Présentation du projet

1. Problème d'optimisation
2. Algorithme SQP
3. Logiciel

1 Problème d’optimisation

- Formulation du problème
- Lagrangien
- Conditions d’optimalité
- Système non linéaire
- Méthode de Newton

1 Problème d'optimisation

Formulation

$\min\limits_{x \in R^n} f(x)$ sous $c(x) = 0$ $\rightarrow$ problème d'optimisation non linéaire avec contrainte

Notations

- x : n **variables** ou paramètres ou inconnues $\rightarrow$ vecteur de R^n
- f : **critère** ou fonction **coût** ou fonction objectif $\rightarrow$ fonction de R^n dans R
- c : m **contraintes d'égalité** $\rightarrow$ fonction de R^n dans R^m

Résolution pratique

Pas de solution analytique $\rightarrow$ méthodes numériques itératives
$\rightarrow$ recherche d'un **minimum local**

1 Problème d'optimisation

Problème avec contrainte

$$\min_{x \in R^n} f(x) \text{ sous } c(x) = 0 \qquad \rightarrow \quad \text{problème noté (PO)}$$

Fonction de Lagrange (ou lagrangien)

- Le **lagrangien** du problème (PO) est la fonction L de R^{n+m} dans R

$$L(x,\lambda) = f(x) + \lambda^T c(x) \qquad \Leftrightarrow \qquad L(x,\lambda) = f(x) + \sum_{j=1}^{m} \lambda_j c_j(x)$$

- Les λ_j sont les **multiplicateurs de Lagrange** ≈ coefficients de pénalisation des contraintes

→ 1 multiplicateur par contrainte
→ expression des conditions d'optimalité à partir du lagrangien

1 Problème d'optimisation

Conditions d'optimalité

$\min_{x \in R^n} f(x)$ sous $c(x) = 0$ $\rightarrow$ lagrangien $L(x,\lambda) = f(x) + \lambda^T c(x)$

Conditions nécessaire d'ordre 1

- x* minimum local $\Rightarrow$ Il existe un unique $\lambda^* \in R^m$ tel que : $\nabla L(x^*,\lambda^*) = 0$

  $\rightarrow$ (x^*, λ^*) est un point stationnaire du lagrangien

  Conditions de Karush-Kuhn-Tucker **(conditions KKT)**

- Il faut résoudre un système d'équations non linéaires de taille n+m :

$$\nabla L(x,\lambda) = 0 \iff \begin{cases} \nabla_x L(x,\lambda) = 0 \\ \nabla_\lambda L(x,\lambda) = 0 \end{cases} \iff \begin{cases} \nabla f(x) + \nabla c(x)\lambda = 0 \\ c(x) = 0 \end{cases}$$

1 Problème d'optimisation

Système non linéaire

$F(z) = 0 \quad \rightarrow$ système de N équations à N inconnues $\quad \Leftrightarrow \begin{cases} F_1(z_1,\ldots,z_N) = 0 \\ \vdots \\ F_N(z_1,\ldots,z_N) = 0 \end{cases}$

Méthode de Newton

- Valeur initiale $z^0 \quad \rightarrow$ Nouvelle valeur $z^1 = z^0 + \delta z$ telle que $F(z^1) = 0$
- Ordre 1 : $\quad F(z^0 + \delta z) \approx F(z^0) + \nabla F(z^0)^T \delta z = 0$

$$\Rightarrow \quad \delta z = -\nabla F(z^0)^{-T} F(z^0)$$

$$\Rightarrow \boxed{z^1 = z^0 - \nabla F(z^0)^{-T} F(z^0)} \quad \rightarrow \textbf{itération Newton}$$

- Itérations : $\quad z^0, z^1, \ldots, z^k, \ldots \quad$ jusqu'à obtenir $F(z^*) = 0$ à une précision donnée
- **Convergence rapide, mais pas garantie** $\quad \rightarrow$ Technique de globalisation nécessaire

2 Algorithme SQP

- Principe
- Problème quadratique
- Résolution
- Organigramme
- Quasi-Newton
- Modification du hessien
- Globalisation

2 Algorithme SQP

Principe

- **On cherche à résoudre les conditions KKT par la méthode de Newton.**
- Système à résoudre : F(z) = 0 avec $z = \begin{pmatrix} x \\ \lambda \end{pmatrix}$

$$F(z) = \nabla L(x,\lambda) = \begin{pmatrix} \nabla f(x) + \nabla c(x)\lambda \\ c(x) \end{pmatrix}$$

Itération Newton

- Point courant : $z_k = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix}$
- Déplacement : $\delta z = \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix}$ vérifiant $\nabla F(z_k)^T \delta z = -F(z_k)$

$$\Leftrightarrow \nabla^2 L(x_k,\lambda_k) \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = -\nabla L(x_k,\lambda_k)$$

$$\Leftrightarrow \begin{cases} \nabla^2_{xx} L(x_k,\lambda_k) d_x + \nabla c(x_k) d_\lambda & = -\nabla_x L(x_k,\lambda_k) \\ \nabla c(x_k)^T d_x & = -c(x_k) \end{cases}$$

2 Algorithme SQP

Problème quadratique équivalent

On se place au point (x_k, λ_k).

