UNIVERSITÉ DE MONTRÉAL

LA MÉTHODE DES RÉSIDUS CONJUGUÉS POUR CALCULER LES DIRECTIONS EN OPTIMISATION CONTINUE

MARIE-ANGE DAHITO
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AOÛT 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

LA MÉTHODE DES RÉSIDUS CONJUGUÉS POUR CALCULER LES DIRECTIONS EN OPTIMISATION CONTINUE

présenté par : DAHITO Marie-Ange
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ANJOS Miguel F., Ph. D., président
M. ORBAN Dominique, Doctorat, membre et directeur de recherche
M. PRUDHOMME Serge, Ph. D., membre et codirecteur de recherche
M. AUDET Charles, Ph. D., membre

DÉDICACE

À ma famille et mes amis...

REMERCIEMENTS

Ce document, signant la fin d'une maîtrise, voit le jour grâce à bien des personnes que je me dois de remercier.

Tout d'abord, je tiens à remercier les membres du jury constitué de M. Miguel Anjos, M. Charles Audet, M. Serge Prudhomme et M. Dominique Orban, pour leurs appréciations à l'égard de mon travail.

J'adresse également mes remerciements à mes directeurs M. Dominique Orban et M. Serge Prudhomme pour leur assistance, leurs conseils, mais également leur confiance. Plus particulièrement je souhaite remercier M. Dominique Orban pour son soutien, pour le précieux savoir qu'il m'a transmis en optimisation continue, ainsi que pour le temps qu'il m'a consacré.

Je n'oublie pas mes collègues du GERAD qui se sont toujours montrés disponibles et encourageants. En plus d'avoir été une source de connaissances, ils ont aussi su apporter une atmosphère à la fois studieuse et agréable au bureau, ce qui a facilité le déroulement de cette maîtrise.

Je souhaite enfin exprimer ma gratitude à ma famille et à mes amis proches qui, même de l'autre côté de l'Atlantique, m'ont exprimé un soutien indéniable.

RÉSUMÉ

La méthode du gradient conjugué (CG) est une méthode proposée par Hestenes et Stiefel afin de résoudre des systèmes linéaires symétriques et définis positifs. En optimisation non linéaire sans contraintes, on y recourt de manière quasi-systématique pour le calcul des directions. Lorsque la matrice du système n'est pas définie positive, la variante en recherche linéaire proposée par Dembo et Steihaug, et celle de Steihaug en régions de confiance, permettent tout de même d'utiliser CG.

La méthode des résidus conjugués (CR) est également proposée par Hestenes et Stiefel pour les cas où la matrice est définie positive. Elle partage avec CG la décroissance monotone du modèle quadratique, ce qui en fait un bon candidat pour les méthodes de région de confiance. De plus, les résidus dans CR décroissent de manière monotone, ce qui est intéressant, en particulier pour les méthodes de type Newton inexact, souvent utilisées en recherche linéaire.

Dans cet ouvrage, nous proposons des variantes de CR pour les cas où la matrice n'est pas définie positive et étudions la performance de ces modifications dans des contextes de recherche linéaire et de région de confiance. Pour ce faire, nous comparons la performance de nos algorithmes aux variantes de CG correspondantes. Nous nous intéressons également à CRLS qui est l'équivalent de CR pour les moindres carrés linéaires, et suggérons une modification de cette méthode pour traiter le cas de la courbure nulle.

Nos résultats montrent que CR est essentiellement équivalente à CG, et parfois meilleur, notamment pour résoudre les problèmes non convexes. CR présente aussi un léger avantage dans la résolution de problèmes convexes en recherche linéaire. Cette méthode permet souvent d'effectuer moins de produits hessien-vecteur que CG. La résolution des problèmes aux moindres carrés non linéaires montre une équivalence en termes de performance entre LSMR et LSQR qui sont les variantes, construites à partir du processus de Lanczos, de CRLS et CGLS pour résoudre l'équation normale. LSMR montre néanmoins un léger avantage en termes d'évaluation du résidu.

ABSTRACT

The conjugate gradient method (CG) is a proven method for computing directions in unconstrained nonlinear optimization. This method, described by Hestenes and Stiefel, solves symmetric positive-definite linear systems. If the operator is not positive definite, the extension proposed by Dembo and Steihaug for linesearch and that of Steihaug for trust regions, make CG suitable again.

The conjugate residual method (CR) was also proposed by Hestenes and Stiefel for positive-definite operators. Like CG, CR makes the quadratic model decrease monotonically, which makes it relevant in a trust-region context. But CR also makes the residual decrease monotonically, a particularly interesting property for inexact Newton methods, often used in a linesearch context.

In linesearch and trust-region contexts, we propose modifications of CR when the operator is not positive definite, and compare their performance with those of the corresponding extensions of CG. Our tests show that CR is, for the most part, equivalent to CG. It performs better than CG on nonconvex problems, and slightly better on convex problems in a linesearch context. The advantage is often in terms of operator-vector products.

Finally, we consider the CRLS variant of CR for linear least-squares and investigate the case of zero curvature. We perform experiments with LSMR and LSQR, which are the versions of CRLS and CGLS built from the Lanczos process, to solve the normal equation. This reveals that LSMR performs as well as LSQR and enables slight savings in terms of residual evaluations.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CR	Conjugate Residual method
CRLS	Conjugate Residual method for Least Squares
CG	Conjugate Gradient method
CGLS	Conjugate Gradient method for Least Squares
CN1	Condition nécessaire d'optimalité d'ordre 1
$\|.\|$	Norme euclidienne
$\succ$	Défini positif
$\succeq$	Semi-défini positif
f	Fonction objectif
Lettre latine majuscule (A)	Matrice
Lettre latine minuscule (a)	Vecteur
Lettre grecque minuscule (α)	Scalaire
H	Approximation du hessien de f au point courant x
g	Gradient de f au point courant x
J	Jacobien de f au point courant x
F	Résidu aux moindres carrés
s.c.	Sous contrainte

CHAPITRE 1 INTRODUCTION

En optimisation continue sans contraintes, on souhaite minimiser une fonction f non linéaire, à valeurs réelles et possédant des dérivées secondes continues. Il existe deux principales familles de méthodes qui sont les méthodes de recherche linéaire et les méthodes de région de confiance. Afin d'obtenir une solution au problème d'optimisation, les méthodes traditionnelles doivent faire face à des sous-problèmes qui impliquent la résolution, exacte ou approchée, de systèmes linéaires.

La méthode du gradient conjugué, ou Conjugate Gradient method (CG) en anglais, est une méthode éprouvée et largement utilisée pour résoudre ces sous-problèmes. Une méthode qui lui est similaire en de nombreux points est la méthode des résidus conjugués, ou Conjugate Residual method (CR) en anglais. Néanmoins, le comportement de CR dans ce cadre n'est pas décrit dans la littérature. Il s'agit alors ici de l'étudier et de comparer sa performance à celle de CG.

1.1 Définitions et concepts de base

Dans cette section, nous définissons de manière plus détaillée le cadre de la recherche, ainsi que les concepts de base nécessaires à la compréhension du problème. Nous considérons des problèmes d'optimisation *non linéaires continus* et ne possédant *aucune contrainte* et nous souhaitons résoudre les sous-problèmes associés à l'aide de *méthodes itératives*. Tout d'abord, posons le cadre de l'optimisation continue sans contraintes.

1.1.1 L'optimisation continue sans contraintes

L'optimisation continue sans contraintes est une branche des mathématiques numériques qui vise à minimiser une fonction à valeurs réelles, c'est-à-dire à trouver un point où elle atteint sa plus faible valeur. Nous considérons une fonction, appelée *fonction objectif* et notée f, définie sur $\mathbb{R}^n$, et deux fois continûment différentiable. Nous nous focalisons sur des fonctions non linéaires, les problèmes linéaires n'étant pas rencontrés en optimisation sans contraintes.

En pratique, selon les propriétés de f, il est rare de pouvoir trouver un minimum global, ni même de pouvoir assurer que la solution candidate obtenue est un minimum local. Ainsi, les méthodes d'optimisation traditionnelles recherchent plutôt un *point stationnaire*, c'est-à-dire un point où le gradient s'annule. Néanmoins, dans certains cas, il est possible de savoir si un minimum local ou global existe et s'il est unique. Les sous-sections suivantes permettent

de formaliser ce problème de recherche de point stationnaire et définissent des notions utiles. Nous désignons une matrice par une lettre latine majuscule, un vecteur par une lettre latine minuscule et un scalaire par une lettre grecque minuscule. Tous les vecteurs qui interviennent dans ce mémoire sont en colonne.

1.1.2 Concepts utiles

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable. Le *gradient* de f en $x = [x_1\ x_2\ \dots\ x_n]^T \in \mathbb{R}^n$, noté $\nabla f(x)$, est le vecteur des dérivées partielles de f par rapport à chacune de ses variables :

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \dots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}.$$

Si l'objectif est de la forme $f(x) = \frac{1}{2}\|F(x)\|^2$, avec $F(x) = [f_1(x_1, \dots, x_n)\ \dots\ f_m(x_1, \dots, x_n)]^T$ un système de $m \in \mathbb{N}\backslash\{0\}$ fonctions non linéaires, le *jacobien* de F est la matrice de taille $m \times n$:

$$J(x) = \begin{bmatrix} \nabla f_1(x)^T \\ \nabla f_2(x)^T \\ \dots \\ \nabla f_m(x)^T \end{bmatrix}.$$

Si f est de classe C^2, c'est-à-dire que ses dérivées secondes existent et sont continues, on définit le *hessien* de f comme la matrice de ses dérivées secondes par rapport à chaque variable. L'appartenance de f à C^2 offre la propriété que cette matrice est symétrique. Elle s'écrit alors

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ & & \ddots & \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}.$$

Au voisinage de $x \in \mathbb{R}^n$, les valeurs de f peuvent être approchées grâce à la formule de Taylor-Young de degré 2. Soit $s \in \mathbb{R}^n$, alors

$$f(x+s) = f(x) + \nabla f(x)^T s + \tfrac{1}{2} s^T \nabla^2 f(x) s + o(\|s\|^2). \tag{1.1}$$

L'égalité (1.1) est utile pour la construction locale de modèles quadratiques de f. En effet,

plutôt que de directement minimiser l'objectif, les méthodes itératives qu'on étudie se servent de modèles de celui-ci qui, eux, sont minimisés. Cela permet, à chaque itération, de réduire la valeur de f en réduisant celle du modèle. Ceci est expliqué plus en détail dans la section 1.1.4.

1.1.3 Définitions fondamentales

Le point x est un *minimum local* de f si et seulement s'il existe un voisinage ouvert $\mathbb{V}$ contenant x tel que pour tout point $y \in \mathbb{V}$, $f(y) \geq f(x)$. Pour f différentiable, ce point vérifie $\nabla f(x) = 0$. Si pour $y \neq x$ l'inégalité est stricte, on parle de *minimum local strict*. Si elle est vraie quelque soit le voisinage, alors x est un *minimum global*.

On dit que f admet un *maximum local* en x si et seulement si x est un minimum local de $-f$. Il s'agit d'un *maximum local strict* de f si et seulement s'il s'agit d'un minimum local strict de $-f$. Le même raisonnement s'applique au *maximum global*.

Pour f différentiable, on dit que x est un *point de selle* s'il vérifie $\nabla f(x) = 0$ mais ne correspond ni à un minimum local ni à un maximum local.

Le problème d'optimisation consiste à trouver un *point stationnaire* de f. Un point x est stationnaire si et seulement s'il vérifie $\nabla f(x) = 0$. Ce point peut correspondre à un minimum local, à un maximum local ou encore à un point de selle.

Le concept de *direction de descente* est aussi évoqué dans ce mémoire : pour $\nabla f(x) \neq 0$, $p \in \mathbb{R}^n$ est une direction de descente pour f en x si et seulement si $p^T \nabla f(x) < 0$. Il existe alors une longueur de pas $\alpha > 0$ telle que $f(x + \alpha p) < f(x)$.

La propriété de convexité est également importante en optimisation. Une fonction réelle $f \in C^2$ est dite *strictement convexe* si et seulement si pour tout $x \in \mathbb{R}^n$, $\nabla^2 f(x)$ est défini positif, c'est-à-dire que $v^T \nabla^2 f(x) v > 0$ pour tout vecteur v non nul de $\mathbb{R}^n$. Si une fonction strictement convexe possède un minimum local, alors on peut également affirmer que celui-ci est en fait global et unique. C'est l'un des rares cas où il est possible de certifier qu'on a atteint une solution au problème de minimisation de f.

Soit le point courant x, par la suite nous notons $H = H^T \approx \nabla^2 f(x)$, $g = \nabla f(x)$, $J = J(x)$ et $F = F(x)$. Si on considère un itéré x_k, ces notations prennent un indice k. Nous notons également $H \succ 0$ pour indiquer que H est définie positive, et $H \succeq 0$ pour dire que H est semi-définie positive. La norme utilisée dans tout le document est la norme euclidienne.

1.1.4 Modèle quadratique et conditions d'optimalité

Soient $f : \mathbb{R}^n \to \mathbb{R}$ de classe C^2 et $x \in \mathbb{R}^n$. On souhaite faire décroître f autour de x. Par (1.1), on peut construire un modèle de f autour de x :

$$f(x+s) \approx q(s) = f(x) + g^T s + \tfrac{1}{2} s^T H s. \tag{1.2}$$

Notons que $\nabla q(s) = g + Hs$ et $\nabla^2 q(s) = H$.

On souhaite désormais réduire la valeur de f à l'aide de q, c'est-à-dire trouver un pas s tel que $f(x+s) < f(x)$. Pour cela il est nécessaire de rappeler les conditions d'optimalité d'ordre 1 et 2 pour $f \in C^2$.

Condition nécessaire d'optimalité d'ordre 1 (CN1) : si x_* est un minimum local de $f \in C^1$ alors $\nabla f(x_*) = 0$.

Conditions nécessaires d'optimalité d'ordre 2 : si x_* est un minimum local de $f \in C^2$ alors $\nabla f(x_*) = 0$ et $\nabla^2 f(x_*) \succeq 0$.

Conditions suffisantes d'optimalité d'ordre 2 : soit $x_* \in \mathbb{R}^n$ tel que $\nabla f(x_*) = 0$ et $\nabla^2 f(x_*) \succ 0$, alors x_* est un minimum local strict de $f \in C^2$.

La recherche d'un point stationnaire de q se résume alors à chercher s_* tel que la CN1 soit respectée, autrement dit tel que

$$Hs_* = -g. \tag{1.3}$$

Cette équation se nomme *équation de Newton* si H désigne le hessien, ou *équation de quasi-Newton* lorsqu'il s'agit d'une approximation de celui-ci.

1.1.5 Taux de convergence rencontrés

Appelons x_0 l'estimation initiale de x_* un minimum local strict de f, et $\{x_k\}$ la suite des estimations produites par une méthode de recherche linéaire ou de région de confiance.

Il y a convergence *globale* si quelque soit x_0, la méthode identifie un point stationnaire dans le sens où $\lim_{k\to\infty} \nabla f(x_k) = 0$.

On parle de convergence *locale* s'il existe un voisinage $\mathbb{V}$ de x_* tel que, pour tout $x_0 \in \mathbb{V}$, $\lim_{k\to\infty} x_k = x_*$

Si $\lim_{k\to\infty} x_k = x_*$, le taux de convergence est dit *superlinéaire* si $\lim_{k\to+\infty} \frac{\|x_{k+1}-x_*\|}{\|x_k-x_*\|} = 0$. On parle de convergence *quadratique* s'il existe une constante $\gamma > 0$ telle que $\limsup_{k\to+\infty} \frac{\|x_{k+1}-x_*\|}{\|x_k-x_*\|^2} = \gamma$.

1.2 Éléments de la problématique

CR et CG sont des méthodes itératives qui ont été développées afin de résoudre des systèmes linéaires tels que (1.3) lorsque H est définie positive. Si f est strictement convexe alors, en tout point $x \in \mathbb{R}^n$, $\nabla^2 f(x) \succ 0$. Ainsi, CR et CG peuvent résoudre (1.3) en prenant $H = \nabla^2 f(x)$ et $g = \nabla f(x)$. Les algorithmes 1 et 2 décrivent les deux méthodes et montrent une grande similitude dans l'écriture de celles-ci.

