MINOQS
Un solveur en optimisation non linéaire

Campagne ADT 2012 — 17 avril 2012

Récapitulatif

- **Acronyme :** MINOQS
- **Porteur de l'ADT :** Jean Charles Gilbert (Jean-Charles.Gilbert@inria.fr)
  CRI : Paris-Rocquencourt **EPI :** Pomdapi (suite d'Estime)
- **Il s'agit d'une nouvelle ADT pour deux ans**
- **Les partenaires internes (EPI/CRI) et externes (autres labos, industriels) de l'ADT :** pas d'autre partenaire.
- **Résumé**

  L'Action de Développement Technologique MINOQS a pour but de développer en C++ une bibliothèque de logiciels permettant de résoudre des problèmes d'optimisation différentiable sous contraintes, en implémentant l'approche newtonienne, celle connue sous le sigle SQP (pour Sequential Quadratique Programming) ou OQS (pour Optimisation Quadratique Successive). Une originalité de l'approche suivie est d'être fondée sur un solveur de problème d'optimisation quadratique convexe utilisant la technique du lagrangien augmenté. Le solveur permettra de traiter les problèmes denses et creux (ou avec des structures plus spécifiques), de même que les problèmes de grande taille (jusqu'à plusieurs millions de variables).

  La seconde année sera consacrée à l'extension des possibilités du solveur, notamment (i) en soignant les cas particuliers tels que la résolution des problèmes de grande taille avec contraintes de borne, (ii) en introduisant des solveurs de systèmes linéaires directs plutôt qu'itératifs, avec mise à jour des facteurs singuliers, (iii) en rendant l'approche plus robuste par régions de confiance, (iv) en permettant l'utilisation des dérivées secondes, (v) en offrant la possibilité de pouvoir traiter une partie des contraintes d'inégalité par points intérieurs, (vi) en résolvant les sous-problèmes quadratiques de manière approchée et (vii) en créant des fonctionnalités permettant de résoudre efficacement des problèmes de commande optimale. Certains de ces sujets de deuxième année ont un aspect plus exploratoire.

Table des matières

Abréviations

ADT : Action de Développement Technologique
CRI : Centre de Recherche de l'Inria
EPI : Équipe-Projet de l'Inria
IJD : Ingénieur Jeune Diplômé
MINOQS : MINimisation par OQS
OQS : Optimisation Quadratique Successive
PQO : Problème Quadratique Osculateur
SQP : Sequential Quadratique Programming, traduit par OQS

1 Contexte

L'Inria a une longue tradition dans l'étude des problèmes d'optimisation continue (différentiable ou non lisse) et dans le développement de solveurs adaptés à ces problèmes ; mentionnons les feues épis PROMATH et NUMOPT, ainsi que les actuelles épis BIPOP, COMMANDS et POMDAPI. Certains de ces solveurs (de problèmes d'optimisation, ce qui sera toujours sous-entendu dans le mot *solveur* ci-dessous) ont été rassemblés dans la bibliothèque `Modulopt`, qui est consultée par les connaisseurs. En tirant sans vergogne la couverture à soi, on peut citer : le solveur spécialisé sur la résolution de problèmes de très grande taille sans contrainte `M1qn3`, qui est largement utilisé en prévision météorologique (4 fois par jour à Météo-France pour établir les bulletins quotidiens) et en océanographie (45 téléchargements en 2011, essentiellement par des unités de recherche en météorologie et en océanographie), et le solveur `SQPlab`, qui est une version classique de SQP/OQS (voir ci-dessous) écrite en Matlab (232 téléchargements en 2011, par un public assez varié : des étudiants, des chercheurs, des ingénieurs, des curieux).