- L'**itération de Newton** pour résoudre les conditions KKT donne le système en variables (d_x,d_λ)

$$\boxed{\begin{cases} \nabla^2_{xx}L(x_k,\lambda_k)d_x + \nabla c(x_k)d_\lambda & = -\nabla_x L(x_k,\lambda_k) \\ \nabla c(x_k)^T d_x & = -c(x_k) \end{cases}}$$

- Le **problème quadratique-linéaire** (QP) en variables $d_{QP} \in R^n$

$$\min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T \nabla^2_{xx}L(x_k,\lambda_k)d_{QP} + \nabla_x L(x_k,\lambda_k)^T d_{QP} \quad \text{sous} \quad \nabla c(x_k)^T d_{QP} + c(x_k) = 0$$

a pour conditions d'ordre 1 le système en variables (d_{QP},λ_{QP})

$$\boxed{\begin{cases} \nabla^2_{xx}L(x_k,\lambda_k)d_{QP} + \nabla_x L(x_k,\lambda_k) + \nabla c(x_k)\lambda_{QP} = 0 \\ \nabla c(x_k)^T d_{QP} + c(x_k) = 0 \end{cases}} \quad \rightarrow \text{ multiplicateurs } \lambda_{QP}$$

- **Les 2 systèmes linéaires sont identiques** en posant : $\begin{cases} d_{QP} = d_x \in R^n \\ \lambda_{QP} = d_\lambda \in R^m \end{cases}$

2 Algorithme SQP

Interprétation

- L'itération de Newton au point (x_k, λ_k) équivaut à la résolution du problème quadratique :

$$\min_{d_{QP} \in R^n} \frac{1}{2} d_{QP}^T \nabla_{xx}^2 L(x_k, \lambda_k) d_{QP} + \nabla_x L(x_k, \lambda_k)^T d_{QP} \qquad \rightarrow \text{ L à l'ordre 2 en x}$$

$$\text{sous} \quad \nabla c(x_k)^T d_{QP} + c(x_k) = 0 \qquad \rightarrow \text{ c à l'ordre 1 en x}$$

$\rightarrow$ Minimiser un modèle quadratique local du lagrangien
sous un modèle linéaire local des contraintes

- Les itérations de Newton pour résoudre les conditions KKT sont équivalentes
à la résolution d'une suite de problèmes quadratiques.

$\rightarrow$ **Programmation Quadratique Séquentielle** (SQP)

2 Algorithme SQP

Résolution du problème quadratique

- En pratique, on remplace le problème quadratique

$$\min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T \nabla^2_{xx} L(x_k,\lambda_k) d_{QP} + \nabla_x L(x_k,\lambda_k)^T d_{QP} \quad \text{sous} \quad \nabla c(x_k)^T d_{QP} + c(x_k) = 0$$

de solution (d_{QP}, λ_{QP}) $\rightarrow \begin{cases} d_x = d_{QP} \\ d_\lambda = \lambda_{QP} \end{cases} \rightarrow \begin{cases} x_{k+1} = x_k + d_{QP} \\ \lambda_{k+1} = \lambda_k + \lambda_{QP} \end{cases}$

- par le problème quadratique

$$\min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T \nabla^2_{xx} L(x_k,\lambda_k) d_{QP} + \nabla f(x_k)^T d_{QP} \quad \text{sous} \quad \nabla c(x_k)^T d_{QP} + c(x_k) = 0$$

de solution (d_{QP}, λ_{QP}) $\rightarrow \begin{cases} d_x = d_{QP} \\ d_\lambda = \lambda_{QP} - \lambda_k \end{cases} \rightarrow \begin{cases} x_{k+1} = x_k + d_{QP} \\ \lambda_{k+1} = \lambda_{QP} \end{cases}$

2 Algorithme SQP

Résolution du problème quadratique

- On peut calculer explicitement la solution du problème quadratique local

$$\min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T \nabla_{xx}^2 L(x_k,\lambda_k) d_{QP} + \nabla f(x_k)^T d_{QP} \quad \text{sous} \quad \nabla c(x_k)^T d_{QP} + c(x_k) = 0$$

$$\Leftrightarrow \min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T Q d_{QP} + g^T d_{QP} \quad \text{sous} \quad A d_{QP} = b \quad \text{avec} \quad \begin{cases} Q = \nabla_{xx}^2 L(x_k,\lambda_k) \\ g = \nabla f(x_k) \end{cases} \quad \text{et} \quad \begin{cases} A = \nabla c(x_k)^T \\ b = -c(x_k) \end{cases}$$

- Si Q est définie positive, la solution est : $\boxed{\begin{cases} \lambda_{QP} = -\left(AQ^{-1}A^T\right)^{-1}\left(AQ^{-1}g + b\right) \\ d_{QP} = -Q^{-1}\left(A^T\lambda_{QP} + g\right) \end{cases}}$

(Pour des problèmes de grande taille, une résolution itérative est plus efficace)

- Le déplacement à réaliser est : $\begin{cases} d_x = d_{QP} \\ d_\lambda = \lambda_{QP} - \lambda_k \end{cases} \rightarrow \begin{cases} x_{k+1} = x_k + d_{QP} \\ \lambda_{k+1} = \lambda_{QP} \end{cases}$

2 Algorithme SQP

Difficultés pratiques

L'algorithme SQP présente les mêmes défauts que la méthode de Newton.