Algorithme 1 : CG

Initialiser $s_0 = 0$, $r_0 = -g$, $p_0 = r_0$
pour $k = 1, 2, 3, \ldots$ **faire**
  $\alpha_k = \frac{\|r_{k-1}\|^2}{p_{k-1}^T H p_{k-1}}$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k H p_{k-1}$
  $\beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$
  $p_k = r_k + \beta_k p_{k-1}$
Retourner s_k

Algorithme 2 : CR

Initialiser $s_0 = 0$, $r_0 = -g$, $p_0 = r_0$
pour $k = 1, 2, 3, \ldots$ **faire**
  $\alpha_k = \frac{r_{k-1}^T H r_{k-1}}{\|H p_{k-1}\|^2}$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k H p_{k-1}$
  $\beta_k = \frac{r_k H r_k}{r_{k-1} H r_{k-1}}$
  $p_k = r_k + \beta_k p_{k-1}$
Retourner s_k

Les méthodes CG et CR sont appelées des méthodes de Krylov. Elles génèrent des estimations s_k appartenant au $k^{\text{ième}}$ espace de Krylov : $K_k = vect\{g, Hg, \ldots, H^{k-1}g\}$. À chaque itération, le résidu est $r_k = -g - Hs_k$. On peut alors définir une fonction résidu sur $\mathbb{R}^n$

$$r : s \mapsto -g - Hs. \tag{1.4}$$

CG choisit s_k pour minimiser la fonction quadratique q (1.2) dans K_k, tandis que dans CR, s_k minimise $\frac{1}{2}\|r\|^2$ dans K_k. Plus de détails sur ces méthodes sont donnés dans la section 2.2.1.

Nous listons ci-dessous quelques propriétés importantes de CG et CR.

Théorème 1.2.1. *(Steihaug, 1983). Si $H \succ 0$, les propriétés suivantes sont vérifiées par CG pour $k = 0, 1, \ldots$:*

$$q(s_{k+1}) < q(s_k); \tag{1.5}$$

$$\alpha \mapsto q(s_k + \alpha(s_{k+1} - s_k)) \text{ décroît de manière monotone sur } [0, 1]. \tag{1.6}$$

Théorème 1.2.2. *(Fong et Saunders, 2012). Si $H \succ 0$, les propriétés suivantes sont vérifiées*

par CR pour $k = 0, 1, \ldots$ *:*

$$q(s_{k+1}) < q(s_k); \tag{1.7}$$

$$\alpha \mapsto q(s_k + \alpha(s_{k+1} - s_k)) \text{ } \textit{décroît de manière monotone sur } [0, 1]; \tag{1.8}$$

$$\|r(s_{k+1})\| < \|r(s_k)\|; \tag{1.9}$$

$$\alpha \mapsto \|r(s_k + \alpha(s_{k+1} - s_k))\| \text{ } \textit{décroît de manière monotone sur } [0, 1]. \tag{1.10}$$

La décroissance monotone de la quadratique pour ces méthodes est illustrée par la figure 1.2. Quant au résidu, on observe bien sur la figure 1.1 que la décroissance de $\|r_k\|$ est monotone pour CR, mais pas pour CG. Ces graphes ont été produits en utilisant la matrice sts4098 et le membre de droite qui lui est associé dans la banque de données *SuiteSparse Matrix Collection*[1].

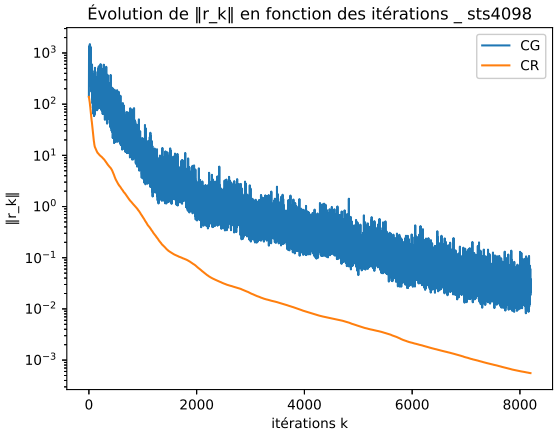

Figure 1.1 Décroissance du résidu dans CG et CR sur la matrice sts4098

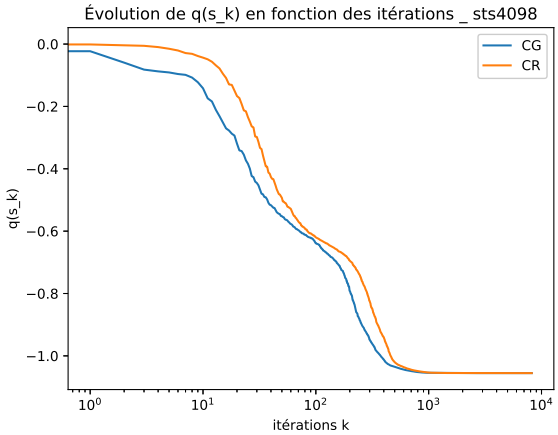

Figure 1.2 Décroissance de la quadratique dans CG et CR sur la matrice sts4098

Pour un objectif de classe C^2, les méthodes d'optimisation continue les plus adaptées peuvent se diviser en deux familles : les méthodes de recherche linéaire et les méthodes de région de confiance. Celles-ci se distinguent principalement par leur processus d'exploration à la recherche d'un point stationnaire. Les sous-sections qui suivent offrent un aperçu de ces processus.

1. SuiteSparse Matrix Collection : https ://sparse.tamu.edu

1.2.1 Convergence des méthodes de recherche linéaire

Une première famille de méthodes regroupe celles de type recherche linéaire. Il s'agit de méthodes qui, étant donné une estimation x_k, cherchent une direction s_k et une longueur de pas $t_k > 0$ tels que $f(x_k + t_k s_k) < f(x_k)$. Les méthodes de type Newton inexact sont souvent utilisées dans ce contexte. Le critère de convergence globale couramment utilisé est $\|g_k\| \leq \epsilon_a + \epsilon_r \|g_0\|$ où $g_k = \nabla f(x_k)$ et ϵ_a et ϵ_r désignent des tolérances absolue et relative. La convergence locale est assurée en utilisant le critère d'arrêt

$$\|r_k\| \leq \eta_k \|g_k\|, \tag{1.11}$$

où $r_k = H_k s_k + g_k$ est le résidu et s_k est une solution approchée de (1.3).

Chaque itération de recherche linéaire nécessite la résolution d'un sous-problème afin de trouver s_k. Il correspond à la CN1 de q, formulée sans les indices k par (1.3), d'où la nécessité d'utiliser une méthode de résolution de systèmes linéaires.

CG est utilisée afin de résoudre (1.3) lorsque la matrice est définie positive. Elle permet de faire décroître la norme du résidu comme le requiert la condition d'arrêt (1.11), bien que cette décroissance ne soit pas monotone (voir figure 1.1).

La performance de CR dans le calcul des directions en recherche linéaire n'est pas connue. Cependant, CR permet également de faire décroître la norme du résidu mais de manière monotone d'une itération à la suivante, comme l'indiquent la propriété (1.9) et la figure 1.1. Cette décroissance peut se montrer avantageuse lorsqu'on dispose d'un nombre limité d'itérations par exemple. De plus, elle pousse à penser qu'il serait pertinent d'utiliser CR dans les méthodes de type Newton inexact en recherche linéaire, mais également que CR pourrait se montrer plus performante que CG, d'où le désir d'étudier son comportement pour la recherche linéaire.

1.2.2 Convergence des méthodes de région de confiance

Les méthodes de région de confiance recherchent, à l'itération k, un pas s_k afin de mettre à jour l'estimation de la solution x_* et de sorte que $f(x_k + s_k) < f(x_k)$. Elles se différencient principalement de celles de recherche linéaire par le fait qu'elles recherchent un pas optimal en explorant toutes les directions de l'espace, mais dans un rayon limité. Elles se nomment ainsi car l'utilisateur d'un algorithme de région de confiance doit fournir un rayon $\Delta_k > 0$ qui correspond à la norme maximale admissible pour s_k. Nous verrons plus tard que ce rayon peut être modifié à chaque itération en fonction de la décroissance obtenue sur la valeur de

la fonction objectif. Une autre distinction majeure est que le pas calculé peut être rejeté s'il ne produit pas une décroissance jugée assez importante de l'objectif vis-à-vis du modèle quadratique, ce qui entraîne alors le calcul d'un nouveau pas.

La **condition de décroissance suffisante** impose l'existence d'une constante $\kappa \in (0, 1)$, qui ne dépend pas de k, telle que

$$q_k(s_k) \leq -\kappa \|g_k\| \min\left(\frac{\|g_k\|}{1+\|H_k\|}, \Delta_k\right) \tag{1.12}$$

où $q_k(s_k) = g_k^T s_k + \frac{1}{2} s_k^T H_k s_k$.

La convergence globale d'une méthode de région de confiance (Conn *et al.*, 2000) est assurée si, à chaque itération k, s_k satisfait (1.12). Ici encore, CG est souvent utilisée et la raison en est la propriété (1.5) de décroissance monotone de q_k. Cependant, il s'agit d'une propriété également partagée par CR d'après (1.7), mais cette dernière méthode n'est pas utilisée dans ce contexte non plus. On veut alors évaluer sa performance en région de confiance et la comparer à celle de CG.

Néanmoins, que ce soit en recherche linéaire ou en région de confiance, les hessiens doivent être définis positifs afin de pouvoir utiliser les algorithmes de base de CG et CR.

1.2.3 Le cas de la courbure négative

Si $H_k = \nabla^2 f(x_k)$ et que la fonction objectif est strictement convexe, alors $H_k \succ 0$ quel que soit k. En revanche si l'objectif présente de la courbure négative, c'est-à-dire s'il n'est pas convexe, alors il se peut que H_k ne soit pas définie positive pour une itération k.

Une extension de CG pour traiter ces cas de courbures négatives en recherche linéaire est décrite par Dembo et Steihaug (1983). Une modification pour la région de confiance existe également et est proposée par Steihaug (1983). De telles modifications ne sont pas décrites dans la littérature pour CR, ce qui constitue la dernière motivation de ce travail.

Ces observations nous permettent de définir plus précisément les objectifs de recherche.

1.3 Objectifs de recherche

Ce document vise à établir la viabilité de CR à déterminer les directions en optimisation non linéaire continue, autant au sein de méthodes de recherche linéaire qu'en régions de confiance, que l'objectif soit convexe ou non. Il s'agit là de combler un trou dans la littérature. En effet, tandis que des variantes de CG sont proposées depuis 1983 en recherche linéaire et en régions de confiance pour la résolution de problèmes non convexes, peu de travaux ont été publiés

sur CR et aucune modification dans ces contextes n'a été proposée.

Ainsi, les objectifs de la recherche sont, dans un premier temps, de fournir des algorithmes modifiés de CR offrant une extension à la résolution de problèmes non convexes et pouvant être utilisés dans des contextes de recherche linéaire et de région de confiance.

Il s'agit ensuite d'étudier le comportement de CR dans le calcul des directions en optimisation continue, incluant des résultats théoriques de convergence locale et globale.

Enfin, on souhaite comparer la performance de CR à celle de la méthode éprouvée qu'est CG et évaluer la pertinence de son utilisation à des fins de résolution de problèmes sans contraintes. Pour ce faire, on utilise des problèmes de bibliothèques couramment utilisées dans le domaine.

1.4 Plan du mémoire

La suite de ce mémoire comporte quatre chapitres et une conclusion. Tout d'abord, nous présentons une courte revue de littérature permettant de se familiariser avec les méthodes utilisées dans le cadre de la recherche. Ainsi, au chapitre 2 nous introduisons CG et CR et leurs inscriptions dans des méthodes de recherche linéaire et de région de confiance. Nous formalisons également les problèmes aux moindres carrés non linéaires et présentons les variantes de CG et CR qui leur sont adaptées. Puis, au chapitre 3, nous expliquons la démarche adoptée pour cette étude. Le chapitre 4 résume par un article le travail de recherche et présente les résultats théoriques et expérimentaux obtenus. Une discussion générale de l'étude et des résultats obtenus est faite dans le chapitre 5. Nous discutons des tests de détection de courbure négative ou nulle et expliquons pourquoi ils diffèrent dans des contextes de recherche linéaire et de région de confiance. Nous expliquons aussi le choix de certains paramètres et des critères de performance utilisés. Enfin nous concluons cette étude par une synthèse de celle-ci. Nous présentons aussi quelques limites de nos travaux et proposons des pistes d'améliorations futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous effectuons une revue de littérature des concepts et méthodes utilisés dans ce mémoire. Nous décrivons les deux types de méthodes pour l'optimisation sans contraintes et leurs sous-problèmes, mais également les méthodes de Krylov CR et CG utilisées dans le cadre de la recherche. Enfin nous présentons les problèmes aux moindres carrés non linéaires et les variantes de CR et CG qui leur sont adaptées.

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction de classe C^2 qu'on souhaite minimiser. On considère le problème d'optimisation

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.1}$$

La résolution de (2.1) peut se faire à l'aide d'une méthode itérative de type recherche linéaire ou région de confiance.

2.1 Les familles de méthodes d'optimisation

Les méthodes d'optimisation continue sans contraintes les plus adaptées pour $f \in C^2$ se divisent en deux familles : les méthodes de recherche linéaire et les méthodes de région de confiance. Bien que le problème d'optimisation soit le même, chaque famille résout un sous-problème qui lui est propre.

2.1.1 Les méthodes de recherche linéaire

À chaque itération, un algorithme de recherche linéaire choisit une direction de descente s_k et une longueur de pas $t_k > 0$ tels que $f(x_k + t_k s_k) < f(x_k)$. La résolution du problème $\min_t f(x_k + ts_k)$ peut être effectuée afin de trouver un t_k optimal mais est en pratique très coûteuse. L'algorithme génère plutôt un certain nombre de longueurs de pas jusqu'à en trouver une qui respecte un certain critère. On réfère souvent aux *conditions de Wolfe* :

$$f(x_k + t_k s_k) \leq f(x_k) + \alpha t_k \nabla f(x_k)^T s_k \tag{2.2}$$

$$\nabla f(x_k + t_k s_k)^T s_k \geq \beta \nabla f(x_k)^T s_k \tag{2.3}$$

avec $0 < \alpha < \beta < 1$. L'inégalité (2.2) est également utilisée toute seule, elle se nomme *critère d'Armijo* et assure une décroissance suffisante. Numériquement, une longueur de pas satisfaisant (2.2) est calculée par essais à partir d'une estimation initiale t. Si celle-ci vérifie le critère d'Armijo alors elle est retenue. Sinon, on la divise par un certain facteur tant que

le critère n'est pas respecté. Cette recherche est de type « backtracking », qu'on pourrait traduire par recherche « en arrière », et est résumée par l'algorithme 3. Ce dernier est assuré de trouver une longueur de pas satisfaisant (2.2) par le théorème 2.1.1.

Théorème 2.1.1. *Soient f de classe C^2 et $\alpha \in (0,1)$. Si $\nabla f(x_k)^T s_k < 0$, alors il existe $z > 0$ tel que pour tout $t \in (0,z]$, l'inégalité (2.2) est respectée en remplaçant t_k par t.*

Algorithme 3 : Recherche d'Armijo « en arrière »

Initialiser $t > 0$, $\gamma > 1$
tant que (2.2) n'est pas satisfait **faire**
    $t \leftarrow t/\gamma$
Retourner t

À l'itération k, le sous-problème d'une méthode de recherche linéaire consiste en la résolution de (1.3) auquel on ajoute des indices k, afin de calculer le pas s_k. Il est donc nécessaire de recourir à une méthode de résolution de systèmes linéaires pour calculer s_k.

L'une des méthodes de recherche linéaire les plus utilisées est la méthode de Newton. L'algorithme 4 décrit la méthode de Newton utilisée avec le critère d'Armijo. Plus de détails sur les méthodes de recherche linéaire sont disponibles dans (Nocedal et Wright, 1999).

Algorithme 4 : Méthode de Newton

Paramètres : $\epsilon_a > 0$, $\epsilon_r > 0$
Calculer $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$
Initialiser $k = 0$
tant que $\|g_k\| > \epsilon_a + \epsilon_r \|g_0\|$ **faire**
    Choisir $H_k = H_k^T \approx \nabla^2 f(x_k)$ et calculer une solution approchée s_k de (1.3) qui est une direction de descente pour f
    Calculer $t_k > 0$ satisfaisant (2.2)
    $x_{k+1} = x_k + t_k s_k$
    $k \leftarrow k + 1$
Retourner x_k

Une approximation du hessien en x_k est souvent utilisée, par exemple lorsqu'il n'est pas disponible, ou encore lorsque son évaluation est coûteuse ; on parle alors de méthode quasi-Newton.

2.1.2 Les méthodes de région de confiance

Dans les méthodes de région de confiance, on cherche une direction s_k en explorant toutes les directions possibles dans une région de rayon $\Delta_k > 0$, centrée en x_k et appelée *région*

de confiance. Ce nom provient du fait qu'on estime que le modèle quadratique de f est une bonne approximation de l'objectif dans cette région et qu'on peut alors lui faire confiance. À chaque itération, le pas s_k est calculé en résolvant le sous-problème de région de confiance :

$$\begin{aligned} \min_{s\in\mathbb{R}^n} \quad & q_k(s) = g_k^T s + \tfrac{1}{2} s^T H_k s \\ \text{s.c.} \quad & \|s\| \leq \Delta_k. \end{aligned} \tag{2.4}$$

En négligeant la contrainte de région de confiance, la CN1 appliquée à q_k donne (1.3) avec les indices k. Si la solution de (2.4) se trouve à l'intérieur de la région de confiance, alors la solution à ce problème est la même que celle du système (1.3).

