ et $V(t)$, il est techniquement impossible de procéder à l'intégration complète du système en une seule opération. L'intégration numérique est divisée en trois périodes temporelles.

$$[0, t_1] \quad [t_1, t_1 + t_2] \quad [t_1 + t_2, t_1 + t_2 + t_3]$$

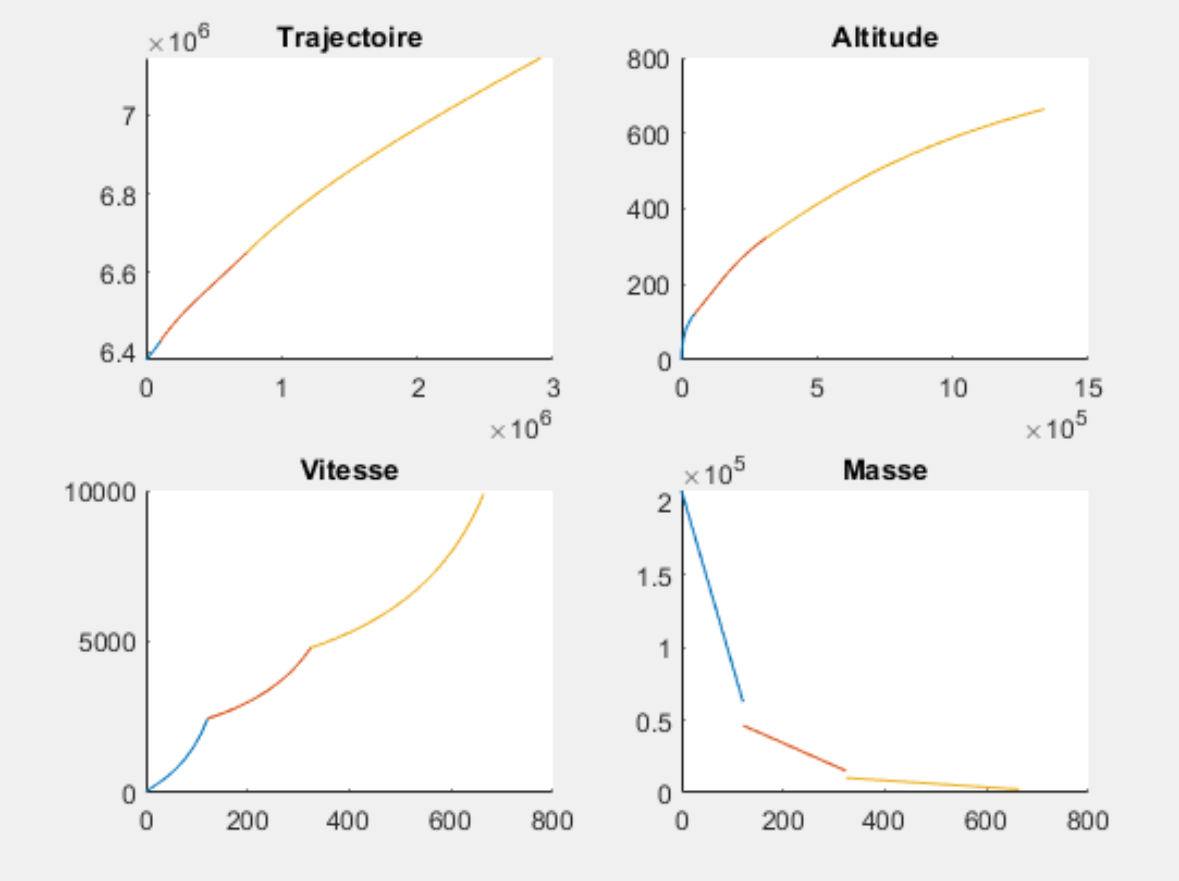
t_j désigne la durée de combustion des ergols de l'étage j qui est donné par:

$$t_j = \frac{m_{e_j}}{q_j}$$

et on obtient le variable changement de masse q_j par

$$\dot{M} = \frac{T_j}{v_{e_j}} = q_j$$

Pour conclure, à la fin de chaque intégration, il faut soustraire à la masse obtenue la quantité $k_j m_{e_j}$ correspondant au largage de l'étage. On utilisera l'intégrateur ODE45 de MATLAB 3 fois pour



résoudre le problème avec pour valeur initiale à l'étage j : la masse mj calculé à l'étape précédente, aj l'accélération initiale de l'étage j et vj la vitesse propulsive de l'étage j.

Nous presonterons ci dessus un figure ilustrant les resultat obtenue pour la simulation de la "Trajectoire" , "Altitude" , "Vitesse" , "Masse" pour les données du lanceur Arian1:

1. Trajectoire :
   La trajectoire semble suivre une courbe caractéristique d'un lancement orbital, montrant une trajectoire ascendante initiale suivie d'une courbure progressive en orbite. Les étapes de propulsion sont clairement observables, indiquées par les changements de direction de la trajectoire.

2. Altitude :
   L'altitude augmente au fur et à mesure que le lanceur progresse dans sa trajectoire. Les changements d'altitude correspondent aux étapes de propulsion et à la consommation d'ergol.

3. Vitesse :
   La vitesse augmente au début du lancement, montrant l'effet combiné de la propulsion et de la diminution de la masse. Les baisses de vitesse correspondent aux changements d'étapes, où la propulsion cesse temporairement.

4. Masse :
   La masse diminue à chaque étape, reflétant la consommation d'ergol et la perte de masse lors de l'éjection des étages. Les paliers dans la courbe de masse indiquent les moments où un étage est largué.