- La matrice $Q = \nabla^2_{xx} L(x_k, \lambda_k)$ est coûteuse à calculer (hessien)
  $\rightarrow$ On remplace Q par une approximation H du hessien
  $\rightarrow$ **Méthode de quasi-Newton**

- La matrice H n'est pas forcément définie positive
  $\rightarrow$ On remplace H par une matrice définie positive « proche » H'
  $\rightarrow$ **Modification du hessien**

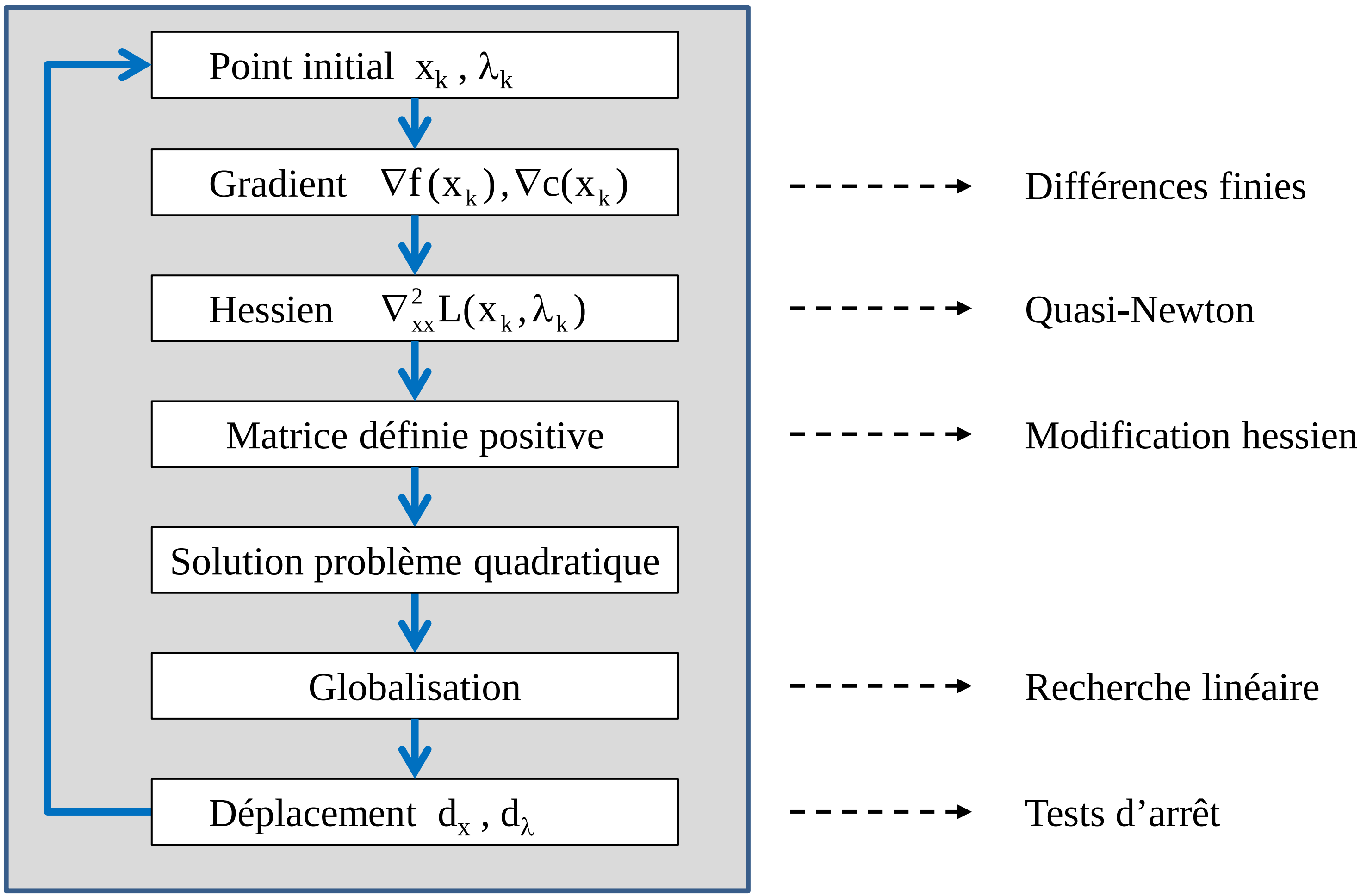
- Le problème quadratique n'est qu'une approximation locale du « vrai » problème
  $\rightarrow$ Le déplacement calculé ne produit pas forcément une amélioration
  $\rightarrow$ **Méthode de globalisation**

$$\begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} \longrightarrow \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix}$$

$$\nabla L(x_k, \lambda_k) \qquad \nabla L(x_{k+1}, \lambda_{k+1}) \xrightarrow{?} \left\| \nabla L(x_{k+1}, \lambda_{k+1}) \right\| < \left\| \nabla L(x_k, \lambda_k) \right\|$$