Pour $g_k \neq 0$, on définit le *point de Cauchy* comme $s_c = -\alpha g_k$ avec α la solution du problème :

$$\begin{aligned} \min_{\alpha} \quad & q_k(-\alpha g_k) = -\alpha \|g_k\|^2 + \tfrac{1}{2}\alpha^2 g_k^T H_k g_k \\ \text{s.c.} \quad & 0 \leq \alpha \leq \Delta_k / \|g_k\|. \end{aligned} \tag{2.5}$$

Il s'agit du pas qui minimise q_k dans la direction de la plus forte pente $-g_k$, sous la contrainte de région de confiance.

Théorème 2.1.2. *(Conn* et al.*, 2000, théorèmes 6.4.4 et 6.4.6). Soit $f \in C^2$ bornée inférieurement et telle que $\|\nabla^2 f(x)\|$ est uniformément borné pour tout x. Si* (1.12) *est vérifiée, alors*

$$\lim_{k\to\infty} \|g_k\| = 0.$$

Le théorème 2.1.2 établit que la convergence d'une méthode de région de confiance est assurée si la condition de décroissance suffisante (1.12) est satisfaite. Celle-ci impose d'atteindre une portion de la décroissance obtenue avec le point de Cauchy. Plus de détails sont donnés par Conn *et al.* (2000, §6.3.4).

Lors d'une itération de région de confiance, le pas calculé peut être rejeté s'il n'est pas considéré assez bon et le rayon de la région de confiance peut être adapté. Pour cela on choisit des paramètres $0 < \eta_1 \leq \eta_2 < 1$ pour évaluer la qualité de s_k et $0 < \gamma_1 \leq \gamma_2 < 1$ pour adapter Δ_k. Un indicateur de la qualité de s_k est le ratio

$$\sigma_k = \frac{f(x_k) - f(x_k + s_k)}{q_k(x_k) - q_k(x_k + s_k)}. \tag{2.6}$$

Itération rejetée

Si σ_k est petit, cela signifie que la décroissance de f à partir du point x_k est petite devant celle de q_k. On présume que q_k ne représente pas f suffisamment bien dans un rayon Δ_k autour de x_k. Le rayon est réduit afin que l'approximation soit plus fine et on calcule un nouveau pas. L'itération n'est pas réussie. D'après Conn *et al.* (2000, théorème 6.4.2), il ne peut y avoir qu'un nombre fini d'itérations rejetées, ce qui garantit l'acceptation d'un pas après un nombre fini d'itérations.

Itération acceptée

Si la décroissance de f est convenable comparée à celle de q_k, alors q_k est une assez bonne approximation de f dans la région de confiance considérée : Δ_k peut soit être maintenu à sa valeur, soit être augmenté dans l'espoir de faire de plus grands pas et que le modèle approche toujours bien f. On parle d'itération réussie.

Si σ_k est proche de 1, voire supérieur à 1, la décroissance de l'objectif est très bonne relativement à celle du modèle quadratique. On peut alors imaginer que q_k est une bonne approximation sur une région de plus grand rayon et donc augmenter Δ_k. Dans ce cas, on parle d'itération très réussie.

L'algorithme

À partir de Conn *et al.* (2000), on décrit une procédure de région de confiance à travers l'algorithme 5.

2.2 Le calcul des pas

2.2.1 Les méthodes de Krylov

Les méthodes que nous utilisons pour résoudre les problèmes d'optimisation impliquent la résolution de systèmes linéaires qui sont parfois de grande taille. Il est alors avantageux de disposer de méthodes peu coûteuses qui limitent la mémoire utilisée pour le stockage des données du problème.

Les méthodes de Krylov sont des méthodes itératives de résolution de systèmes linéaires

$$Ax = b, \tag{2.7}$$

avec A une matrice carrée de taille $n \in \mathbb{N}$, b un vecteur de $\mathbb{R}^n$ et $x \in \mathbb{R}^n$ le vecteur solution

Algorithme 5 : Méthode de région de confiance

Paramètres : $\Delta_0 > 0$, $0 < \eta_1 \leq \eta_2 < 1$, $0 < \gamma_1 \leq \gamma_2 < 1$, $\epsilon_a > 0$, $\epsilon_r > 0$
Calculer $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$
tant que $\|g_k\| > \epsilon_a + \epsilon_r \|g_0\|$ **faire**
  Choisir $H_k = H_k^T \approx \nabla^2 f(x_k)$
  Calculer s_k solution approchée de (2.4)
  Évaluer σ_k avec (2.6)
  si $\sigma_k < \eta_1$ **alors** {itération non réussie}
    $x_{k+1} = x_k$, $f_{k+1} = f_k$, $g_{k+1} = g_k$
    Choisir $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$
  sinon {itération réussie}
    $x_{k+1} = x_k + s_k$
    Calculer $f_{k+1} = f(x_{k+1})$, $g_{k+1} = \nabla f(x_{k+1})$
    si $\sigma_k \geq \eta_2$ **alors** {itération très réussie}
      Choisir $\Delta_{k+1} \in [\Delta_k, \infty)$
    sinon
      Choisir $\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k]$
  $k \leftarrow k + 1$
Retourner x_k

recherché. Elles ont la particularité, contrairement aux méthodes directes, de n'utiliser aucune factorisation matricielle et sont donc à privilégier pour des systèmes de taille importante ou pour les cas où A n'est pas disponible explicitement. Ces méthodes résolvent donc également (1.3) et nous utilisons cette dernière formulation dans un souci de cohérence avec le reste du document.

Ces méthodes génèrent des itérés s_k dans des *espaces de Krylov*. Le $k^{\text{ième}}$ espace de Krylov associé à H et g est le sous-espace généré par les puissances de 0 à $k-1$ de H multiplié par g :

$$K_k = vect\{g, Hg, H^2 g, \ldots, H^{k-1} g\}.$$

Les sous-sections qui suivent présentent les deux principales méthodes de Krylov mentionnées dans ce mémoire.

2.2.2 La méthode du gradient conjugué

La méthode du gradient conjugué (CG) est une méthode de Krylov introduite par Hestenes et Stiefel (1952) et qui peut être construite à partir du processus de Lanczos (1950) comme le proposent Paige et Saunders (1975). Elle résout (1.3) lorsque $H = H^T \succ 0$. De par cette

propriété de H, la solution de (1.3) est aussi celle de

$$\min_{s\in\mathbb{R}^n} q(s) = g^T s + \tfrac{1}{2} s^T H s. \tag{2.8}$$

En effet, q est alors strictement convexe et son unique minimum s_* vérifie (1.3) par la CN1.

À chaque itération $k \geq 1$, CG minimise q dans le $k^{\text{ième}}$ espace de Krylov (Hestenes et Stiefel, 1952, théorème 4.3). Autrement dit, pour $s_0 = 0$, s_k est solution de

$$\min_{s\in K_k} q(s). \tag{2.9}$$

Ceci a pour conséquence qu'analytiquement, pour $H \succ 0$, l'algorithme se termine en maximum n itérations.

La méthode génère des directions de recherche p_i, $i = 0, 1, 2, \ldots$ qui sont conjuguées par rapport à H, autrement dit $p_i^T H p_j = 0$ pour $i \neq j$. Les résidus $r_i = -g - H s_i = -\nabla q(s_i)$ sont orthogonaux, c'est-à-dire que $r_i^T r_j = 0$ pour $i \neq j$. La conjugaison des directions de recherche par rapport à H est assurée par un facteur β_k lors de la mise à jour $p_k = r_k + \beta_k p_{k-1}$. De plus, ces directions sont générées dans K_{k+1}.

À l'itération k, l'algorithme met à jour l'estimé $s_k = s_{k-1} + \alpha_k p_{k-1}$ en utilisant une longueur de pas α_k qui minimise q dans la direction p_{k-1}. En effet, puisque q est quadratique, son développement de Taylor de degré 2 est exact et on a :

$$q(s_{k-1} + \alpha p_{k-1}) = q(s_{k-1}) - \alpha r_{k-1}^T p_{k-1} + \tfrac{1}{2}\alpha^2 p_{k-1}^T H p_{k-1}.$$

La CN1 appliquée à cette expression par rapport à la variable α donne $\alpha_k = \frac{r_{k-1}^T p_{k-1}}{p_{k-1}^T H p_{k-1}}$. À la première itération, sachant que $p_0 = r_0 = -g$, cette longueur de pas mène au point de Cauchy.

Hestenes et Stiefel (1952) expliquent que $r_{k-1}^T p_{k-1} = \|r_{k-1}\|^2$, ce qui est utilisé dans la description de l'algorithme 1. Celui-ci présente CG telle que décrite dans l'article de Hestenes et Stiefel (1952). En pratique, la boucle `Pour` est souvent remplacée par une boucle `Tant que` et le critère d'arrêt est de la forme $\|r_k\| \leq \tau_a + \tau_r \|g\|$ avec $\tau_a > 0$ et $\tau_r > 0$ des tolérances absolue et relative. Quelques propriétés de CG sont données dans le résultat suivant.

Théorème 2.2.1. *(Hestenes et Stiefel, 1952, théorème* 5:1*) et (Steihaug, 1983, théorème 2.1). Si on appelle l la dernière itération effectuée par CG, c'est-à-dire telle que $r_l = 0$, les*

propriétés suivantes sont satisfaites par l'algorithme 1 pour $H \succ 0$, si $s_0 = 0$:

$$\alpha_k > 0, \qquad k = 1, 2, \ldots, l \tag{2.10}$$

$$\|s_{k+1}\| > \|s_k\|, \qquad k = 0, 1, \ldots, l-1 \tag{2.11}$$

$$r_i^T p_j = 0, \qquad i > j, \qquad i = 1, 2, \ldots, l \text{ et } j = 0, 1, \ldots, i-1 \tag{2.12}$$

$$r_i^T p_j = \|r_i\|^2, \qquad i \leq j, \qquad i, j = 0, 1, \ldots, l \tag{2.13}$$

$$r_k^T p_k > 0, \qquad k = 0, 1, \ldots, l-1 \tag{2.14}$$

$$g^T s_k < 0, \qquad k = 1, 2, \ldots, l. \tag{2.15}$$

Démonstration. À l'itération k, $r_{k-1} \neq 0$ sinon l'algorithme s'arrête car le système est résolu. Puisque $H \succ 0$, alors $p_{k-1}^T H p_{k-1} > 0$ et, par définition de α_k, la propriété (2.10) est vérifiée. La propriété (2.11) est prouvée par Steihaug (1983, théorème 2.1). Les égalités (2.12) et (2.13) proviennent de Hestenes et Stiefel (1952, théorème 5:1). Puisque $r_k \neq 0$, la propriété (2.14) en découle directement.

Enfin, par construction on a $s_k = \sum_{i=1}^{k} \alpha_i p_{i-1}$. Ainsi,

$$\begin{aligned} g^T s_k &= -r_0^T s_k \\ &= -\sum_{i=1}^{k} \alpha_i r_0^T p_{i-1} \\ &= -\sum_{i=1}^{k} \alpha_i \|r_0\|^2 && \text{d'après (2.13)} \\ &< 0 && \text{d'après (2.10).} \end{aligned}$$

□

Notons que $\nabla q(s_k) = -r_k$ dans CG, (2.14) montre alors que p_k est toujours une direction de descente pour la quadratique. De plus, (2.15) établit que pour tout $1 \leq k \leq l$, s_k est une direction de descente pour f.

2.2.3 La méthode des résidus conjugués

La méthode des résidus conjugués (CR) a été introduite en même temps que CG par Hestenes et Stiefel (1952), bien qu'elle ne portait alors pas encore son nom. Stiefel (1955) la décrit

plus tard sous le nom maintenant connu. Il s'agit également d'une méthode de Krylov pour résoudre (1.3) lorsque $H \succ 0$. CR possède de nombreuses similitudes, notamment en termes d'écriture de l'algorithme, avec CG (voir algorithmes 1 et 2). Son équivalent, en arithmétique exacte, construit à partir du processus de Lanczos (1950) est MINRES (Paige et Saunders, 1975). Cette dernière méthode est numériquement plus stable et peut être utilisée lorsque H n'est pas définie positive.

Les résidus dans CR sont définis comme dans CG. Mais cette fois-ci, ce sont eux qui sont construits de sorte à être conjugués par rapport à H et on a donc : $r_i^T H r_j = 0$ pour $i \neq j$. Le nom de la méthode provient de cette propriété. Les directions de recherche sont désormais conjuguées, non plus par rapport à H, mais à H^2. L'algorithme débute avec une estimation initiale s_0 de la solution et la met à jour à chaque itération k : $s_k = s_{k-1} + \alpha_k p_{k-1}$.

La principale différence avec CG réside dans la quantité minimisée par s_k à chaque itération. En effet, CR choisit s_k pour minimiser la norme du résidu (Stiefel, 1955), de sorte qu'il soit solution de

$$\min_{s \in K_k} \tfrac{1}{2}\|r(s)\|^2, \text{ avec } r(s) = -(g + Hs).$$

C'est de ce fait que provient la propriété de décroissance monotone de la norme du résidu (voir figure 1.1). La longueur de pas α_k est choisie pour minimiser

$$\begin{aligned} \tfrac{1}{2}\|r(s_k)\|^2 &= \tfrac{1}{2}\|g + H(s_{k-1} + \alpha_k p_{k-1})\|^2 \\ &= \tfrac{1}{2}\|\alpha_k H p_{k-1} - r_{k-1}\|^2. \end{aligned}$$

En développant puis en dérivant par rapport à α_k, la CN1 donne $\alpha_k = \frac{r_{k-1}^T H p_{k-1}}{\|H p_{k-1}\|^2}$. En utilisant la définition $p_{k-1} = r_{k-1} + \beta_{k-1} p_{k-2}$ et (Fong, 2011, théorème 2.1.4), montrant que $r_i^T H p_j = 0$ pour $i > j$, la longueur du pas devient : $\alpha_k = \frac{r_{k-1}^T H r_{k-1}}{\|H p_{k-1}\|^2}$.

L'algorithme 2 décrit CR. Il est facile d'observer la similitude avec CG. À l'exception des formules de mise à jour de α_k et β_k, l'écriture des algorithmes 1 et 2 est identique. Ici également, en pratique, la boucle `Pour` est souvent remplacée par une boucle `Tant que` et le critère d'arrêt est de la forme $\|r_k\| \leq \tau_a + \tau_r \|g\|$ avec $\tau_a > 0$ et $\tau_r > 0$ des tolérances absolue et relative.

2.2.4 Adaptation de CG à la courbure négative

Nous avons établi plus haut le lien entre la résolution de (1.3) et la minimisation de q. Si f n'est pas convexe alors elle présente de la courbure nulle ou négative, c'est-à-dire qu'il existe au moins un point $x \in \mathbb{R}^n$ tel que $v^T \nabla^2 f(x) v \leq 0$ pour un certain vecteur $v \in \mathbb{R}^n$. Si H

est le hessien de f en x, il n'est donc pas défini positif et le modèle quadratique en ce point n'est donc pas convexe. Les algorithmes 1 et 2 ne peuvent alors pas être utilisés tels quels. Cependant, des modifications de CG existent, autant en recherche linéaire qu'en région de confiance, afin de faire face à ces situations où H est seulement symétrique et c'est ce que nous décrivons dans les parties qui suivent.

Adaptation de CG en recherche linéaire

On considère toujours (1.3) mais tel que H n'est pas nécessairement définie positive. Dembo et Steihaug (1983) proposent une modification de CG qui consiste en la vérification du type de courbure le long de la direction de recherche. À chaque itération k, on regarde le signe de $p_{k-1}^T H p_{k-1}$. L'idée générale est que lorsqu'une courbure négative est trouvée dans une direction de recherche, l'algorithme s'arrête en retournant l'approximation trouvée à l'itération précédente.

La version proposée a pour source Dembo et Steihaug (1983, Theorème 2.4) qui établit que si H n'est pas définie positive, alors soit $p_k^T H p_k \leq 0$ à une certaine itération k de CG, soit g est orthogonal aux sous-espaces propres associés aux valeurs propres négatives ou nulles. De plus, Dembo et Steihaug (1983, Lemme A.2) montrent que s_k est une direction de descente pour f en x tant que $p_{k-1}^T H p_{k-1} > 0$. Mais alors si $p_0^T H p_0 \leq 0$, il faut choisir une direction de descente à retourner puisque $s_0 = 0$. La direction $-g$ est choisie puisque $-g^T \nabla f(x) = -\|g\|^2 < 0$ et il s'agit bien d'une direction de descente.