`SQPlab` [25 ; 2007] est un solveur généraliste de problèmes d'optimisation non linéaire non convexe, permettant de minimiser une fonction $f : \mathbb{R}^n \to \mathbb{R}$ sous des contraintes d'égalité et d'inégalité spécifiées respectivement par des fonctions $c_E : \mathbb{R}^n \to \mathbb{R}^{m_E}$ et $c_I : \mathbb{R}^n \to \mathbb{R}^{m_I}$. Nous écrirons un tel *problème d'optimisation non linéaire* comme suit :

$$\begin{cases} \inf_x f(x) \\ c_E(x) = 0 \\ c_I(x) \leqslant 0. \end{cases} \tag{1.1}$$

Ce problème a des variables cachées, les *variables duales* $\lambda \in \mathbb{R}^m$ associées à ses $m = m_E + m_I$ contraintes, qui ne se voient donc pas dans (1.1), mais qui apparaissent lorsqu'on écrit les conditions d'optimalité du problème et que l'on doit générer lorsqu'on cherche à résoudre le problème (1.1). Localement, c'est-à-dire proche d'une solution primale-duale (x_*, λ_*) de (1.1), l'*approche SQP/OQS*, que l'on peut interpréter comme une méthode de Newton, ou plus précisément de Bouligand-Newton, calcule l'itéré primal-dual suivant (x_{k+1}, λ_{k+1}), à partir de l'itéré primal-dual courant (x_k, λ_k), en calculant d'abord une solution primale-duale $(d_k, \lambda_k^{\text{PQ}})$ du *problème quadratique osculateur* (PQO) suivant

$$\begin{cases} \inf_d \nabla f(x_k)^\mathsf{T} d + \frac{1}{2} d^\mathsf{T} \nabla^2_{xx} \ell(x_k, \lambda_k) d \\ c_E(x_k) + c'_E(x_k) d = 0 \\ c_I(x_k) + c'_I(x_k) d \leqslant 0, \end{cases} \tag{1.2}$$

où $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m \mapsto \ell(x, \lambda) = f(x) + \lambda^\mathsf{T} c(x)$ est le *lagrangien* du problème (1.1) ; puis en incrémentant $x_{k+1} := x_k + d_k$ et $\lambda_{k+1} := \lambda_k^{\text{PQ}}$; pour les détails, voir par exemple la partie III de [6 ; 2006] et ses références. On transforme ainsi le problème complexe (1.1) en une suite de problèmes quadratiques de la forme (1.2), d'apparence anodine, certes plus simples à résoudre que (1.1), mais qui sont quand même NP-ardus lorsque le hessien du lagrangien $\nabla^2_{xx} \ell(x_k, \lambda_k)$ n'est pas semi-défini positif. On comprendra alors que l'efficacité de cette approche repose en grande partie sur la qualité du solveur des problèmes quadratiques de la forme (1.2).

Le but principal de l'action de développement technologique MINOQS est de développer un solveur efficace de problèmes d'optimisation non linéaire de la forme (1.1), fondé sur

l'approche SQP/OQS qui vient d'être décrite et sur une résolution des PQO (1.2) par l'algorithme du lagrangien augmenté, une technique que nous avons mise au point depuis quelques années [15, 10 ; 2005-2012]. Comme nous le verrons l'ADT ne se limite pas à cet objectif, mais celui-ci en est le cœur.

Il y a dans le monde beaucoup de solveurs SQP/OQS : celui proposé dans Matlab, `Fmincon`, est notoirement peu efficace, souillé par un vice originel laissé tel quel depuis 20-30 ans, tandis que `Snopt` de Stanford est un exemple de belle réussite, maintenu et amélioré constamment par des professionnels de l'optimisation numérique. De notre côté, hormis `SQPlab` (4500 lignes de Matlab), déjà mentionné, nous avons aussi développé le solveur non linéaire `SQPpro` [26 ; 2008] (4000 lignes de Fortran 2003), fondé sur le solveur de problèmes quadratiques `Qpal` [27 ; 2008] (8000 lignes de Fortran 2003). Ces développements ont été réalisés suite à l'observation de l'efficacité de cette approche sur des problèmes de grande taille [16 ; 2005], lors d'une collaboration avec l'Institut Français du Pétrole [14, 15 ; 2004-05].