2 Algorithme SQP

Organigramme



2 Algorithme SQP

Quasi-Newton

- Le problème quadratique à l'itération k s'écrit :

$$\min_{d_{QP}\in R^n} \frac{1}{2} d_{QP}^T Q d_{QP} + g^T d_{QP} \quad \text{sous} \quad A d_{QP} = b \quad \text{avec} \quad \begin{cases} Q = \nabla^2_{xx} L(x_k, \lambda_k) \\ g = \nabla f(x_k) \end{cases} \quad \text{et} \quad \begin{cases} A = \nabla c(x_k)^T \\ b = -c(x_k) \end{cases}$$

- Le hessien du lagrangien $Q = \nabla^2_{xx} L(x_k, \lambda_k)$ est très coûteux à calculer.
  On le remplace par une **approximation H_k** construite à partir du dernier déplacement

$$\begin{cases} d_{k-1} = x_k - x_{k-1} & \to \text{ variation de } x \\ y_{k-1} = \nabla_x L(x_k, \lambda_k) - \nabla_x L(x_{k-1}, \lambda_k) & \to \text{ variation de } \nabla_x L \text{ avec } \lambda = \lambda_k \text{ (nouvelle valeur)} \end{cases}$$

- **Formule BFGS** : $H_k = H_{k-1} + \dfrac{y_{k-1} y_{k-1}^T}{y_{k-1}^T d_{k-1}} - \dfrac{H_{k-1} d_{k-1} d_{k-1}^T H_{k-1}}{d_{k-1}^T H_{k-1} d_{k-1}}$ si $y_{k-1}^T d_{k-1} > 0$
  sinon $H_k = H_{k-1}$

- **Formule SR1** : $H_k = H_{k-1} + \dfrac{(y_{k-1} - H_{k-1} d_{k-1})(y_{k-1} - H_{k-1} d_{k-1})^T}{d_{k-1}^T (y_{k-1} - H_{k-1} d_{k-1})}$ si $d_{k-1}^T (y_{k-1} - H_{k-1} d_{k-1}) \neq 0$
  sinon $H_k = H_{k-1}$

- Initialisation : $H_0 = I$ (réinitialisation toutes les n itérations)

2 Algorithme SQP

Modification du hessien

- Le hessien du lagrangien Q (ou son approximation H) n'est pas forcément défini positif.

  $Q = \nabla^2_{xx} L(x_k, \lambda_k)$ $\rightarrow$ quelconque loin du minimum (x^*, λ^*)
  $\rightarrow$ peut être indéfini au minimum

  Pour un problème avec contrainte, seul le hessien réduit est nécessairement défini positif.

- Pour que le problème quadratique ait systématiquement une solution,
  on modifie le hessien Q (ou son approximation H) pour obtenir une matrice définie positive.
  La modification doit être « minimale » par rapport à la matrice initiale.

- 2 méthodes de modification
  $\rightarrow$ **factorisation de Cholesky modifiée**
  ou $\rightarrow$ **$H' = H + \tau I$** avec $\tau>0$ suffisamment grand ($\rightarrow$ valeurs propres positives)

- Lien avec la méthode de quasi-Newton :
  BFGS $\rightarrow$ H définie positive $\rightarrow$ $H \neq Q$, pas de modification nécessaire
  SR1 $\rightarrow$ H indéfinie $\rightarrow$ $H \approx Q$, modification nécessaire

2 Algorithme SQP

Globalisation

- L'itération SQP équivaut à une itération de la méthode de Newton, qui n'est pas robuste. Il faut contrôler la solution, et si nécessaire la modifier.

$$\begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} \longrightarrow \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} \quad \rightarrow \text{ acceptable ?}$$

- L'objectif est de résoudre les conditions KKT : $\nabla L(x^*,\lambda^*) = 0$

  Il faudrait donc vérifier : $\|\nabla L(x_{k+1},\lambda_{k+1})\| < \|\nabla L(x_k,\lambda_k)\|$

- L'évaluation de ∇L étant très coûteuse, on utilise à la place une **fonction mérite F(x)**.