Appelons l la dernière itération de CG modifiée, la sortie s de l'algorithme peut se résumer ainsi :

$$s = \begin{cases} -g & \text{si } p_{l-1}^T H p_{l-1} \leq 0 \text{ et } l = 1 \\ s_{l-1} & \text{si } p_{l-1}^T H p_{l-1} \leq 0 \text{ et } l \neq 1 \\ s_l & \text{sinon.} \end{cases}$$

L'algorithme 6 décrit cette modification. Comme pour l'algorithme 1, le critère d'arrêt utilisé en pratique porte sur la norme du résidu. De plus, la condition $p_{k-1}^T H p_{k-1} \leq 0$ est remplacée par $p_{k-1}^T H p_{k-1} \leq \epsilon \|p_{k-1}\|^2$ avec $\epsilon > 0$ une constante, comme l'expliquent Dembo et Steihaug (1983).

Algorithme 6 : CG modifiée pour la recherche linéaire

1: Initialiser $s_0 = 0$, $r_0 = -g$, $p_0 = r_0$
2: **pour** $k = 1, 2, 3, \ldots$ **faire**
3: **si** $p_{k-1}^T H p_{k-1} \leq 0$ **alors**
4: **si** $k = 1$ **alors**
5: **Retourner** $-g$
6: **sinon**
7: **Retourner** s_{k-1}
8: $\alpha_k = \frac{\|r_{k-1}\|^2}{p_{k-1}^T H p_{k-1}}$
9: $s_k = s_{k-1} + \alpha_k p_{k-1}$
10: $r_k = r_{k-1} - \alpha_k H p_{k-1}$
11: $\beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$
12: $p_k = r_k + \beta_k p_{k-1}$
13: **Retourner** s_k

Théorème 2.2.2. *Si H est seulement symétrique, tant que $p_{k-1}^T H p_{k-1} > 0$, les propriétés du théorème 2.2.1 sont satisfaites pour l'algorithme 6, jusqu'à l'itération k.*

Le théorème 2.2.2 vient du fait que tant que $p_{k-1}^T H p_{k-1} > 0$, la condition de la ligne 3 de l'algorithme 6 n'est pas satisfaite à l'itération k, et cet algorithme suit alors les étapes de l'algorithme 1.

Dans un souci de minimisation du modèle quadratique, il serait pourtant intéressant de pouvoir utiliser l'information de concavité puisqu'elle informe d'une décroissance du modèle. Cette information peut être exploitée si on dispose d'une contrainte de région de confiance.

Adaptation de CG en région de confiance

Dans un contexte de région de confiance, l'approche utilisée pour l'adaptation de CG se rapproche de celle utilisée en recherche linéaire par le fait que la courbure dans la direction de recherche est également vérifiée à chaque itération. Néanmoins, ici on peut utiliser, s'il y a lieu, la non-convexité du modèle et le fait que le rayon de recherche est limité, c'est ce que propose Steihaug (1983). Ainsi, lorsqu'une courbure négative est trouvée dans une direction, celle-ci est suivie jusqu'à la frontière de la région de confiance car le modèle est concave et donc décroît. L'algorithme 7 présente cette modification de CG pour les méthodes de région de confiance.

Algorithme 7 : CG modifiée pour la région de confiance

Initialiser $s_0 = 0$, $r_0 = -g$, $p_0 = r_0$
pour $k = 1, 2, 3, \ldots$ **faire**
  Calculer $\alpha_p > 0$ tel que $\|s_{k-1} + \alpha_p p_{k-1}\| = \Delta$
  si $p_{k-1}^T H p_{k-1} \leq 0$ **alors**
    Retourner $s_{k-1} + \alpha_p p_{k-1}$
  $\alpha_k = \frac{\|r_{k-1}\|^2}{p_{k-1}^T H p_{k-1}}$
  si $\alpha_k \geq \alpha_p$ **alors**
    Retourner $s_{k-1} + \alpha_p p_{k-1}$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k H p_{k-1}$
  $\beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$
  $p_k = r_k + \beta_k p_{k-1}$
Retourner s_k

Théorème 2.2.3. *Si H est seulement symétrique, tant que $p_{k-1}^T H p_{k-1} > 0$ et $\alpha_k \leq \alpha_p$, les propriétés du théorème 2.2.1 sont satisfaites pour l'algorithme 7, jusqu'à l'itération k.*

En effet, tant que $p_{k-1}^T H p_{k-1} > 0$ et $\alpha_k < \alpha_p$, l'algorithme 7 suit les étapes de l'algorithme 1. Si $p_{k-1}^T H p_{k-1} > 0$ et $\alpha_k = \alpha_p$, ceci est toujours vrai, à l'exception que l'algorithme 7 s'arrête après l'itération k et ne calculent donc pas r_k et p_k.

À l'itération k, si p_{k-1} s'avère être une direction de courbure négative, et s'il s'agit d'une direction de descente pour la quadratique, alors q décroît vers $-\infty$ dans cette direction. Puisque l'algorithme 7 s'arrête lorsqu'on rencontre une courbure négative, si on détecte une courbure négative dans la direction p_{k-1}, ceci implique que p_{k-2} est une direction de courbure strictement positive. D'après le théorème 2.2.3, on a alors $\alpha_{k-1} > 0$ et p_{k-1} est une direction de descente pour la quadratique.

Ainsi, si $p_{k-1}^T H p_{k-1} \leq 0$ alors q décroît vers $-\infty$ dans la direction p_{k-1}. Or, d'après (2.11), la norme des itérés croît de manière monotone dans CG, ce qui signifie que si $\|s_k\| > \Delta$, tous les itérés suivants seront également à l'extérieur de la région de confiance. Étant donnée la propriété de décroissance (1.6), Steihaug (1983) propose alors de suivre la direction de courbure négative p_{k-1} jusqu'à la frontière de la région de confiance. Si $H \succ 0$, il suffit de vérifier que l'itéré ne quitte pas la région de confiance. Soit $\Delta > 0$ le rayon de la région de confiance, appelons $\alpha_p > 0$ la longueur de pas telle que $\|s_{k-1} + \alpha_p p_{k-1}\| = \Delta$. Appelons l la dernière itération de CG modifiée. La sortie s de l'algorithme peut se résumer comme suit :

$$s = \begin{cases} s_{l-1} + \alpha_p p_{l-1} & \text{si } p_{l-1}^T H p_{l-1} \leq 0 \text{ ou } \alpha_l \geq \alpha_p \\ s_l & \text{sinon.} \end{cases}$$

Le résultat qui suit donne une borne inférieure sur le taux de décroissance de la quadratique au point renvoyé par l'algorithme 7.

Théorème 2.2.4. *(Yuan, 2000, théorème 2). Quelque soit $\Delta > 0$ le rayon de la région de confiance, si $H \succ 0$, soient s la solution renvoyée par l'algorithme 7 et s_* la solution globale de* (2.4) *(sans les indices), alors*

$$q(s) \leq \tfrac{1}{2} q(s_*). \tag{2.16}$$

Modification de Luenberger pour CR

Luenberger (1970) s'intéresse à la résolution, par CR, du problème à $m \leq n$ contraintes d'égalité linéaires

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & g^T s + \tfrac{1}{2} s^T H s \\ \text{s.c.} \quad & Bs = d \end{aligned}$$

où $H = H^T \succ 0$ est une matrice carrée de taille n, $g \in \mathbb{R}^n$, B est une matrice de taille $m \times n$, $d \in \mathbb{R}^m$ et s est le vecteur solution de $\mathbb{R}^n$ recherché. Ce problème peut se rapporter à la résolution du système linéaire

$$\begin{bmatrix} H & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} s \\ \lambda \end{bmatrix} = \begin{bmatrix} -g \\ d \end{bmatrix}$$

ou encore, de manière équivalente,

$$Ax = b,$$

avec A une matrice symétrique non singulière de taille $m + n$, toujours indéfinie, et x et b des vecteurs de $\mathbb{R}^{m+n}$.

Puisque la matrice est indéfinie, il se peut que $r_k^T A r_k = 0$ pour un certain k, donc que $\alpha_k = 0$ (voir ligne 3 de l'algorithme 2). Ceci impliquerait que β_{k+1}, défini à la ligne 6 de l'algorithme 2, n'est pas défini. Luenberger (1970) propose une modification de α_k lorsque celui-ci devient nul dans l'algorithme 2. Cette variante de CR est une alternative pour traiter le cas où la matrice n'est pas définie positive, mais ne s'inscrit pas dans un contexte où on souhaite retourner une direction de descente. Dans le cadre de l'optimisation sans contraintes, des modifications semblables à celles vues pour CG n'ont jamais été proposées pour CR. Elles font l'objet de l'article proposé au chapitre 3. Nous considérons une approche différente de Luenberger (1970), principalement parce que nous proposons des algorithmes qui vérifient la courbure du modèle quadratique et qui s'inscrivent dans des contextes de recherche linéaire et de région de confiance.

Si l'objectif est une fonction aux moindres carrés non linéaire, des adaptations de CG et CR peuvent être utilisées. Elles sont décrites dans la section qui suit.

2.3 Les problèmes aux moindres carrés non linéaires

Dans cette section, on s'intéresse à la minimisation sans contraintes d'un objectif aux moindres carrés non linéaires. Pour cela on s'intéresse à $F : \mathbb{R}^n \to \mathbb{R}^m$ composée de m fonctions non linéaires et ayant la forme $F(x) = [f_1(x)\ f_2(x)\ \ldots\ f_m(x)]^T$ où chaque f_i est de classe C^2. Le problème aux moindres carrés se formule

$$\min_{x\in\mathbb{R}^n} f(x) = \tfrac{1}{2}\|F(x)\|^2. \tag{2.17}$$

Puisque f est C^2, on peut définir son gradient et son hessien :

$$\begin{aligned}\nabla f(x) &= J(x)^T F(x),\\ \nabla^2 f(x) &= J(x)^T J(x) + \sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x).\end{aligned}$$

La méthode de Newton appliquée à (2.17) requiert $\nabla f(x_k)$ et $\nabla^2 f(x_k)$ à chaque itération k. Il faut alors également calculer les hessiens des f_i pour chaque estimation x_k. La méthode de Gauss-Newton approxime $\nabla^2 f(x) \approx J(x)^T J(x)$ dans la méthode de Newton. L'approche proposée par Levenberg (1944) et Marquardt (1963) utilise cette approximation mais dans un contexte de région de confiance. Elle consiste à utiliser le modèle de Gauss-Newton

$$q^{\mathrm{GN}}(s) = \tfrac{1}{2}\|J(x)s + F(x)\|^2. \tag{2.18}$$

Pour un rayon de région de confiance $\Delta > 0$, le sous-problème à résoudre à chaque itération est :

$$\begin{aligned}\min_{s\in\mathbb{R}^n}\quad & q^{\mathrm{GN}}(s)\\ \text{s.c.}\quad & \|s\| \le \Delta.\end{aligned}$$

L'application de la CN1 à (2.18) donne le système linéaire

$$J^T J s = -J^T F, \tag{2.19}$$

où J et F désignent respectivement $J(x)$ et $F(x)$. L'équation (2.19) est appelée *équation normale*. Si J est de rang plein, c'est-à-dire si l'image de l'application linéaire correspondante est de dimension m, alors $J^T J \succ 0$. Dans ce cas, les CG et CR classiques peuvent alors être utilisées. Si J n'est pas de rang plein, alors $J^T J$ est seulement semi-définie positive et le cas de la courbure nulle est possible pour q^{GN}. C'est la raison pour laquelle Levenberg et Marquardt proposent une méthode de région de confiance.

Néanmoins, il est possible de tirer avantage de la structure du problème. Les méthodes Conjugate Gradient method for Least Squares (CGLS) et Conjugate Residual method for Least Squares (CRLS) sont des adaptations respectives de CG et CR aux problèmes aux moindres carrés linéaires, qui offrent de meilleurs résultats que CGLS et CRLS en arithmétique flottante, notamment lorsque J est mal conditionnée. En effet, $cond(J^TJ) = cond(J)^2$ et donc J^TJ est d'autant plus mal conditionnée.

2.3.1 CGLS et LSQR

CGLS est une variante de CG décrite par Hestenes et Stiefel (1952) pour traiter l'équation normale (2.19). Son nom lui est plus tard donné par Paige et Saunders (1982). En arithmétique exacte, CGLS est exactement CG appliquée à (2.19). L'algorithme 8 décrit les étapes de la méthode.

Algorithme 8 : CGLS

Initialiser $s_0 = 0$, $r_0 = -F$, $p_0 = u_0$
pour $k = 1, 2, 3, \ldots$ **faire**
  $\alpha_k = \frac{\|J^T r_{k-1}\|^2}{\|Jp_{k-1}\|^2}$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k J p_{k-1}$
  $\beta_k = \frac{\|J^T r_k\|^2}{\|J^T r_{k-1}\|^2}$
  $p_k = J^T r_k + \beta_k p_{k-1}$
Retourner s_k

LSQR de Paige et Saunders (1982) est une méthode équivalente à CGLS en arithmétique exacte. Son implémentation diffère par l'utilisation du processus de bidiagonalisation de Golub et Kahan (1965) et se montre plus stable numériquement, d'où son utilisation en pratique plutôt que CGLS.

Björck *et al.* (1998) proposent des variantes de CGLS qui permettent de calculer J^Tr_k et Jp_k par récurrence, ce qui permet de ne pas avoir à calculer de produit par J ou J^T et ainsi réduire le coût de l'algorithme. Cependant, ils montrent aussi qu'un calcul récursif altère la stabilité de la méthode. La version classique de CGLS où le résidu n'est pas calculé récursivement a une stabilité et une précision comparables à celles de LSQR.

2.3.2 CRLS et LSMR

CRLS (Fong, 2011) est la variante de CR pour les problèmes aux moindres carrés linéaires. Bien qu'elle ne soit pas mentionnée sous ce nom, la plus vieille référence connue pour cette

méthode date de Björck (1979). CRLS possède également une version analytiquement équivalente construite à partir du processus de bidiagonalisation de Golub et Kahan (1965) et appelée LSMR (Fong et Saunders, 2011). Cette dernière méthode correspond en fait à l'application de MINRES à l'équation (2.19). LSMR est numériquement plus stable, ce qui lui vaut d'être préférée à CRLS pour des tests numériques. Nous présentons les étapes de CRLS à travers l'algorithme 9. La quantité $J^T J p_k$ peut être obtenue par récurrence (Fong, 2011), permettant de réduire le nombre de produits matrice-vecteur et donc le coût de l'algorithme. Des tests menés par Björck et Saunders (2017) montrent que la version classique de CRLS possède une stabilité et une précision inférieures à celles de LSMR. Ils proposent cependant une version de CRLS comparable à LSMR mais qui requiert un produit opérateur-vecteur supplémentaire par itération.

Algorithme 9 : CRLS

Initialiser $s_0 = 0$, $r_0 = -J^T F$, $p_0 = r_0$
pour $k = 1, 2, 3, \ldots$ **faire**
  $\alpha_k = \frac{\|Jr_{k-1}\|^2}{\|J^T J p_{k-1}\|^2}$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k J^T J p_{k-1}$
  $\beta_k = \frac{\|Jr_k\|^2}{\|Jr_{k-1}\|^2}$
  $p_k = r_k + \beta_k p_{k-1}$
Retourner s_k

CHAPITRE 3 DÉMARCHE ADOPTÉE

Ce mémoire a pour objectif d'étudier la performance de CR à calculer les directions au sein de méthodes de recherche linéaire et de région de confiance. Pour ce faire, nous fournissons des adaptations de CR à des matrices qui ne sont pas définies positives pour ces deux types de méthodes d'optimisation. Celles-ci sont décrites dans l'article présenté au chapitre 4.

Cet article présente les méthodes proposées pour la recherche linéaire, les régions de confiance, mais aussi les problèmes aux moindres carrés non linéaires. Il décrit aussi des propriétés clés de CR, établit des résultats de convergence locale et globale et compare les algorithmes proposés à CG. Le développement des algorithmes prend en compte les propriétés de la version classique de CR dans un souci de conservation de celles-ci, notamment la décroissance monotone du modèle quadratique et de la norme du résidu. Le contexte d'utilisation est aussi important, impliquant que le pas renvoyé par la méthode reste une direction de descente pour la fonction objectif au point courant. La performance est évaluée en utilisant des problèmes tirés de bibliothèques d'optimisation et en effectuant des comparaisons avec CG puisqu'il s'agit d'une méthode éprouvée. La modification pour les problèmes aux moindres carrés non linéaires est effectuée sur CRLS pour traiter les cas de courbure nulle. CRLS est traitée uniquement dans un contexte de région de confiance. Cependant, les tests sont effectués avec LSMR et LSQR par souci de stabilité numérique.

Le chapitre 5 est une discussion générale qui revient principalement sur des points évoqués dans l'article et explique le choix de certains critères et paramètres.

Enfin, nous concluons sur l'étude et donnons des pistes possibles de travaux futurs.