Les deux codes cousins `SQPlab` et `SQPpro` ont un gros défaut qui limite pour l'instant leur diffusion, celui de ne pas pouvoir traiter efficacement les problèmes quadratiques osculateurs (1.2) *non* réalisables (c'est-à-dire ayant des contraintes incompatibles). Comme ces problèmes non réalisables ne manquent pas de se présenter, même si le problème original (1.1) est réalisable, `SQPlab` et `SQPpro` échouent lamentablement dans bon nombre de cas. Il existe bien sûr des remèdes permettant de faire face à ces PQO non réalisables, mais ils ont tous des points faibles (introduction de variables supplémentaires ou de non-différentiabilité), si bien que nous préférions explorer d'abord ce qu'un solveur par lagrangien augmenté pouvait apporter avant de les implémenter. L'étude de l'algorithme du lagrangien augmenté sur des problèmes quadratiques non réalisables à été menée récemment [10 ; 2012] et a montré que l'algorithme calculait une « solution » intéressante ; elle est en effet, d'une part, une solution du problème quadratique réalisable *le plus proche* dans un sens bien précis et, d'autre part, une direction de descente d'une fonction de pénalisation exacte associée au problème (1.1). Ce travail est l'élément déclencheur de cette ADT.

Une solution naturelle serait d'apporter des modifications aux solveurs `SQPlab`, `Qpal` et `SQPpro`. Nous avons toujours l'intention de le faire pour le premier (qui est un « jouet Matlab » à destination des étudiants apprenant l'optimisation numérique ou des ingénieurs se faisant la main sur de petits problèmes, que nous maintenons comme accroche vers `Minoqs` et comme outil permettant de faire rapidement des expérimentations), mais l'expérience acquise dans le développement de `Qpal` et `SQPpro` nous a montré les limites du Fortran 2003, langage en lequel ces deux derniers solveurs sont écrits. Il est en effet intéressant de pouvoir proposer des solveurs qui n'imposent pas un choix de structure de données ; par exemple, il est naturel de pouvoir l'utiliser pour des structures matricielles pleines et creuses, mais aussi pour des structures plus spécifiques à certains problèmes [24 ; 2011] ; autre exemple : une matrice hessienne peut-être formée de toutes les dérivées secondes de la fonction considérée ou être approché par une technique quasi-newtonienne pleine ou à mémoire limitée (directe ou inverse), ce qui requiert dans chaque cas des structures de données différentes. En Fortran 2003, cela peut se réaliser par l'introduction de *modules*, sortes d'objets primitifs. C'est ce concept que nous avons utilisé dans `Qpal` et `SQPpro`. Cependant ces objets sont statiques (ils ne peuvent pas être passés en argument d'une procédure), ce qui entraîne diverses limitations, comme celle de ne pas pouvoir appeler aisément le solveur 2 fois dans un même code sur des problèmes de dimension différente !

Il est très vraisemblable que les versions futures de Fortran permettront une utilisation dynamique des objets mais, las d'attendre, nous avons décidé de passer à C++, dont le polymorphisme permet de résoudre élégamment les problèmes que nous venons de mentionner. Ce projet de développement prend alors une envergure plus grande, ce qui nous a conduit à envisager une ADT.

On notera que nous ne sommes pas les seuls à vouloir construire des solveurs d'optimisation plus génériques, plus à même d'être adaptés à des structures de données différentes. Récemment une telle approche orientée-objet à été implémentée en C++ pour l'optimisation quadratique convexe par Gertz et Wright [24 ; 2011] dans leur solveur OOQP ; celui-ci est fondé sur un algorithme de points intérieurs alors que notre approche utilise le lagrangien augmenté.

2 Objectifs

On cherchera dans un premier temps à disposer d'une version C++ de base des logiciels suivants :

- Qpal : solveur de problèmes d'optimisation quadratique convexe utilisant la méthode du lagrangien augmenté, pouvant gérer les problèmes non réalisables,
- Minoqs : solveur de problèmes d'optimisation non linéaire différentiable par l'approche SQP/OQS utilisant Qpal comme solveur de problèmes quadratiques et pouvant tirer parti des « solutions » de problèmes quadratiques *non réalisables* que Qpal peut fournir ; il étend de cette manière les fonctionnalités de SQPpro.