  La fonction mérite doit mesurer l'amélioration simultanée :

  - du critère f(x) à minimiser $\rightarrow \min\limits_x f(x)$
  - des contraintes c(x) à annuler $\rightarrow \min\limits_x \|c(x)\|_1 = \sum_{i=1}^{m} |c_i(x)|$

2 Algorithme SQP

Fonction mérite

- On peut prendre comme fonction mérite F(x) :

$$\boxed{F(x) = f(x) + \rho \|c(x)\|_1}$$ → critère augmenté avec une **pénalisation ρ**

$$F(x) = f(x) + \lambda^T c(x) + \rho \|c(x)\|_1$$
$$= L(x, \lambda) \qquad + \rho \|c(x)\|_1$$

→ lagrangien augmenté avec une pénalisation ρ pour λ fixé ($\lambda = \lambda_{k+1}$)

- La pénalisation ρ doit être adaptée pour que les contraintes soient respectées avec la précision voulue à la fin des itérations.
  Si les contraintes sont insuffisamment respectées, la pénalisation doit être augmentée.
  On peut également prévoir des pénalisations différentes sur chaque contrainte.

- La solution QP n'est acceptée que si elle fait décroître la fonction mérite.

  $\rightarrow \ F(x_{k+1}) < F(x_k)$

  Si la solution QP n'est pas acceptable, on l'utilise comme **direction de recherche** pour construire un déplacement acceptable.

2 Algorithme SQP

Recherche linéaire

- On cherche à partir de x_k un pas s dans la direction d vérifiant la **condition d'Armijo** :

$\boxed{F(x_k + sd) < F(x_k) + c_1 s F_d'(x_k)}$ $\rightarrow$ condition de décroissance suffisante (c_1 = 0,1)

- La **direction** de recherche linéaire d est donnée par la solution du problème QP.

$$\min_{d \in R^n} \frac{1}{2} d^T Q d + g^T d \quad \text{sous} \quad Ad = b \quad \text{avec} \quad \begin{cases} Q = \nabla^2_{xx} L(x_k, \lambda_k) \\ g = \nabla f(x_k) \end{cases} \quad \text{et} \quad \begin{cases} A = \nabla c(x_k)^T \\ b = -c(x_k) \end{cases}$$

- La **dérivée directionnelle** $F_d'(x_k)$ de la fonction mérite dans la direction d est :

$$F_d'(x_k) = \left(\frac{dF(x_k + sd)}{ds} \right)_{s=0} = \nabla f(x_k)^T d - \rho \left\| c(x_k) \right\|_1$$

- La direction d doit être une **direction de descente** pour la fonction mérite : $F_d'(x_k) < 0$

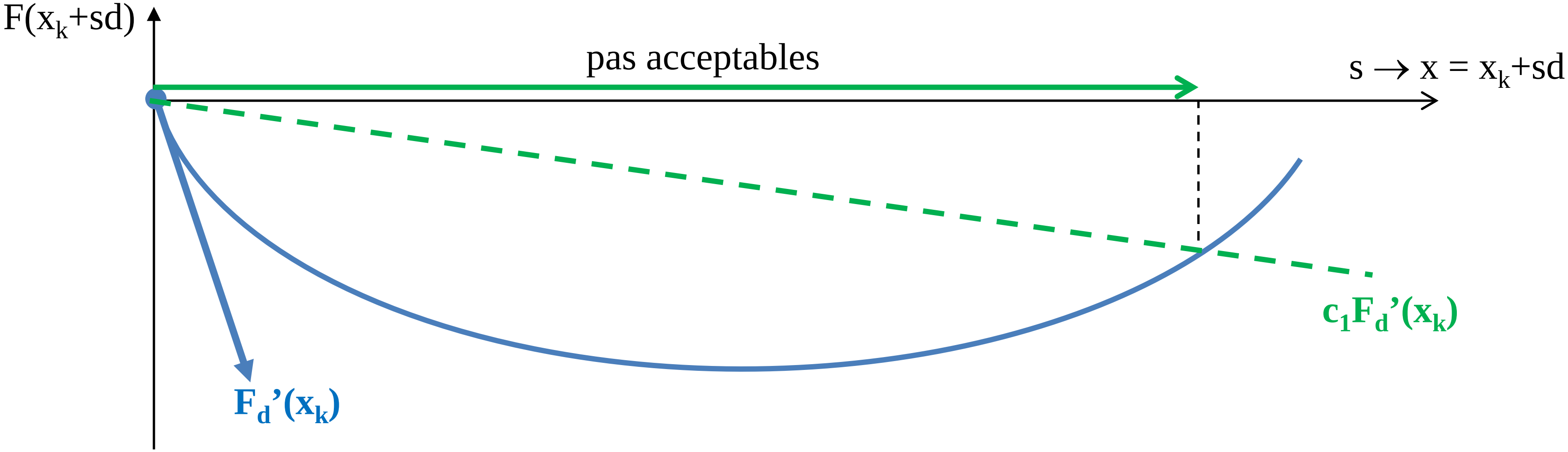
$\rightarrow$ vérifié si Q >0 (modification du hessien)
et ρ assez grand (supérieur aux multiplicateurs λ)
$\rightarrow$ sinon il faut réinitialiser le hessien (H_0 = I) et / ou augmenter ρ