CHAPITRE 4 ARTICLE 1: THE CONJUGATE RESIDUAL METHOD IN LINESEARCH AND TRUST-REGION METHODS

THE CONJUGATE RESIDUAL METHOD IN
LINESEARCH AND TRUST-REGION METHODS

Marie-Ange Dahito and Dominique Orban

Manuscript submitted to *SIAM Journal on Optimization* (SIOPT)

Abstract. The minimum residual method (MINRES) of Paige et Saunders (1975), which is often the method of choice for symmetric linear systems, is a generalization of the conjugate residual method (CR), proposed by Hestenes et Stiefel (1952). Like the conjugate gradient method (CG), CR possesses properties that are desirable for unconstrained optimization, but is only defined for symmetric positive-definite operators. CR's main property, that it minimizes the residual, is particularly appealing in inexact Newton methods, typically used in a linesearch context. CR is also relevant in a trust-region context as it causes monotonic decrease of convex quadratic models (Fong et Saunders, 2012). We investigate modifications that make CR suitable, even in the presence of negative curvature, and perform comparisons on convex and nonconvex problems with the conjugate gradient method. We complete our investigation with an extension suitable for nonlinear least-squares problems. Our experiments reveal that CR performs as well as or better than CG, and mainly yields savings in operator-vector products.

4.1 Introduction

We consider the large-scale unconstrained problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x), \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice-continuously differentiable and possibly nonconvex. We wish to identify a first-order critical point of (4.1), i.e., $x_* \in \mathbb{R}^n$ such that $\nabla f(x_*) = 0$, using either a linesearch or a trust-region method. In a linesearch method, we compute a step as an approximate solution of

$$Hs = -g, \tag{4.2}$$

with $H = H^T \approx \nabla^2 f(x)$ and $g = \nabla f(x)$, while in a trust-region method, the step is an approximate solution of

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \; q(s) \quad \text{subject to } \|s\| \leq \Delta, \qquad q(s) := g^T s + \tfrac{1}{2} s^T H s, \tag{4.3}$$

where $\Delta > 0$ is the trust-region radius. We consider the inexact solution of (4.2) and (4.3) via an iterative method.

CG (Hestenes et Stiefel, 1952) has long been a workhorse in optimization because of its desirable properties. In particular, it is well suited for (4.2) because H is often forced to be positive definite so as to obtain a descent direction. Dembo et Steihaug (1983) develop modifications of CG to cope with directions of negative curvature in the context of inexact Newton methods that ensure global and fast local convergence. CG is also well suited for (4.3) because the value of q is monotonically decreasing along the CG iterations and the norm of the iterates is monotonically increasing. Thus if the iterates were to leave the trust region, they would never return. The k-th CG solves

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \; q(s) \quad \text{subject to } s \in K_k,$$

where $K_k = \text{Span}\{g, Hg, H^2 g, \ldots, H^{k-1} g\}$ is the k-th Krylov subspace. Steihaug (1983) describes truncated CG, in which directions of negative curvature are followed to the boundary of the trust region. Yuan (2000) shows that the approximate solution s^C identified by the truncated CG is such that $q(s^C) \leq \frac{1}{2} q(s^*)$, where s^* is a global solution of (4.3).

CR was introduced by Hestenes et Stiefel (1952) and Stiefel (1955), although the name is not mentioned in the first paper. Like CG, CR is a Krylov subspace method to solve (4.2) with positive definite H. At iteration k CR minimizes the residual norm $\|Hs + g\|_2$ in K_k. The search occurs along directions p_i that are conjugate with respect to H^2, while the residuals $r_i := -g - Hs_i$ are conjugate with respect to H, which explains the name of the method. In inexact Newton methods, we seek a step s such that $\|Hs + g\| \leq \tau \|g\|$ for a certain tolerance $\tau > 0$. Monotonicity of the residual norm seems like an appealing property to attain this stopping condition. Fong et Saunders (2012) establish that when H is positive definite, the values of q decrease monotonically along the CR iterations and the norm of the iterates is also monotonically increasing. Those observations lead us to believe that CR could also be well suited to computing steps in both linesearch and trust-region methods.

We describe modifications of CR analogous to those of CG to cope with directions of negative curvature in both linesearch and trust-region methods, and establish that global and local convergence properties are preserved.

Our numerical experiments on convex problems indicate that CR performs comparably to CG. CR exhibits a slight advantage over CG in terms of function, gradient and Hessian-vector evaluations on nonconvex problems, which makes it a viable general-purpose subproblem solver.

Paige et Saunders (1975) describe MINRES, which generates the same iterates as CR in the definite case but is more general in that it also solves indefinite systems. Because the implementation of MINRES is substantially more complicated than that of CR, we do not consider it in this paper. We note however that it should be possible to implement our modified versions of CR as part of MINRES.

The rest of this paper is organized as follows. In section 4.2, we introduce the basic CR and its main properties. Section 4.3 gives a variant of CR in a linesearch inexact-Newton context, corresponding global and local convergence results, and numerical comparisons against CG. In section 4.4, we study CR in a trust-region context, provide theoretical results ensuring global convergence, and report on numerical experience. In section 4.5, we provide a variant of CR suitable for the solution of nonlinear least-squares problems in a trust-region context, and present numerical comparisons with the corresponding variant of CG. Concluding remarks appear in section 4.6. Complete details of our numerical experiments are provided in the appendix.

Notation

The norm used throughout is the Euclidean norm. Uppercase Latin letters denote matrices, lowercase Latin letters denote vectors, and Greek letters denote scalars.

4.2 Derivation of CR

Luenberger (1970) and Fong (2011) establish properties of CR when H is symmetric positive definite. We base our description of CR on the pseudocode of Fong et Saunders (2012) with some modifications explained below.

The main idea behind CR is to solve (4.2) with H symmetric and positive definite by solving the equivalent problem

$$\underset{s\in\mathbb{R}^n}{\text{minimize}}\ \bar{q}(s), \qquad \bar{q}(s) := \tfrac{1}{2}\|\nabla q(s)\|^2 = \tfrac{1}{2}\|g + Hs\|^2. \tag{4.4}$$

Note that $\nabla\bar{q}(s) = H(g + Hs)$ and $\nabla^2\bar{q}(s) = H^2$. For any s, let $r := -g - Hs$ denote the residual of (4.2) so that $\nabla\bar{q}(s) = -Hr$. The algorithm starts from any s_0 and initializes

$r_0 = -g - Hs_0$. It is customary to set $s_0 := 0$ and $r_0 := -g$. Given an iterate s_k and a descent direction p_k, CR computes a steplength $\alpha_{k+1} > 0$ such that $s_{k+1} := s_k + \alpha_{k+1}p_k$ minimizes $\bar{q}$ in the direction p_k, i.e., such that $\nabla\bar{q}(s_{k+1})^T p_k = 0$. Thus by construction, CR produces monotonic $\|r_k\|$.

Because $\bar{q}$ is quadratic,

$$\nabla\bar{q}(s_{k+1}) = \nabla\bar{q}(s_k) + \alpha_{k+1}\nabla^2\bar{q}(s_k)p_k, \tag{4.5}$$

and

$$\nabla\bar{q}(s_{k+1})^T p_k = -r_k^T H p_k + \alpha_{k+1}p_k^T H^2 p_k,$$

so that

$$\alpha_{k+1} = r_k^T H p_k / p_k^T H^2 p_k.$$

As we show below in (4.15), $r_k^T H p_k = r_k^T H r_k$. Thanks to (4.5), the residual at s_{k+1} can be updated as $r_{k+1} = r_k - \alpha_{k+1}Hp_k$. The next search direction is defined as $p_{k+1} := r_{k+1} + \beta_{k+1}p_k$, where β_{k+1} is chosen so that $p_{k+1}^T H^2 p_k = 0$, i.e., $\beta_{k+1} = -r_{k+1}^T H^2 p_k / p_k^T H^2 p_k$. Luenberger (1970, Theorem 1) shows that a consequence of the choice of p_{k+1} is that the search directions are H^2-conjugate and that $\beta_{k+1} = r_{k+1}^T H r_{k+1} / r_k^T H r_k$.

Algorithm 10 describes the classical CR. It differs from the one presented by Fong et Saunders (2012) by the addition of $\rho_k = \|r_k\|^2$, $\nu_k = \|r_k\|$ and a stopping criterion with absolute and relative tolerances.

Algorithm 10 : CR for (4.2)

Require: H, g, $\tau_a > 0$, $\tau_r > 0$
Initialize: $k = 0$, $s_0 = 0$, $r_0 = -g$, $u_0 = Hr_0$, $\zeta_0 = r_0^T u_0$, $p_0 = r_0$, $q_0 = u_0$, $\rho_0 = r_0^T r_0$, $\nu_0 = \sqrt{\rho_0}$
while $\nu_k > \tau_a + \tau_r\|g\|$ **do**
  $k \leftarrow k + 1$
  $\alpha_k = \zeta_{k-1} / \|q_{k-1}\|^2$
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k q_{k-1}$
  $\rho_k = \rho_{k-1} - \alpha_k \zeta_{k-1}$ $\{\rho_k = \|r_k\|^2\}$
  $\nu_k = \sqrt{\rho_k}$
  $u_k = Hr_k$
  $\zeta_k = r_k^T u_k$ $\{\zeta_k = r_k^T H r_k\}$
  $\beta_k = \zeta_k / \zeta_{k-1}$
  $p_k = r_k + \beta_k p_{k-1}$
  $q_k = u_k + \beta_k q_{k-1}$ $\{q_k = Hp_k\}$
return s_k

Theorem 1 summarizes properties of CR that we use throughout the rest of this paper.

Theorem 1. *Assume H is positive definite in Algorithm 10 and $r_i \neq 0$ for $i = 0, \ldots, k$ in Algorithm 10. The following properties hold:*

$$p_i^T H^2 p_j = 0, \qquad i \neq j, \qquad i, j = 0, \ldots, k, \tag{4.6}$$

$$r_i^T H r_j = 0, \qquad i \neq j, \qquad i, j = 0, \ldots, k, \tag{4.7}$$

$$p_i^T H r_j = 0, \qquad 0 \leq i < j \leq k, \tag{4.8}$$

$$r_i^T H r_i > 0, \qquad i = 0, \ldots, k, \tag{4.9}$$

$$\text{Span}\{r_0, r_1, \ldots, r_k\} = \text{Span}\{g, Hg, \ldots, H^k g\}, \tag{4.10}$$

$$\|s_i\| < \|s_j\|, \qquad i < j, \qquad i, j = 0, \ldots, k, \tag{4.11}$$

$$\alpha_i > 0, \qquad i = 1, \ldots, k, \tag{4.12}$$

$$p_i^T r_j > 0, \qquad 0 \leq i, j \leq k. \tag{4.13}$$

Proof. Properties (4.6)–(4.8) come respectively from (Luenberger, 1970, Theorem 1 (a), (d) and (b)), (4.9) follows from the positive definiteness of H, (4.11) is proven by Fong (2011, Theorem 2.1.6), and (4.12)–(4.13) come from (Fong, 2011, Theorem 2.1.5).

By construction, s_i minimizes the norm of the residual in K_i. We show by induction that $r_i \in K_{i+1}$ and $p_i \in K_{i+1}$. Initially, $r_0 = p_0 = g \in K_1$. Let the assumption be satisfied for index i. We have $r_{i+1} = r_i - \alpha_{i+1} H p_i$. Since r_i and p_i are in K_{i+1}, we know $Hp_i \in K_{i+2}$ and so $r_{i+1} \in K_{i+2}$. Also, $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$ so $p_{i+1} \in K_{i+2}$, which establishes (4.10). □

Theorem 2. *Assume H is positive definite in Algorithm 10. For all $k \geq 0$,*

$$q_k = Hp_k, \tag{4.14}$$

$$\zeta_k = r_k^T q_k, \tag{4.15}$$

$$\rho_k = \|r_k\|^2, \tag{4.16}$$

$$\nu_k = \|r_k\|. \tag{4.17}$$

Proof. Equality (4.14) can be established by induction. Initially, $q_0 = u_0 = Hr_0 = Hp_0$. Now assume that the property holds for index k. Then, by recurrence,

$$q_{k+1} = u_{k+1} + \beta_{k+1} q_k = Hr_{k+1} + \beta_{k+1} Hp_k = H(r_{k+1} + \beta_{k+1} p_k) = Hp_{k+1}.$$

We have $r_k^T q_k = r_k^T (u_k + \beta_k q_{k-1})$. But (4.8) and (4.14) yield $r_k^T q_{k-1} = 0$, so $r_k^T q_k = r_k^T H r_k = \zeta_k$, which establishes (4.15).

Because $\rho_0 = r_0^T r_0 = \|r_0\|^2$, (4.16) holds for $k = 0$. Then, (4.8), (4.14), (4.15) and a recursion assumption yield

$$\|r_{k+1}\|^2 = r_{k+1}^T(r_k - \alpha_{k+1}q_k) = (r_k - \alpha_{k+1}q_k)^T r_k = \rho_k - \alpha_{k+1}\zeta_k = \rho_{k+1}.$$

Property (4.17) follows because $\nu_k = \sqrt{\rho_k}$. □

The following result guides the detection of nonpositive curvature in the next sections, and parallels a similar result for CG. Its proof is as in Dembo et Steihaug (1983) with directions d_k replaced by residuals r_k.

Theorem 3 (Dembo et Steihaug, 1983, Theorem A.5)**.** *Consider Algorithm 10. Let m be the number of distinct eigenvalues of H. If g has a nonzero projection on each eigenspace and if*

$$r_i^T H r_j = 0, \quad i, j = 0, \ldots, m-1, \ (i \neq j), \tag{4.18}$$

$$r_i^T H r_i > 0, \quad i = 0, \ldots, m-1, \tag{4.19}$$

$$\text{Span}\{r_0, \ldots, r_{m-1}\} = \text{Span}\{g, Hg, \ldots, H^{m-1}g\}, \tag{4.20}$$

then H is positive definite.

4.3 CR in a linesearch context

In this section, we are interested in solving (4.1) by way of a linesearch method. The main motivation is that because CR produces monotonic $\|r_k\|$, it appears suitable for computing steps in an inexact Newton scheme. The linesearch subproblem is described by (4.2). As f may be nonconvex, H may not be positive definite.

Motivated by saddle-point systems and constrained problems, Luenberger (1970) modifies CR to solve indefinite systems. His modification is active when Algorithm 10 computes $\alpha_k = 0$, which may happen if H is not positive definite, and coincides with the original CR in the positive definite case. A disadvantage of his approach is that one must decide numerically when α_k should be treated as zero. Our strategy differs in that we check the curvature and not the step length, in the spirit of the modified CG developed by Dembo et Steihaug (1983). We maintain all the properties of CR because our modification stops when negative curvature is encountered.

4.3.1 Linesearch CR algorithm

Let $\pi_i = p_i^T p_i$, $\rho_i = r_i^T r_i$ for $i = 0, 1, \ldots, k$ and $\epsilon > 0$ be a constant value. In Algorithm 11 we modify CR in the context of a linesearch method to solve (4.2) when H is not necessarily positive definite.

Algorithm 11 : Modified CR (linesearch version)

Require: H, g, $\tau_a > 0$, $\tau_r > 0$, $\epsilon > 0$
1: **Initialize:** $k = 0$, $s_0 = 0$, $r_0 = -g$, $u_0 = Hr_0$, $\zeta_0 = r_0^T u_0$, $p_0 = r_0$, $q_0 = u_0$, $\delta_0 = \zeta_0$, $\rho_0 = r_0^T r_0$, $\mu_0 = \rho_0$, $\pi_0 = \rho_0$, $\nu_0 = \sqrt{\rho_0}$
2: **while** $\nu_k > \tau_a + \tau_r \|g\|$ **do**
3: $k \leftarrow k + 1$
4: **if** $\delta_{k-1} \leq \epsilon \pi_{k-1}$ or $\zeta_{k-1} \leq \epsilon \rho_{k-1}$ {(near) negative curvature detected} **then**
5: **if** $k = 1$ **then**
6: **return** $-g$
7: **else**
8: **return** s_{k-1}
9: $\alpha_k = \zeta_{k-1} / \|q_{k-1}\|^2$
10: $s_k = s_{k-1} + \alpha_k p_{k-1}$
11: $r_k = r_{k-1} - \alpha_k q_{k-1}$
12: $\rho_k = \rho_{k-1} - \alpha_k \zeta_{k-1}$ $\{\rho_k = \|r_k\|^2\}$
13: $\nu_k = \sqrt{\rho_k}$ $\{\nu_k = \|r_k\|\}$
14: $u_k = Hr_k$
15: $\zeta_k = r_k^T u_k$ $\{\zeta_k = r_k^T H r_k\}$
16: $\beta_k = \zeta_k / \zeta_{k-1}$
17: $p_k = r_k + \beta_k p_{k-1}$
18: $\pi_k = \rho_k + 2\beta_k(\mu_{k-1} - \alpha_k \delta_{k-1}) + \beta_k^2 \pi_{k-1}$ $\{\pi_k = \|p_k\|^2\}$
19: $\mu_k = \rho_k + \beta_k(\mu_{k-1} - \alpha_k \delta_{k-1})$ $\{\mu_k = p_k^T r_k\}$
20: $q_k = u_k + \beta_k q_{k-1}$ $\{q_k = Hp_k\}$
21: $\delta_k = \zeta_k + \beta_k^2 \delta_{k-1}$ $\{\delta_k = p_k^T H p_k\}$
22: **return** s_k

The main difference with Algorithm 10 resides in the condition on line 4 and recursion for π_k. We first note that Theorem 1 continues to hold for Algorithm 11 for as long as the search directions and residuals are directions of positive curvature.