Ces versions devraient être terminées dans les 8-12 premiers mois et devraient déjà être diffusables. La suite de l'ADT portera sur l'amélioration de ces deux solveurs sur les points qui sont détaillés dans la section suivante.

3 Sortie

Avoir à sa disposition un code d'optimisation généraliste ouvre la voie à beaucoup de sujets de recherche (si l'on aime ce genre de sujet) et de collaborations avec des partenaires industriels. Les possibilités de recherche sont nombreuses et sous-jacentes dans la description des points à développer ci-dessous. Quant aux collaborations avec des industriels, il faut noter qu'un bon nombre d'entre elles ont été avortées dans le passé du fait du manque de logiciel adéquat directement disponible. Parmi le public intéressé, on peut aussi citer les développeurs de plateformes de logiciels scientifiques, comme LifeV, NSP [35] ou Scilab [36].

Nous décrivons ci-dessous les points de développement plus fins que nous voudrions entreprendre dans cette ADT, sans doute en trop grand nombre. Certains d'entre eux sont plus exploratoires et conduiront de ce fait à des publications. L'ordre dans lequel ils sont présentés est probablement assez proche de celui dans lequel ils seront considérés.

(*i*) Les problèmes de grande taille (avec des millions de variables) sans contrainte peuvent déjà être résolus par le solveur dédié M1qn3 [31 ; 1989], dont la méthodologie fondée sur des mises-à-jour de ℓ-BFGS *inverses* a également été implémentée dans SQPpro [26 ; 2008]. La possibilité d'utiliser des mises-à-jour de ℓ-BFGS *directes*, en

utilisant l'approche de Byrd, Nocedal et Schnabel [8 ; 1994] est également offerte dans SQPpro et devra être implémentée dans Minoqs. Cette technique permet de résoudre des problèmes de très grande taille avec contraintes (avec une efficacité non nécessairement garantie et qui dépend du problème). Le sujet est aujourd'hui sans surprise, mais il s'agira d'éclaircir la légère perte de performance de SQPpro par rapport au code L-BFGS-B, lorsque le problème n'a que des contraintes de borne. Même si ce sujet ne conduira sans doute pas à une publication, il est important de le traiter proprement, pour assurer la crédibilité du logiciel.

(*ii*) Les systèmes linéaires semi-définis positifs qui ne manquent pas de se présenter dans la résolution de (1.1) par l'algorithme SQP/OQS, par l'intermédiare des PQO (1.2), sont résolus dans Qpal par des méthodes itératives (du gradient conjugué tronqué par les bornes et de l'activation de contraintes). Cette technique est adaptée aux grands problèmes, mais pour les petits problèmes (souvent ceux utilisés par les primo-acquéreurs pour tester les qualités du solveur — voir CUTEr [32] et Libopt [29]), les méthodes de factorisation sont souvent plus efficaces. La difficulté d'utilisation des méthodes de factorisation vient de ce que, lorsqu'elles sont combinées avec les méthodes d'activation de contraintes, il faut pouvoir mettre à jour des facteurs de Cholesky singuliers de matrices *semi*-définies positives, *de manière stable*. On partira des heuristiques proposées dans l'étude préliminaire [9].

(*iii*) SQPpro globalise l'algorithme SQP/OQS local par recherche linéaire : le nouvel itéré primal s'écrit $x_{k+1} = x_k + \alpha_k d_k$, où d_k est solution de (1.2) et $\alpha_k > 0$ est un pas déterminé de manière à faire décroître une fonction de mérite. Cette technique est facile à mettre en œuvre et permet de résoudre beaucoup de problèmes, mais dans certains cas difficiles, l'utilisation de régions de confiance assure plus de robustesse. L'incorporation des régions de confiance est l'étape suivante dans le développement de Minoqs.