2 Algorithme SQP

Condition d'Armijo

- On cherche à partir de x_k un pas s dans la direction d vérifiant la condition d'Armijo :

$\boxed{F(x_k + sd) < F(x_k) + c_1 s F_d'(x_k)}$ $\rightarrow$ condition de décroissance suffisante (c_1 = 0,1)



- Réglage du pas : $s_0 = 1$ $\rightarrow$ pas de Newton (solution QP)
  $s_{j+1} = s_j/2$ $\rightarrow$ jusqu'à vérifier la condition d'Armijo

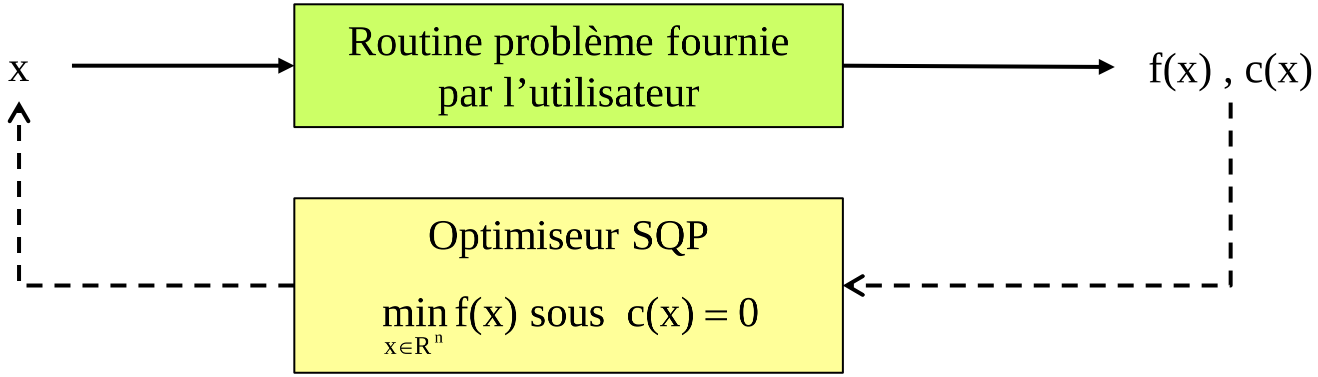
3 Logiciel

- Spécifications
- Validation
- Application lanceur

3 Logiciel

Spécifications du logiciel

- L’utilisateur fournit la routine problème qui :
  - reçoit en entrée les valeurs des variables x
  - renvoie en sortie les valeurs du critère f, des contraintes c



- Le branchement de la routine problème doit être possible sans modifier le code de l’optimiseur.
  $\rightarrow$ permet d’appliquer l’optimiseur à différents problèmes

3 Logiciel

Spécifications du logiciel

- Le nombre de variables et de contraintes est défini par l'utilisateur.

- L'optimiseur prend en compte des bornes sur les variables.

- Il est possible sans modifier la routine problème de :
  - fixer des variables
  - désactiver des contraintes
  - sélectionner le critère

  $\rightarrow$ utile pour traiter différentes étapes d'optimisation sans modifier le code

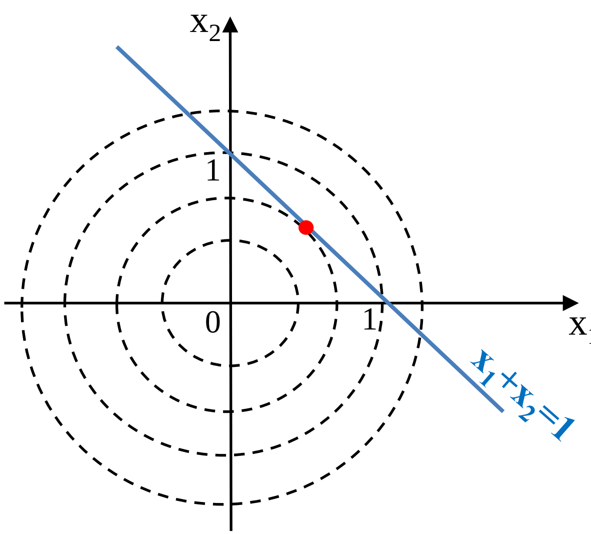
- Les conditions d'arrêt sont définies par l'utilisateur :
  - nombre maximal d'évaluations
  - nombre maximal d'itérations
  - déplacement insuffisant (x)
  - amélioration insuffisante (f)

- Le logiciel affiche des informations sur les étapes et la progression de l'optimisation

  $\rightarrow$ utile pour reprendre un point intermédiaire comme réinitialisation

3 Logiciel

Validation du logiciel



- Cas tests de solution connue
- Fonctions analytiques (critère, contraintes)