Corollary 1. *Let $\epsilon > 0$. If $p_i^T H p_i > \epsilon \pi_i$ and $r_i^T H r_i > \epsilon \rho_i$ for $i = 0, \ldots, k$. The properties of Theorem 1 hold for Algorithm 11 at iterations $i = 0, \ldots, k$.*

Because of (4.7), there will be an iteration k such that $r_k^T H r_k \leq 0$ if H is not positive definite and Algorithm 11 does not terminate earlier. Thus, the residual will eventually signal the presence of nonpositive curvature. However, because of (4.6), we must check the sign of

$p_k^T H p_k$ explicitly at each iteration to discover whether p_k is a direction of positive curvature or not. If nearly negative curvature is detected, either the previous iterate s_{k-1} is returned if $k \geq 2$ or $-g$ is returned if $k = 1$ as it is known to be a descent direction.

Theorem 4. *Let H be symmetric but not necessarily positive definite. Identities* (4.14)–(4.17) *continue to hold for Algorithm 11. The following equalities also hold:*

$$\delta_k = p_k^T H p_k, \tag{4.21}$$

$$\mu_k = p_k^T r_k, \tag{4.22}$$

$$\pi_k = \|p_k\|^2. \tag{4.23}$$

Proof. The proof of (4.14)–(4.17) is the same as in Theorem 2. We have $\delta_0 = \zeta_0$ and $\forall k \geq 1, \delta_k = \zeta_k + \beta_k^2 \delta_{k-1}$. By induction we can show that $\delta_k = p_k^T H p_k$. Indeed, as $p_0 = r_0$ then $\delta_0 = p_0^T H p_0$. Now suppose the property satisfied for $k \geq 0$. We then have

$$p_{k+1}^T H p_{k+1} = p_{k+1}^T q_{k+1} = (r_{k+1} + \beta_{k+1} p_k)^T (u_{k+1} + \beta_{k+1} q_k).$$

Since H is symmetric, $p_k^T u_{k+1} = r_{k+1}^T q_k = 0$ according to (4.8). This leads to

$$p_{k+1}^T H p_{k+1} = r_{k+1}^T u_{k+1} + \beta_{k+1}^2 p_k^T q_k = \zeta_{k+1} + \beta_{k+1}^2 \delta_k = \delta_{k+1}$$

Thus, (4.21) is established.

We have $\mu_0 = \rho_0 = \|r_0\|^2$. But $r_0 = p_0$ so (4.22) is satisfied at iteration 0. Suppose by induction that it is verified at iteration $k \geq 0$. Then

$$\begin{aligned} p_{k+1}^T r_{k+1} &= (r_{k+1} + \beta_{k+1} p_k)^T r_{k+1} \\ &= \rho_{k+1} + \beta_{k+1} p_k^T r_{k+1} \\ &= \rho_{k+1} + \beta_{k+1} p_k^T (r_k - \alpha_{k+1} q_k) \\ &= \rho_{k+1} + \beta_{k+1} \mu_k - \alpha_{k+1} \beta_{k+1} \delta_k \\ &= \mu_{k+1} \end{aligned}$$

which establishes (4.22).

Similarly, (4.23) can be shown by induction. First, $\pi_0 = \rho_0 = \|p_0\|^2$. If $k \geq 0$ satisfies the property then

$$p_{k+1}^T p_{k+1} = (r_{k+1} + \beta_{k+1} p_k)^T (r_{k+1} + \beta_{k+1} p_k) = \rho_{k+1} + 2\beta_{k+1} r_{k+1}^T p_k + \beta_{k+1}^2 \pi_k.$$

The update for r_{k+1} yields

$$r_{k+1}^T p_k = (r_k - \alpha_{k+1} q_{k-1})^T p_k = \mu_k - \alpha_{k+1}\delta_k.$$

Finally,

$$\pi_{k+1} = \rho_{k+1} + 2\beta_{k+1}(\mu_k - \alpha_{k+1}\delta_k) + \beta_{k+1}^2 \pi_k.$$

□

The last result of this section follows from Theorem 3 and justifies that if g is not orthogonal to the eigenspaces of H associated with nonpositive eigenvalues, Algorithm 11 eventually detects nonpositive curvature.

Theorem 5 (Dembo et Steihaug, 1983, Theorem 2.4)**.** *Let the stopping criterion in Algorithm 11 be $\|r_k\| = 0$ and let $\epsilon = 0$. If H is not positive definite, either*

1. *$r_k^T H r_k \leq 0$ for a certain iteration k, or*
2. *g is orthogonal to the eigenspaces associated with the nonpositive eigenvalues of H.*

4.3.2 Global convergence

In this subsection we show that the truncated-Newton method paired with Algorithm 11 is globally convergent. Our analysis follows that of Dembo et Steihaug (1983), and so does Algorithm 12.

Algorithm 12 : Truncated Newton Method for (4.1)

Require: x_0, $\epsilon_a > 0$, $\epsilon_r > 0$

Compute: $f_0 = f(x_0)$, $g_0 = g(x_0)$, set $k = 0$
while $\|g_k\| > \epsilon_a + \epsilon_r \|g_0\|$ **do**
  Choose $H_k = H_k^T \approx \nabla^2 f(x_k)$, $\tau_a > 0$, $\tau_r > 0$
  Compute s_k such that $\|H_k s + g_k\| \leq \tau_a + \tau_r \|g_k\|$ {use Algorithm 11}
  Compute $t_k > 0$ that satisfies the Wolfe conditions

$$f(x_k + t_k s_k) \leq f_k + \alpha t_k g_k^T s_k \qquad \alpha \in (0, \tfrac{1}{2}) \tag{4.24}$$
$$\nabla f(x_k + t_k s_k)^T s_k \geq \beta g_k^T s_k \qquad \beta \in (\alpha, 1) \tag{4.25}$$

  $x_{k+1} = x_k + t_k s_k$
  $g_{k+1} = \nabla f(x_{k+1})$
  $k \leftarrow k + 1$

Lemma 1 (Dembo et Steihaug, 1983, Lemma A.2)**.** *Let $\epsilon > 0$ be as in Algorithm 11. Assume that there exists $M > 0$ such that $\|H_k\| \leq M$ for all k in Algorithm 12. If $g_k \neq 0$, there exist constants $\gamma_0 > 0$ and $\gamma_1 > 0$ that only depend on ϵ and M such that*

$$g_k^T s_k \leq -\gamma_0 \|g_k\|^2, \quad \textit{and} \tag{4.26}$$

$$\|s_k\| \leq \gamma_1 \|g_k\|. \tag{4.27}$$

Proof. In Step 4 of Algorithm 12, Algorithm 11 is initialized with $r_0 = -g_k$. If $r_0^T H_k r_0 \leq \epsilon \rho_0$ then $s_k = -g_k$ and we may choose $\gamma_0 = 1$ in (4.26). Otherwise, the i-th iteration of Algorithm 11 sets $s_i = \sum_{j=1}^{i} \alpha_j p_{j-1}$ where $\alpha_j = r_{j-1}^T H_k r_{j-1} / \|H_k p_{j-1}\|^2$. Thus, $g_k^T s_i = -\sum_{j=1}^{i} \alpha_j r_0^T p_{j-1}$. For $j = 1, \ldots, i$, we have from Corollary 1 and (4.9) that $r_{j-1}^T H_k r_{j-1} > 0$, while (4.13) yields $r_0^T p_{j-1} > 0$. Hence, for $j = 1, \ldots, i$,

$$g_k^T s_i \leq -r_0^T p_0 \frac{r_0^T H_k r_0}{\|H_k p_0\|^2} \leq -\|g_k\|^2 \frac{g_k^T H_k g_k}{\|H_k\|^2 \|g_k\|^2} \leq -\frac{g_k^T H_k g_k}{\|H_k\|^2}.$$

But $g_k^T H_k g_k > \epsilon g_k^T g_k$, so $g_k^T s_i \leq -\epsilon \|g_k\|^2 / \|H_k\|^2 \leq -\epsilon \|g_k\|^2 / M^2$. Thus, (4.26) holds with $\gamma_0 = \min(1, \epsilon / M^2)$.

Similarly, if $s_i = -g_k$, we may choose $\gamma_1 = 1$ in (4.27). Otherwise,

$$\|s_i\| = \left\| \sum_{j=1}^{i} \frac{r_{j-1}^T H_k r_{j-1}}{\|H_k p_{j-1}\|^2} p_{j-1} \right\| \leq \sum_{j=1}^{i} \frac{r_{j-1}^T H_k r_{j-1}}{\|H_k p_{j-1}\|^2} \|p_{j-1}\| \leq \frac{r_0^T H_k r_0}{\|H_k p_0\|^2} \|p_0\|.$$

But $\epsilon \|p_0\|^2 < p_0^T H_k p_0 \leq \|p_0\| \|H_k p_0\|$, so $\|H_k p_0\| > \epsilon \|p_0\|$. Moreover $r_0^T H_k r_0 \leq \|H_k\| \|r_0\|^2$. Thus, $\|s_i\| \leq \|H_k\| / \epsilon^2 \, \|g_k\| \leq M / \epsilon^2 \, \|g_k\|$. Finally, (4.27) holds with $\gamma_1 = \max(1, M/\epsilon^2)$. □

Global convergence of Algorithm 12 follows from the next two results, whose proofs are identical to those of (Dembo et Steihaug, 1983, Theorems 2.1 and A.3).

Theorem 6 (Dembo et Steihaug, 1983, Theorem A.3)**.** *Consider Algorithm 12 in which steps are computed using Algorithm 11. The sequence of iterates $\{x_k\}$ is well defined and $\lim_{k \to \infty} \|g_k\| = 0$.*

Theorem 7 (Dembo et Steihaug, 1983, Theorem 2.1)**.** *If the sequence $\{x_k\}$ generated by Algorithm 12 has a limit point x^* where $H(x^*)$ is positive definite then the whole sequence $\{x_k\}$ converges to x^*.*

4.3.3 Local convergence

We now show that Algorithm 12 has fast local convergence properties provided the tolerances τ_a and τ_r are chosen appropriately.

Lemma 2. *Consider Algorithm 12 in which steps are computed using Algorithm 11 and assume that $g_k \neq 0$ and that there exists $M > 0$ such that $\|H_k\| \leq M$ and H_k is positive definite for all k. If $\tau_a = 0$ and $\tau_r = o(1)$, then $\|g_k + H_k s_k\| = o(\|s_k\|)$.*

Proof. The termination condition of CR is $\|g_k + H_k s_k\| \leq \tau_k$ with $\tau_k = \tau_a + \tau_r \|g_k\| = o(\|g_k\|)$ by assumption. Moreover, CR does at least one iteration. Indeed, if it were not the case and $s_k = 0$, that would mean that $\|g_k\| \leq \tau_r \|g_k\|$, and therefore that $g_k = 0$, so that Algorithm 12 would have stopped earlier. Because H_k is positive definite,

$$g_k^T H_k g_k = p_{k0}^T H_k p_{k0} > \epsilon \|p_{k0}\|^2 = \epsilon \|g_k\|^2.$$

By (4.11), $\|s_k\| \geq \|s_{k1}\| = \alpha_{k1} g_k$, the second index being the CR iteration. So,

$$\frac{\|g_k + H_k s_k\|}{\|s_k\|} \leq \frac{\tau_k}{\alpha_{k1} \|g_k\|} = \frac{\|H_k g_k\|^2}{g_k^T H_k g_k} \frac{\tau_k}{\|g_k\|} \leq \frac{\|H_k\|^2 \|g_k\|^2}{g_k^T H_k g_k} \frac{\tau_k}{\|g_k\|} \leq \frac{M^2}{\epsilon} \frac{\tau_k}{\|g_k\|}.$$

Because $\tau_k = o(\|g_k\|)$, we have $\|g_k + H_k s_k\| = o(\|s_k\|)$. □

Lemma 2 allows us to establish the following result, which parallels (Dembo et Steihaug, 1983, Theorem 2.2) and is an application of (Dennis et Moré, 1977, Theorem 6.4).

Theorem 8. *Let the sequence of iterates $\{x_k\}$ generated by Algorithm 12 converge to x^* such that $H(x^*)$ is symmetric positive definite. If the Goldstein-Armijo conditions* (4.24) *and* (4.25) *are satisfied and $\|g_k + H_k s_k\| = o(\|s_k\|)$, then for ϵ sufficiently small in Algorithm 11, there exists an iteration k_0 such that the stopping criterion on the residual norm in Algorithm 11 is satisfied and such that for all $k \geq k_0$, $t_k = 1$ is an acceptable step for the linesearch.*

The following result states the local convergence rate. The proof is identical to that of (Dembo et Steihaug, 1983, Theorem 2.3).

Theorem 9 (Dembo et Steihaug, 1983, Theorem 2.3)**.** *Let the sequence $\{x_k\}$ generated by Algorithm 12 converge to x^* where $H(x^*)$ is positive definite. Suppose that H is Lipschitz continuous at x^*. If the k-th iteration of Algorithm 12 sets $\tau_a = 0$ and $\tau_r = \min(1/k, \|g(x_k)\|^t)$ for some $0 < t \leq 1$, then $\{x_k\}$ converges at a rate $1 + t$.*

4.3.4 Numerical results

In this section, we compare the performance of Algorithm 12 where either CG or CR is used to compute steps. We use the Julia [1] programming language, version 0.6 to implement all three of Algorithm 11, the truncated CG algorithm of Dembo et Steihaug (1983), and Algorithm 12.

We use convex and nonconvex unconstrained problems from the CUTEst collection of Gould *et al.* (2015), accessed via the CUTEst.jl [2] Julia interface, and the modified CUTE problems of Lukšan *et al.* (2010) as implemented in the Julia library OptimizationProblems.jl [3]. We temporarily exclude nonlinear least-squares problems, which are evaluated in section 4.5. We further eliminate problems with fewer than 10 variables and we remove duplicates. Five additional problems are eliminated because they could not be solved within 1.5 hours: *indefm*, *nonmsqrt*, *sbrybnd*, *scosine* and *sscosine*.

Among our problems, 17 from OptimizationProblems.jl are known to be convex (Fourer *et al.*, 2010), : *arglina*, *arglinb*, *arglinc*, *arwhead*, *bdqrtic*, *clplatea*, *clplateb*, *clplatec*, *dixon3dq*, *dqdrtic*, *dqrtic*, *engval1*, *nondquar*, *power*, *quartc*, *tridia*, *vardim*, and their dimension varies from 10 to 10,000. The other 50 from OptimizationProblems.jl are nonconvex and their dimension ranges from 10 to 100. In total, we have 107 problems of dimension 10 to 100,000.

In Algorithm 11 and truncated CG, we set the maximum number of iterations to the number of variables. We impose a maximum of 10,000 iterations in Algorithm 12 and set $\tau_a = 0$, $\tau_r = \min(0.1, \sqrt{\|g_k\|})$, and $\epsilon_a = \epsilon_r = 10^{-6}$. Finally, we use a simple Armijo backtracking linesearch with $\alpha = 10^{-4}$, which only ensures (4.24). This is the way Algorithm 12 is often implemented in practice.

Our results are presented in the form of $\log_2$-scaled performance profiles of the number of objective evaluations, gradient evaluations, Hessian-vector products, and the sum of the three measures. We report results on convex and nonconvex problems separately, as convex problems do not trigger the modifications to CR pertaining to negative curvature. We also report results on the entire problem set.

Figure 4.1 contains the profiles for convex problems and shows that CG and CR perform equivalently, though CR yields slight savings in terms of all measures.

Figure 4.2 illustrates that both methods are essentially equivalent on nonconvex problems in terms of gradient evaluations. CR shows more substantial savings in terms of function eval-

1. https://julialang.org
2. https://github.com/JuliaSmoothOptimizers/CUTEst.jl
3. https://github.com/JuliaSmoothOptimizers/OptimizationProblems.jl

uations and Hessian-vector products than in the convex case, indicating that fewer iterations of Algorithm 11 than of CG are necessary to attain the stopping criterion, and the search direction induces fewer backtracking steps.

Profiles for all 107 problems together, including those from CUTEst.jl, appear in Figure 4.3. CG performs slightly better in terms of gradient evaluations while CR shows a slight advantage in terms of Hessian-vector products. CR fails on 10 problems overall while CG fails on 6 problems. On those, the maximum number of iterations of Algorithm 12 is reached, except for problem parkch, where the magnitude of the negative curvature directions identified by Algorithm 11 becomes so large that we eventually evaluate the log likelihood objective with arguments that result in NaNs.

In conclusion, the CR variant performs equivalently or slightly better than CG in a linesearch context. The difference is most visible in terms of Hessian-vector products.

Overall, we observe that CR has a slight advantage over CG in an inexact Newton context, where a method that produces monotonic residuals is desirable.