(*iv*) Dans SQPpro, le hessien du lagrangien intervenant dans le critère du PQO (1.2) est approché par une matrice de BFGS ou de ℓ-BFGS directe. L'intérêt de cette technique est, du point de vue de l'utilisateur du code, de ne pas devoir calculer les dérivées secondes des fonctions f et c et, du point de vue algorithmique, de ne devoir résoudre que des PQO convexes, qui n'ont donc pas de minima locaux non-globaux et qui peuvent être résolus en temps polynomial. Très bien ! Si cette approche est souvent satisfaisante, elle manque parfois un peu de précision. Par exemple dans les problèmes de commande optimale, une commande bang-bang ne le sera pas vraiment. On peut parfois améliorer la situation en calculant le hessien du lagrangien (des dérivées secondes donc). Il faut alors pouvoir résoudre le PQO résultant, qui n'est plus convexe. Cela peut se faire de manière approchée en utilisant des régions de confiance. Sur ce plan, les possibilités de l'algorithme du lagrangien augmenté ne sont pas connues à ce jour. Cette étape plus exploratoire est envisagée.

(*v*) Beaucoup pensent que les *algorithmes de points intérieurs* [7 ; 2000] sont une approche concurrente de l'algorithme newtonien SQP/OQS. En réalité les deux approches peuvent être combinées, en offrant à l'utilisateur la possibilité de traiter certaines contraintes d'inégalité par points-intérieurs. Il suffit en effet d'introduire

une pénalisation logarithmique des contraintes désignées, d'appliquer l'algorithme SQP/OQS sur le problème résultant et de faire tendre le paramètre de pénalisation vers zéro. Certains problèmes sont mieux résolus de cette manière (problème de commande optimale avec contraintes sur l'état), ce qui nous conduit à suggérer l'implémentation de cette approche dans `Minoqs`.

(*vi*) On peut améliorer le solveur `Qpal` en résolvant les sous-problèmes quadratiques (ceux générés par l'algorithme du lagrangien augmenté) de manière approchées, c'est-à-dire en ne trouvant qu'un minimiseur approché du lagrangien augmenté pour le multiplicateur courant. Ce dernier problème requiert des avancées théoriques non triviales, dont rien ne garantit l'aboutissement, mais qui fait partie de nos préoccupations. Pour des contributions sur ce sujet, voir [4, 13, 33, 12, 5, 19, 37, 38, 21, 39, 20, 34, 18, 1, 22, 23 ; 1982-2010].

(*vii*) Mentionnons pour terminer la possibilité d'utiliser `Minoqs` pour résoudre des problèmes de commande optimale discrétisés (on parle de *transcription directe*). Les problèmes de ce type se caractérisent, du point de vue de l'optimisation numérique, par une structure particulière de données, due au partitionnement des variables en variables d'état et variables de commande. La structure de données souple, dont on cherche à pourvoir `Minoqs`, devrait permettre de prendre en compte ces problèmes plus facilement. La réalisation de ce projet, qui devrait sans doute requérir davantage de temps, permettrait de relancer une collaboration entreprise avec Jean Clairambault (épi BANG) en chronothérapie des cancers [11 ; 2010], permettant de conclure l'ADT par une belle et utile application.

Nous venons de décrire l'état souhaité des solveurs `Qpal` et `Minoqs` à la sortie de l'ADT, lesquels seront ensuite maintenus par le porteur de l'ADT. Nous avons mentionné au passage les sujets de recherche que ces développements ouvraient.

Nous n'oublions pas la diffusion. Pratiquement chaque étape du développement pourra faire l'objet d'une publication, la première pourrait être dans *Mathematical Programming Computation* dès la mise au point du solveur quadratique `Qpal` pouvant prendre en compte la non-compatibilité des contraintes (c'est ce qu'ont fait Gertz et Wright [24 ; 2011] pour leur solveur `OOQP`).

4 Mise en œuvre

4.1 Identification des rôles et organisation

Il y a un seul responsable de l'ADT, J. Ch. Gilbert, qui sera aussi responsable technique et responsable de la valorisation. Il n'y aura donc pas d'effet de dilution de responsabilité. Il faut sans doute rappeller ici qu'il est l'unique auteur des solveurs `Qpal` et `SQPpro`, que l'on cherche à améliorer dans cette ADT.