Exemple : $\min_{x_1,x_2} x_1^2 + x_2^2$ sous $x_1 + x_2 = 1$

Qualité du logiciel

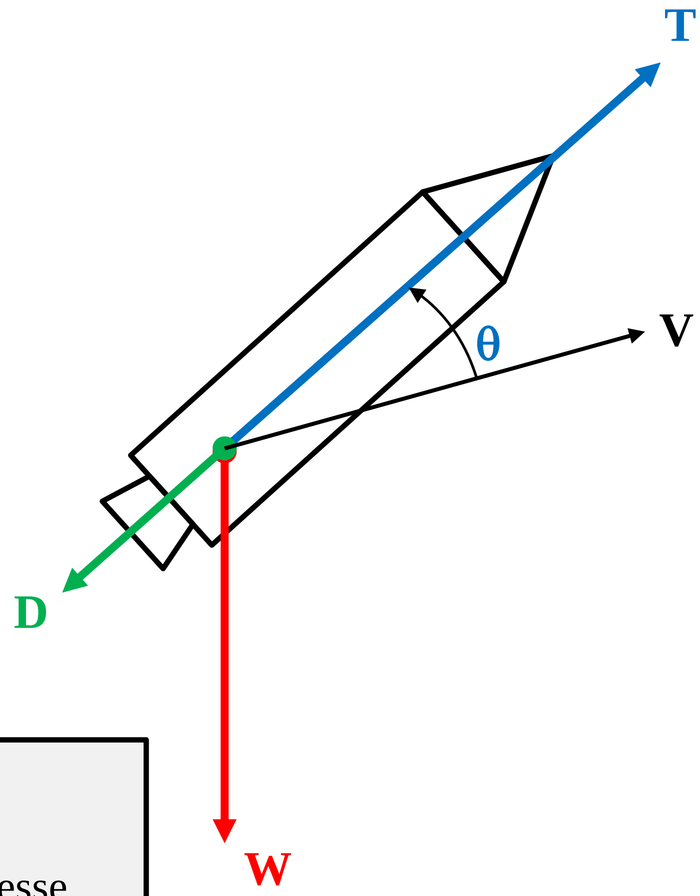
La qualité du logiciel se juge sur :

- la convergence vers la solution correcte avec la précision demandée
- le nombre d'appels de la routine « problème »
- l'adaptation directe à différents problèmes sans modification du code source
- la possibilité pour l'utilisateur de modifier facilement les réglages de l'algorithme

3 Logiciel

Trajectoire de lanceur

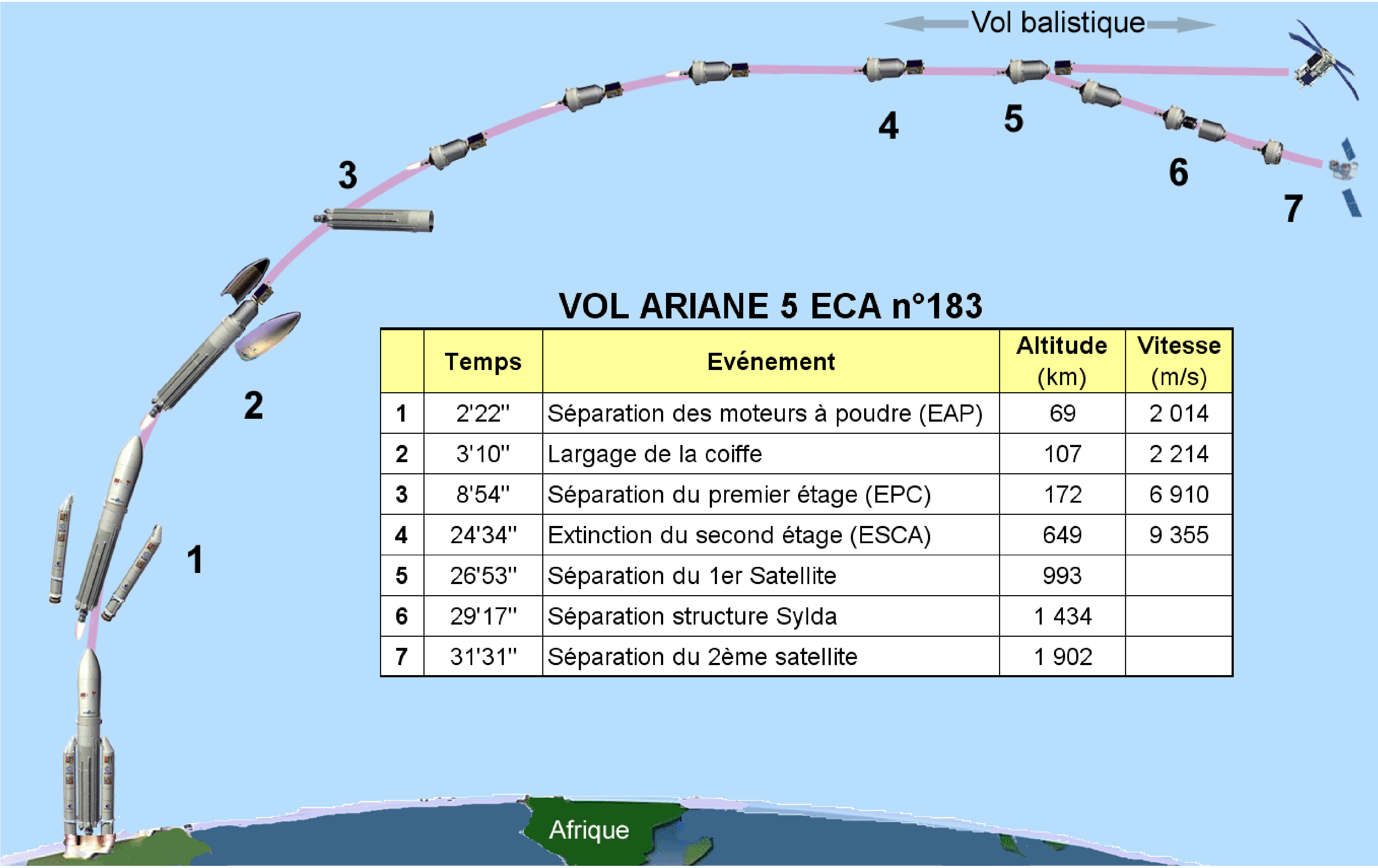
- Décollage du pas de tir
- Plusieurs étages → Allumage du propulseur
  → Combustion des ergols
  → Séparation étage vide
- Injection en orbite → Altitude, Vitesse
- 3 forces → Poussée T
  → Poids W
  → Trainée D
- **Commande** = direction de poussée θ