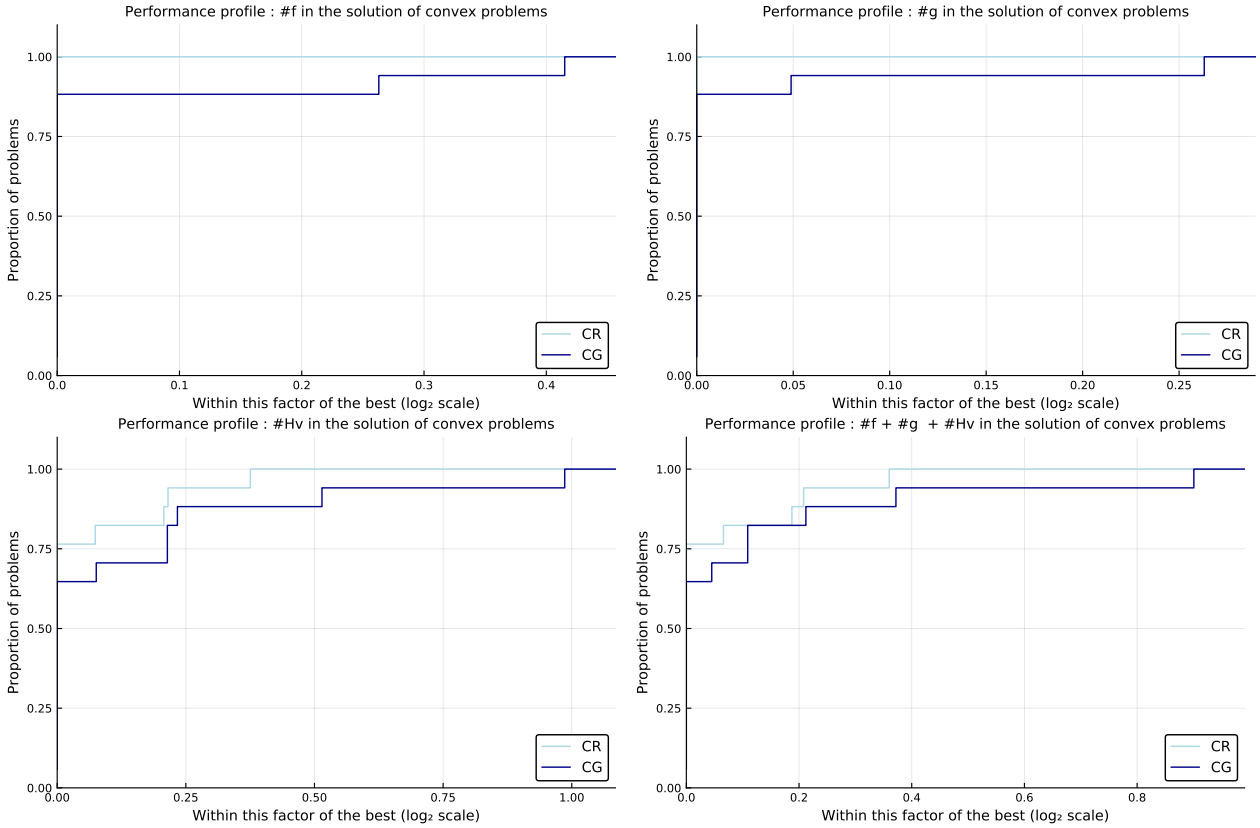

Figure 4.1 Performance of linesearch CR and CG to solve 17 convex problems in terms of evaluations of f, g and products with H.

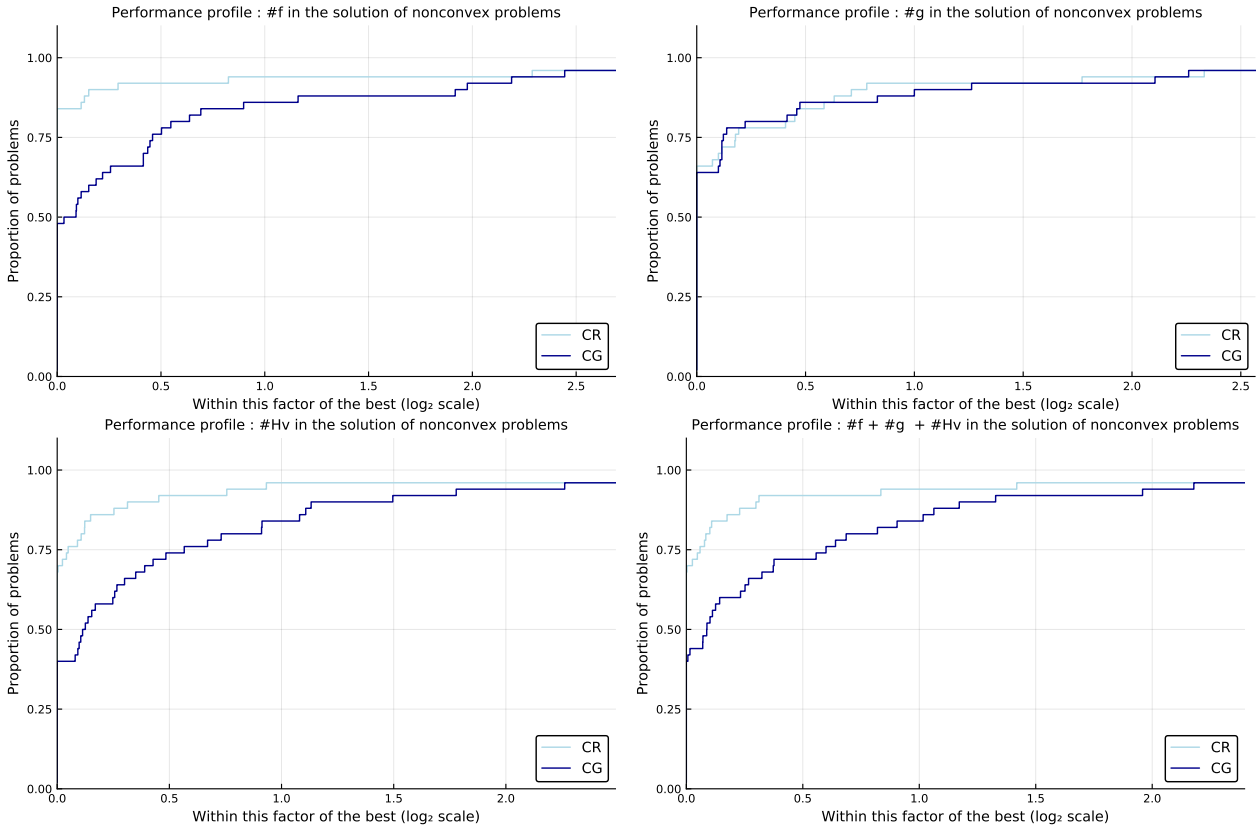

Figure 4.2 Performance of linesearch CR and CG on 50 nonconvex problems in terms of evaluations of f, g and products with H.

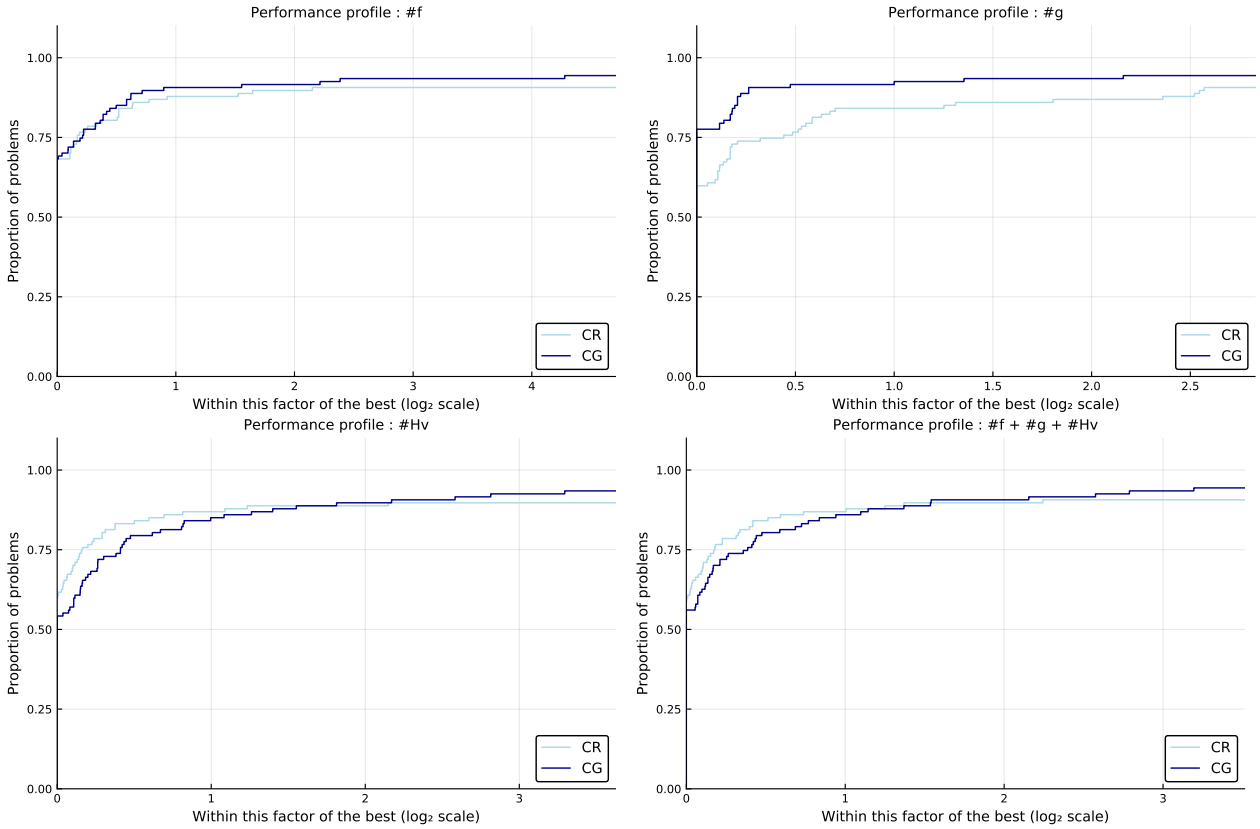

Figure 4.3 Performance of linesearch CR and CG on 107 problems in terms of evaluations of f, g and products with H.

4.4 CR in a trust-region context

Most large-scale implementations of trust-region methods compute an approximate solution s of (4.3) using the truncated CG method of Steihaug (1983). The basic mechanism is to temporarily ignore the trust-region constraint on the step and apply CG as if q were convex. The fundamental properties of CG, reviewed in §4.4.1, make it particularly well suited to this task. Steihaug (1983) describes simple modifications to CG to account for situations where the next CG iterate lies outside the trust region, or where the current CG search direction p is a direction of nonpositive curvature for q, i.e., $p^T H p \leq 0$. Yuan (2000) establishes that truncated CG yields an approximate minimizer s^C such that $q(s^C) \leq \frac{1}{2} q_k(s^*)$, where s^* is a global minimizer of (4.3). Algorithm 13 gives the trust-region process. In the remainder of this section, we introduce a version of CR to be used at line 5. We refer the reader to (Conn *et al.*, 2000) for a comprehensive account of trust-region methods.

Algorithm 13 : Trust-Region Method for (4.1)

Require: x_0, $\Delta_0 > 0$, $\epsilon_a > 0$, $\epsilon_r > 0$, $0 < \eta_1 \leq \eta_2 < 1$, $0 < \gamma_1 \leq \gamma_2 < 1$
1: **evaluate** $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$
2: **initialize** $k = 0$
3: **while** $\|g_k\| > \epsilon_a + \epsilon_r \|g_0\|$ **do**
4: **choose** $H_k = H_k^T \approx \nabla^2 f(x_k)$
5: **compute** s_k by approximately solving (4.3);
stop if $\|H_k s_k + g_k\| \leq \tau_a + \tau_r \|g_k\|$ or $\|s_k\| = \Delta_k$ {use Algorithm 14}
6: **compute** $\sigma_k = (f_k - f(x_k + s_k))/(q_k(x_k) - q_k(x_k + s_k))$
7: **if** $\sigma_k < \eta_1$ {unsuccessful iteration} **then**
8: $x_{k+1} = x_k$, $f_{k+1} = f_k$, $g_{k+1} = g_k$
9: **set** $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$
10: **else** {successful iteration}
11: $x_{k+1} = x_k + s_k$
12: **evaluate** $f_{k+1} = f(x_{k+1})$, $g_{k+1} = \nabla f(x_{k+1})$
13: **if** $\sigma_k \geq \eta_2$ {very successful iteration} **then**
14: **set** $\Delta_{k+1} \in [\Delta_k, \infty)$
15: **else**
16: **set** $\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k]$
17: $k \leftarrow k + 1$
18: **return** x_k

4.4.1 Background

Convergence of trust-region methods to a stationary point hinges around the concept of decrease obtained at the Cauchy point, which is determined by the solution of

$$\underset{\alpha}{\text{minimize}}\ q_k(-\alpha g_k) \quad \text{subject to}\ \ 0 \leq \alpha \leq \Delta/\|g_k\|,$$

i.e., the minimizer of the model along the steepest-descent direction and inside the trust region. The Cauchy point may or may not lie on the boundary. If q_k is convex along the steepest-descent direction but the unconstrained minimizer lies outside the trust region, or if q is concave along the steepest-descent direction, the Cauchy point lies on the boundary. However we compute an approximate solution s_k of (4.3), convergence will be guaranteed provided we satisfy the sufficient decrease condition, which requires that our step achieve at least a fixed fraction of Cauchy decrease at each trust-region subproblem. The sufficient-decrease condition demands that there exist a constant $\kappa \in (0,\, 1)$, independent of k, such that

$$q_k(s_k) \leq -\kappa \|g_k\| \min\left(\frac{\|g_k\|}{1 + \|H_k\|},\, \Delta_k\right). \tag{4.28}$$

We refer to (Conn *et al.*, 2000, §6.3.4) for more information.

If we neglect the trust-region constraint temporarily, the first-order optimality condition of (4.3) may be stated as the symmetric linear system $H_k s = -g_k$.

Both CG and CR can be constructed from the Lanczos (1950) process, which theoretically generates an orthonormal basis $\{v_1, v_2, v_3, \ldots\}$ of the increasing Krylov subspaces $\text{Span}\{-g, -Hg, -H^2 g, \ldots\}$.

CG has several desirable properties that make it a natural candidate for (4.3), even when H_k is indefinite. Those properties follow from the definition of the method and (Steihaug, 1983, Theorem 2.1) and are summarized in the following result.

Theorem 10. *Assume H is positive definite and s_1, s_2, ... are the iterates generated by CG on $Hs = -g$. Then,*

1. $\|s_{j+1}\| > \|s_j\|$ *for* $j = 1, 2, \ldots,$
2. $s_j \in \arg\min \{q(s) \mid s \in \text{Span}\{v_1, \ldots, v_j\}\},$
3. *the function* $\alpha \mapsto q(s_j + \alpha(s_{j+1} - s_j))$ *is monotonically decreasing over* $[0,\, 1]$ *for all* $j = 1, 2, \ldots.$

The properties of Theorem 10 are important in the solution of (4.3) for the following reasons. First, property 2 is relevant because minimizing $q(s)$ is the end goal. In case the solution of (4.3) lies interior to the trust region, standard CG will identify it in at most n iterations in exact arithmetic. If the problem is convex and the solution lies outside the trust-region, property 1 of the iterates implies that there will be an iteration j such that $\|s_j\| \leq \Delta$ but $\|s_{j+1}\| > \Delta$. By property 3, we may compute $\alpha \in [0, 1]$ such that $\|s_j + \alpha(s_{j+1} - s_j)\| = \Delta$ and possibly further improve q. If H is indefinite, CG is guaranteed to observe negative curvature along one of its search directions provided Δ is sufficiently large because those search directions, denoted p_j, are *conjugate* with respect to H, i.e, they satisfy $p_i^T H p_j = 0$ if $i \neq j$. Note that this property is not guaranteed for the Lanczos vectors v_i. Thus if CG encounters a situation where $\|s_j\| < \Delta$ and p_j is a direction of negative curvature, $(p_j^T H p_j < 0)$, then p_j may simply be followed to the boundary of the trust region because it is guaranteed to be a descent direction. Further information on the procedure just outlined, traditionally referred to as the *truncated conjugated gradient* method, or the *Steihaug-Toint* strategy, may be found in (Steihaug, 1983) and (Conn *et al.*, 2000, §7.5.1).

In contrast, CR aims to reduce the residual norm $\|g + Hs\|$ at each iteration. The following result summarizes properties of CR on a positive definite system that are relevant in a trust-region context.