L'organisation se fera de la manière la plus simple : un encadrement journalier durant les premiers mois et au moment de la rédaction d'article, hebdomadaire autrement, comme le porteur du projet en a l'habitude avec ses doctorants.

4.2 Planification prévisionnelle

Le travail devrait suivre d'assez près et dans l'ordre les étapes énumérées aux sections 2 et 3, que nous reprenons ci-dessous sous forme de jalons.

J_1 ($T_0 + 2$ mois) : prise de contact avec le sujet [6 ; 2006, partie III] ; proposition et expérimentation et finalement choix d'une solution à la gestion de données polymorphes en `C++` pour `Qpal` et `Minoqs` [2, 3, 24 ; 1996-2011].

J_2 ($T_0 + 7$ mois) : écriture en `C++` du solveur `Qpal` (solveur de problème quadratique convexe par lagrangien augmenté), *avec prise en compte des contraintes incompatibles* (en plus du passage au `C++`, c'est l'apport numérique essentiel de cette étape, fondé sur [10 ; 2012]) ; comparaison de l'efficacité du solveur par rapport à d'autres solveurs quadratiques (`Quadprog`, `QPA`, *etc*) au moyen de profils de performance [17 ; 2002], ce qui pourra se faire dans l'environnement `Libopt` [29, 30 ; 2008-2009] ; la version `1.0` de `Qpal` devra être prête et mise à disposition en ligne à la fin de cette étape ; un rapport sur l'implémentation et les résultats numériques sera écrit par la suite (modèle [24 ; 2011]) ; le solveur `Qpal` ne devrait plus être significativement modifié après cette étape, à moins que des avancées soient faites sur le sujet (*vi*) de la page 7.

J_3 ($T_0 + 12$ mois) : écriture en `C++` d'une première version du solveur `Minoqs` (solveur de problème d'optimisation non linéaire), incluant

- approximation BFGS et ℓ-BFGS directe du hessien du lagrangien,
- *la prise en compte des problèmes quadratiques osculateurs non réalisables* (en plus du passage au `C++`, c'est l'apport numérique essentiel de cette étape, qui devrait améliorer substantiellement `SQPpro`),
- recherche linéaire sur la fonction de mérite ℓ_1,
- la possibilité de générer par des directives une version légère, en vue d'une diffusion large sous licence libre ;

comparaison de l'efficacité du solveur par rapport à d'autres solveurs SQP/OQS (`Lancelot`, `Snopt`, *etc*) ou des solveurs de points intérieurs (`Ipopt`, `Knitro`, *etc*), ce qui pourra se faire au moyen de profils de performance [17 ; 2002] dans l'environnement `Libopt` [29, 30 ; 2008-2009] ; la version `1.0` de `Minoqs` devra être prête et mise à disposition en ligne à la fin de cette étape ; un rapport sur une étude théorique et numérique du comportement de l'algorithme sur les problèmes d'optimisation non linéaire *non réalisables* sera écrit par la suite.

Environ 1 an se sera donc écoulé après la réalisation de ces trois premières étapes. Le choix des jalons suivants dépendra en partie des résultats numériques obtenus dans les étapes précédentes, des avancées théoriques réalisées entre-temps, de ce qui nous paraît important avec l'expérience acquise dans la réalisation des logiciels et du temps restant disponible. Nous présentons donc les jalons suivants avec une durée estimée du temps de réalisation, plutôt que comme une suite d'événements à réaliser séquentiellement ; l'ordre dans lequel ces tâches peuvent être réalisées importe peu, une fois que le logiciel de base est fonctionnel, mais pourrait être celui proposé. Le temps total dépasse largement une année, ce qui signifie qu'il faudra faire des choix ou que l'ingénieur retenu fera preuve d'une grande efficacité.

J_4 (3 mois) : résolution soignée des très grands problèmes d'optimisation (plusieurs mil-

lions de variables) avec de simples contraintes de borne (point (*i*) de la page 5).