Optimisation de la trajectoire

→ Atteindre l'altitude de l'orbite en maximisant la vitesse

Projet Optimisation



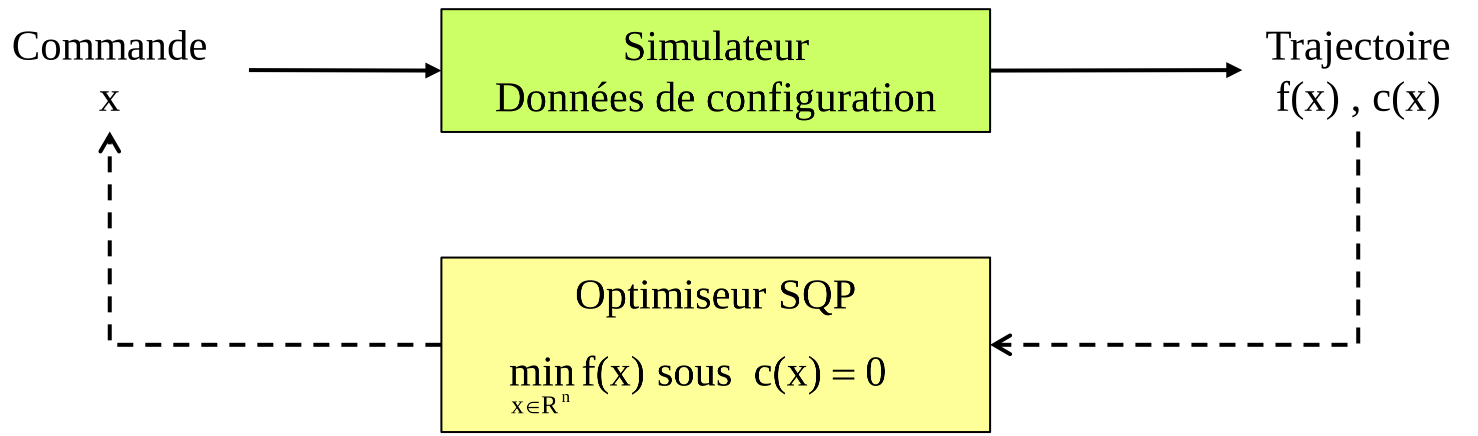
VOL ARIANE 5 ECA n°183

	Temps	Evénement	Altitude (km)	Vitesse (m/s)
1	2'22''	Séparation des moteurs à poudre (EAP)	69	2 014
2	3'10''	Largage de la coiffe	107	2 214
3	8'54''	Séparation du premier étage (EPC)	172	6 910
4	24'34''	Extinction du second étage (ESCA)	649	9 355
5	26'53''	Séparation du 1er Satellite	993	
6	29'17''	Séparation structure Sylda	1 434	
7	31'31''	Séparation du 2ème satellite	1 902	

3 Logiciel

Simulateur de la trajectoire du lanceur

- Entrées :
  - données de configuration $\rightarrow$ fixées
  - loi de commande θ $\rightarrow$ variables x à déterminer
- Sorties :
  - trajectoire du lanceur
  - orbite atteinte (position, vitesse) $\rightarrow$ contraintes $c(x)$ à annuler
  - vitesse finale $\rightarrow$ critère $f(x)$ à minimiser



3 Logiciel

Points importants

- Avancement régulier **chaque séance** pour ne pas prendre de retard
- Valider les routines au fur et à mesure du développement
- Valider le logiciel sur des cas tests connus
- **Programmer proprement** avec un objectif de clarté et de réutilisation sur différents problèmes

Pris en compte dans l'évaluation du travail