J_5 (5 mois) : résolution des systèmes linéaires symétriques semi-définis positifs rencontrés dans `Qpal` par factorisation de Cholesky singulière et mise à jour stable des facteurs singuliers après changement de faces actives (point (*ii*) de la page 6).

J_6 (entre 3 et 6 mois) : globalisation par région de confiance plutôt que par recherche linéaire (point (*iii*) de la page 6) ; le temps de réalisation est peu précis, car il faut modifier à la fois `Qpal` et `Minoqs`, d'une manière encore peu claire aujourd'hui.

J_7 (6 mois) : prise en compte des dérivées secondes (point (*iv*) de la page 6).

J_8 (6 mois) : introduction d'une option points-intérieurs pour traiter par relaxation logarithmique certaines contraintes d'inégalité spécifiées par l'utilisateur (point (*v*) de la page 6).

J_9 (6 mois) : résolution approchée des sous-problèmes quadratiques dans `Qpal` (point (*vi*) de la page 7).

J_{10} (entre 6 et 12 mois) : version « commande optimale » de `Minoqs` (point (*vii*) de la page 7).

Bibliographie

[1] R. Andreani, E.G. Birgin, J.M. Martínez, M.L. Schuverdt (2007). On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18, 1286–1302. 7

[2] R. Bartlett (1996). An introduction to `rSQP++`: An object-oriented framework for reduced-space successive quadratic programming. Report, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh. 8

[3] R. Bartlett, L.T. Biegler (2003). `rSQP++`: An object-oriented framework for successive quadratic programming. In *Large-Scale PDE-Constrained Optimization*, Lecture Notes in Computational Science and Engineering 30, pages 316–330. Springer, Berlin. 8

[4] D.P. Bertsekas (1982). *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press. 7

[5] D.P. Bertsekas (1999). *Nonlinear Programming* (seconde édition). Athena Scientific. 7

[6] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, C. Sagastizábal (2006). *Numerical Optimization – Theoretical and Practical Aspects* (seconde édition). Universitext. Springer Verlag, Berlin. [auteurs] [éditeur] [google books]. 3, 8

[7] R.H. Byrd, J.Ch. Gilbert, J. Nocedal (2000). A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89, 149–185. 6

[8] R.H. Byrd, J. Nocedal, R.B. Schnabel (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63, 129–156. 6

[9] A. Chiche (2006). Étude de la stabilité numérique de méthodes de mise à jour de facteurs de Cholesky. Rapport de Projet Personnel en Laboratoire, ENSTA. 6

[10] A. Chiche, J.Ch. Gilbert (2012). How the augmented Lagrangian algorithm deals with an infeasible convex quadratic optimization problem. Rapport de recherche, INRIA, BP 105, 78153 Le Chesnay, France. (à paraître). 4, 8

[11] J. Clairambault, J.Ch. Gilbert (2010). Optimisation chronothérapeutique en cancérologie : chimiothérapies associées en perfusion chronomodulée. Sujet de stage. [pdf]. 7

[12] A.R. Conn, N.I.M. Gould, A. Sartenaer, Ph.L. Toint (1996). Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6, 674–703. 7

[13] A.R. Conn, N.I.M. Gould, Ph.L. Toint (1991). A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28, 545–572. 7

[14] F. Delbos (2004). *Problèmes d'Optimisation de Grande Taille avec Contraintes en Tomographie de Réflexion 3D*. Thèse de doctorat, University Pierre et Marie Curie (Paris VI), Paris. 4

[15] F. Delbos, J.Ch. Gilbert (2005). Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems. *Journal of Convex Analysis*, 12, 45–69. [rapport] [éditeur]. 4

[16] F. Delbos, J.Ch. Gilbert, R. Glowinski, D. Sinoquet (2006). Constrained optimization in seismic reflection tomography: a Gauss-Newton augmented Lagrangian approach. *Geophysical Journal International*, 164, 670–684. [doi]. 4

[17] E.D. Dolan, J.J. Moré (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–213. [doi]. 8

[18] Z. Dostál (2005). Inexact semimonotonic augmented Lagrangians with optimal feasibility convergence for convex bound and equality constrained quadratic programming. *SIAM Journal on Numerical Analysis*, 43, 96–115. 7

[19] Z. Dostál, A. Friedlander, S.A. Santos (1999). Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints. *Computational Optimization and Applications*, 14, 37–53. 7

[20] Z. Dostál, A. Friedlander, S.A. Santos (2003). Augmented Lagrangians with adaptive precision control for quadratic programming with simple bounds and equality constraints. *SIAM Journal on Optimization*, 13, 1120–1140. 7

[21] Z. Dostál, A. Friedlander, S.A. Santos, K. Alesawi (2000). Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints: corrigendum and addendum. *Computational Optimization and Applications*, 23, 127–133. 7

[22] J. Eckstein, P.J.S. Silva (2010). Proximal methods for nonlinear programming: double regularization and inexact subproblems. *Computational Optimization and Applications*, 46, 279–304. 7

[23] J. Eckstein, P.J.S. Silva (2010). A practical relative error criterion for augmented Lagrangians. Rapport de recherche. 7

[24] E.M. Gertz, S. Wright (2011). Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29, 58–81. 4, 5, 7, 8

[25] J.Ch. Gilbert (2007). SQPlab – A Matlab software for solving nonlinear optimization problems and optimal control problems – Version 0.4.2. [rapport]. 3

[26] J.Ch. Gilbert (2008). SQPpro – A solver of nonlinear optimization problems, using an SQP approach – Version 0.3. [hal]. 4, 5

[27] J.Ch. Gilbert (2008). QPAL – A solver of convex quadratic optimization problems, using an augmented Lagrangian approach – Version 0.5. [hal]. 4

[28] J.Ch. Gilbert (2011). *Éléments d'Optimisation Différentiable – Théorie et Algorithmes*. Syllabus de cours à l'ENSTA, Paris. [internet].

[29] J.Ch. Gilbert, X. Jonsson (2008). LIBOPT – An environment for testing solvers on heterogeneous collections of problems. Soumis à *ACM Transactions on Mathematical Software*. 6, 8

[30] J.Ch. Gilbert, X. Jonsson (2009). LIBOPT – An environment for testing solvers on heterogeneous collections of problems – The manual, version 2.1. Rapport Technique 331 (revised), INRIA, BP 105, 78153 Le Chesnay, France. 8

[31] J.Ch. Gilbert, C. Lemaréchal (1989). Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, 45, 407–435. 5

[32] N.I.M. Gould, D. Orban, Ph.L. Toint (2003). CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. *ACM Transactions on Mathematical Software*, 29, 373–394. http://hsl.rl.ac.uk/cuter-www/interfaces.html. 6

[33] W.W. Hager (1993). Analysis and implementation of a dual algorithm for constrained optimization. *Journal of Optimization Theory and Applications*, 79, 427–462. 7

[34] C. Humes, P.J.S. Silva, B.F. Svaiter (2004). Some inexact hybrid proximal augmented Lagrangian algorithms. *Numerical Algorithms*, 35, 175–184. 7

[35] Nsp (2012). A scientific software package. http://cermics.enpc.fr/~jpc/nsp-tiddly/mine.html. 5

[36] Scilab (2012). Le logiciel libre et gratuit de calcul numérique. http://www.scilab.org. 5

[37] M.V. Solodov, B.F. Svaiter (1999). A hybrid approximate extragradient-proximal point algorithm using the enlargement of a maximal monotone operator. *Set-Valued Analysis*, 7, 323–345. 7

[38] M.V. Solodov, B.F. Svaiter (1999). A hybrid projection-proximal point algorithm. *Journal of Convex Analysis*, 6, 59–70. 7

[39] M.V. Solodov, B.F. Svaiter (2000). An inexact hybrid generalized proximal point algorithm and some new results on the theory of Bregman functions. *Mathematics of Operations Research*, 25, 214–230. 7

Index