Theorem 11 (Fong et Saunders, 2012, Theorems 2.3, 2.5)**.** *Assume H is positive definite and s_1, s_2, ... are the iterates generated by CR on $Hs = -g$. Then, for $j = 1, 2, \ldots$,*

1. $\|s_{j+1}\| \geq \|s_j\|$,
2. $s_j \in \arg\min \{\|g + Hs\| \mid s \in \text{Span}\{v_1, \ldots, v_j\}\}$,
3. $q(s_j + t(s_{j+1} - s_j)) \leq q(s_j)$ *for* $t \in [0, 1]$,
4. $q(s_{j+1}) < q(s_j)$.

Theorem 11 shows that CR possesses properties closely related to those of CG and suggests that CR may be viable as a trust-region subproblem solver. In the next section, we examine how standard CR may be modified to solve (4.3).

4.4.2 Truncated CR

If H is symmetric but not positive definite, q is nonconvex and Algorithm 10 no longer ensures that the search directions p_k are descent directions. Fong (2011, Theorem 2.1.5) shows that $-p_k^T \nabla q(s_k) = p_k^T r_k > 0$ as long as $p_k^T H p_k > 0$, and $\alpha_{k+1} > 0$ as long as $r_k^T H r_k > 0$.

Theorem 1 shows that in CR, the search directions p_k are conjugate with respect to H^2, while the residuals r_k are conjugate with respect to H for as long as positive curvature holds. Thus, $r_k^T H r_k$ is a reliable indicator of convexity, but it is possible to observe $p_k^T H p_k \leq 0$ before $r_k^T H r_k \leq 0$, so that we must monitor the sign of both quantities at each iteration.

In Algorithm 10, we see that if $r_k^T H r_k = 0$ then $\beta_k = \zeta_k = 0$, so $p_k = r_k$. This is a special case of $p_k^T H p_k = 0$. But $\zeta_k = 0$ implies $\alpha_{k+1} = 0$, so if nothing is changed in Algorithm 10, an error will occur because $\zeta_{k+1} = r_{k+1}^T H r_{k+1} = 0$ and β_{k+1} will be undefined.

Our strategy is such that as soon as nonpositive curvature is detected, s_k is updated one last time and the algorithm stops.

Algorithm 14 describes CR for trust-region. A separate procedure, described later in Algorithm 15, returns a boolean `terminate`, a search direction p_{k-1} and a steplength α_k. `terminate` is `false` as long as Algorithm 10 continues to apply, and `true` when either nonpositive curvature is discovered or s_k is on the boundary of the trust region. If q is convex, or if a solution of (4.2) lies inside the trust region and is found before any nonpositive curvature is discovered, Algorithm 14 is equivalent to Algorithm 10.

Algorithm 14 : CR for (4.3) (trust-region version)

Require: H, g, $\Delta > 0$, $\tau_a > 0$, $\tau_r > 0$

Initialize: $k = 0$, $s_0 = 0$, $r_0 = -g$, $u_0 = Hr_0$, $\zeta_0 = r_0^T u_0$, $p_0 = r_0$, $q_0 = u_0$, $\delta_0 = \zeta_0$, $\rho_0 = r_0^T r_0$, $\nu_0 = \sqrt{\rho_0}$, $\mu_0 = \rho_0$
while $\nu_k > \tau_a + \tau_r \|g\|$ **do**
  $k \leftarrow k + 1$
  compute `terminate`, α_k and p_{k-1} using Algorithm 15
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  if `terminate` **then return** s_k $\{q$ is nonconvex or $\|s_k\| = \Delta\}$
  $r_k = r_{k-1} - \alpha_k q_{k-1}$
  $\rho_k = \rho_{k-1} - \alpha_k \zeta_{k-1}$ $\{\rho_k = \|r_k\|^2\}$
  $\nu_k = \sqrt{\rho_k}$ $\{\nu_k = \|r_k\|\}$
  $u_k = Hr_k$
  $\zeta_k = r_k^T u_k$
  $\beta_k = \zeta_k / \zeta_{k-1}$
  $p_k = r_k + \beta_k p_{k-1}$
  $q_k = u_k + \beta_k q_{k-1}$ $\{q_k = Hp_k\}$
  $\mu_k = \rho_k + \beta_k(\mu_{k-1} - \alpha_k \delta_{k-1})$ $\{\mu_k = p_k^T r_k\}$
  $\delta_k = \zeta_k + \beta_k^2 \delta_{k-1}$
return s_k

Whether or not we discover nonpositive curvature along p_{k-1} or r_{k-1}, we must monitor the sign of α_k and select a final iterate inside the trust region. Because r_{k-1} is always a

descent direction, the direction of best decrease among p_{k-1} and r_{k-1} is followed to either the minimum of the quadratic along that direction, or the boundary of the trust region. To that end, we define $\alpha_{p+} > 0$ and $\alpha_{p-} < 0$ as the step lengths to the boundary in the direction p_{k-1}, and $\alpha_r > 0$ the step length in the direction r_{k-1}. Note that α_{p+}, α_{p-} and α_r can be computed at a moderate cost because ρ_{k-1} contains the value of $\|r_{k-1}\|^2$ and $\|p_{k-1}\|^2$ can be recurred as in Algorithm 11.

In Algorithm 15, we use the notation "$\alpha \leftarrow$ *value1* **if** *condition* **else** *value2*" to mean that if *condition* evaluates to true, α receives *value1*, and receives *value2* otherwise. Below, line numbers refer to Algorithm 15. We distinguish several cases.

Case I $p_{k-1}^T H p_{k-1} = 0$ (line 3), which covers the case $r_{k-1}^T H r_{k-1} = 0$ by (4.30): $q(s_{k-1} + \alpha_k p_{k-1}) = q(s_{k-1}) - \alpha_k r_{k-1}^T p_{k-1}$ is linear along p_{k-1}. If H is not positive definite, (4.13) may not hold. We consider two cases:

1. $p_{k-1}^T r_{k-1} = 0$ (line 5): q is constant along p_{k-1}. We reset $p_{k-1} \leftarrow r_{k-1}$, which is a descent direction. For any steplength α, $q(s_{k-1} + \alpha r_{k-1}) = q(s_{k-1}) - \alpha\|r_{k-1}\|^2 + \frac{1}{2}\alpha^2 r_{k-1}^T H r_{k-1} = q(s_{k-1}) - \alpha\rho_{k-1} + \frac{1}{2}\alpha^2\zeta_{k-1}$ is stationary when $\alpha\zeta_{k-1} = \rho_{k-1}$. Thus, we set α_k on line 8:
   (a) $\zeta_{k-1} > 0$: q decreases from $\alpha = 0$ to $\alpha_b = \rho_{k-1}/\zeta_{k-1}$. To remain inside the trust-region, we choose $\alpha_k = \min(\alpha_b, \alpha_r)$.
   (b) $\zeta_{k-1} \leq 0$: q is unbounded below along r_{k-1}. We set $\alpha_k = \alpha_r$.
2. $p_{k-1}^T r_{k-1} \neq 0$ (line 10): q is linear but not constant along p_{k-1} and α_k must be such that $\alpha_k p_{k-1}$ is a descent direction. As r_{k-1} is a descent direction, we consider the decrease in q along both directions and choose the best. Call α_{p*} the optimal steplength along p_{k-1}, which is α_{p+} if $\mu_{k-1} = p_{k-1}^T r_{k-1} > 0$, and α_{p-} otherwise because q is linear along p_{k-1}. Call α_{r*} the optimal steplength along r_{k-1}, i.e., $\min(\alpha_b, \alpha_r)$ if $\zeta_{k-1} > 0$ and α_r otherwise. According to (4.29),

$$\begin{aligned}\xi_k &:= q(s_{k-1} + \alpha_{p*} p_{k-1}) - q(s_{k-1} + \alpha_{r*} r_{k-1}) \\ &= -\alpha_{p*} p_{k-1}^T r_{k-1} + \alpha_{r*}\|r_{k-1}\|^2 + \tfrac{1}{2}(\alpha_{p*}^2 p_{k-1}^T H p_{k-1} - \alpha_{r*}^2 r_{k-1}^T H r_{k-1}) \\ &= -\alpha_{p*}\mu_{k-1} + \alpha_{r*}\rho_{k-1} - \tfrac{1}{2}\alpha_{r*}^2\zeta_{k-1},\end{aligned}$$

   which is computed at line 14. If $\xi_k > 0$, the best decrease occurs along r_{k-1}, which is then chosen as search direction with $\alpha_k = \alpha_{r*}$. Otherwise, we select p_{k-1} with $\alpha_k = \alpha_{p*}$.

Algorithm 15 : Step computation for Algorithm 14

Require: $\Delta > 0$, ζ_{k-1}, δ_{k-1}, ρ_{k-1}, μ_{k-1}, ν_{k-1}, s_{k-1}, r_{k-1}, p_{k-1}, q_{k-1}, $\epsilon > 0$

`terminate = false`
Compute $\alpha_{p+} > 0$, $\alpha_{p-} < 0$ such that $\|s_{k-1} + \alpha p_{k-1}\| = \Delta$, $\alpha \in \{\alpha_{p+}, \alpha_{p-}\}$
if $|\delta_{k-1}| \leq \epsilon \|p_{k-1}\| \|q_{k-1}\|$ $\{p_{k-1}^T H p_{k-1} \approx 0\}$ **then**
  `terminate = true` $\{q$ is nonconvex$\}$
  if $|\mu_{k-1}| \leq \epsilon \|p_{k-1}\| \nu_{k-1}$ $\{p_{k-1}^T r_{k-1} \approx 0\}$ **then**
    $p_{k-1} \leftarrow r_{k-1}$
    Compute $\alpha_r > 0$ such that $\|s_{k-1} + \alpha_r r_{k-1}\| = \Delta$
    $\alpha_k \leftarrow \min(\alpha_r, \rho_{k-1}/\zeta_{k-1})$ **if** $\zeta_{k-1} > 0$ **else** α_r
  else
    Compute $\alpha_r > 0$ such that $\|s_{k-1} + \alpha_r r_{k-1}\| = \Delta$
    if $\zeta_{k-1} > 0$ **then**
      $\alpha_r \leftarrow \min(\alpha_r, \rho_{k-1}/\zeta_{k-1})$
    $\alpha_k \leftarrow \alpha_{p+}$ **if** $\mu_{k-1} > 0$ **else** α_{p-}
    $\xi_k = -\alpha_k \mu_{k-1} + \alpha_r \rho_{k-1} - \frac{1}{2}\alpha_r^2 \zeta_{k-1}$
    if $\xi_k > 0$ $\{q(s_{k-1} + \alpha_r r_{k-1}) < q(s_{k-1} + \alpha_k p_{k-1})\}$ **then**
      $p_{k-1} \leftarrow r_{k-1}$, $\alpha_k \leftarrow \alpha_r$
else if $\delta_{k-1} > 0$ and $\zeta_{k-1} > 0$ **then**
  $\alpha_k = \zeta_{k-1}/\|q_{k-1}\|^2$
  if $\alpha_k \geq \alpha_{p+}$ **then**
    `terminate = true` $\{s_k$ is on the boundary of the trust region$\}$
    $\alpha_k \leftarrow \alpha_{p+}$
else if $\delta_{k-1} > 0$ and $\zeta_{k-1} < 0$ **then**
  `terminate = true`
  $\alpha_k \leftarrow \min(\alpha_{p+}, \mu_{k-1}/\delta_{k-1})$ **if** $\mu_{k-1} > 0$ **else** $\max(\alpha_{p-}, \mu_{k-1}/\delta_{k-1})$
  Compute $\alpha_r > 0$ such that $\|s_{k-1} + \alpha_r r_{k-1}\| = \Delta$
  $\xi_k = -\alpha_k \mu_{k-1} + \alpha_r \rho_{k-1} + \frac{1}{2}(\alpha_k^2 \delta_{k-1} - \alpha_r^2 \zeta_{k-1})$
  if $\xi_k > 0$ **then** $p_{k-1} \leftarrow r_{k-1}$, $\alpha_k \leftarrow \alpha_r$
else if $\delta_{k-1} < 0$ and $\zeta_{k-1} > 0$ **then**
  `terminate = true`
  $\alpha_k \leftarrow \alpha_{p+}$ **if** $\mu_{k-1} > 0$ **else** α_{p-}
  Compute $\alpha_r > 0$ such that $\|s_{k-1} + \alpha_r r_{k-1}\| = \Delta$
  $\alpha_r \leftarrow \min(\alpha_r, \rho_{k-1}/\zeta_{k-1})$
  $\xi_k = -\alpha_k \mu_{k-1} + \alpha_r \rho_{k-1} + \frac{1}{2}(\alpha_k^2 \delta_{k-1} - \alpha_r^2 \zeta_{k-1})$
  If $\xi_k > 0$ **then** $p_{k-1} \leftarrow r_{k-1}$, $\alpha_k \leftarrow \alpha_r$
else if $\delta_{k-1} < 0$ and $\zeta_{k-1} < 0$ **then**
  `terminate = true`
  Compute $\alpha_r > 0$ such that $\|s_{k-1} + \alpha_r p_{k-1}\| = \Delta$
  $\alpha_k \leftarrow \alpha_{p+}$ **if** $\mu_{k-1} > 0$ **else** α_{p-}
  $\xi_k = -\alpha_k \mu_{k-1} + \alpha_r \rho_{k-1} + \frac{1}{2}(\alpha_k^2 \delta_{k-1} - \alpha_r^2 \zeta_{k-1})$
  If $\xi_k > 0$ **then** $p_{k-1} \leftarrow r_{k-1}$, $\alpha_k \leftarrow \alpha_r$
return `terminate`, α_k, p_{k-1}

Case II $p_k^T H p_k > 0$ and $r_k^T H r_k > 0$ (line 17): q is convex along p_k and r_k, so Corollary 2 applies and standard CR is applied within the trust-region.

Case III in all other situations, $p_{k-1}^T H p_{k-1} > 0$ and $r_{k-1}^T H r_{k-1} < 0$ (line 22), $p_{k-1}^T H p_{k-1} < 0$ and $r_{k-1}^T H r_{k-1} > 0$ (line 28), or $p_{k-1}^T H p_{k-1} < 0$ and $r_{k-1}^T H r_{k-1} < 0$ (line 35): negative curvature is detected. We compute ξ_k to select the direction of best decrease between r_{k-1} and p_{k-1}, and follow it to the minimum of q or the trust-region boundary.

4.4.3 Main properties

Theorem 12. *In Algorithm 14, the properties of Theorem 1 continue to hold for symmetric positive definite H.*

Proof. As long as $\delta_{i-1} > 0$ and $\mu_{i-1} > 0$, $i = 0, 1, \ldots, k$, Algorithm 14 coincides with Algorithm 10. For H symmetric positive definite, this is ensured for all iterations. □

Numerically, we relax the condition $\delta_{i-1} = 0$ to $|\delta_{i-1}| \leq \epsilon \|p_{i-1}\| \|q_{i-1}\|$, and $\mu_{i-1} = 0$ to $|\mu_{i-1}| \leq \epsilon \|p_{i-1}\| \nu_{i-1}$ for a user-defined tolerance $\epsilon > 0$.

Corollary 2. *If H is symmetric and $\delta_{i-1} > 0$ and $\mu_{i-1} > 0$ for $i = 0, 1, \ldots, k$, then the properties of Theorem 1 continue to hold at those iterations in Algorithm 14.*

Theorem 13. *Let H be symmetric but not necessarily positive definite. Algorithm 14 continues to satisfy* (4.14)–(4.17) *and* (4.21)–(4.22) *for as long as* `terminate` *is* `false`*. In addition, for $k \geq 0$,*

$$\xi_k = q(s_{k-1} + \alpha_k p_{k-1}) - q(s_{k-1} + \alpha_r r_{k-1}), \tag{4.29}$$

$$\zeta_k = 0 \implies \delta_k = 0. \tag{4.30}$$

Proof. The proof of (4.14)–(4.17) is as in Theorem 2. That of (4.21)–(4.22) is as in Theorem 4. Because q is quadratic, $\nabla q(s_k) = -r_k$, and $\nabla^2 q(s_k) = H$, we have the expansion

$$\begin{aligned} q(s_{k-1} + \alpha_k p_{k-1}) &= q(s_{k-1}) - \alpha_k p_{k-1}^T r_{k-1} + \tfrac{1}{2}\alpha_k^2 p_{k-1}^T H p_{k-1} \\ &= q(s_{k-1}) - \alpha_k \mu_{k-1} + \tfrac{1}{2}\alpha_k^2 \delta_{k-1}. \end{aligned}$$

Likewise,

$$\begin{aligned} q(s_{k-1} + \alpha_r r_{k-1}) &= q(s_{k-1}) - \alpha_r \|r_{k-1}\|^2 + \tfrac{1}{2}\alpha_r^2 r_{k-1}^T H r_{k-1} \\ &= q(s_{k-1}) - \alpha_r \rho_{k-1} + \tfrac{1}{2}\alpha_k^2 \zeta_{k-1}, \end{aligned}$$

and (4.29) follows.

If $\zeta_k = 0$, lines 12 and 13 of Algorithm 14 yield $\beta_k = 0$ and $p_k = r_k$, and $\delta_k = 0$. □

Note that (4.30) also holds for all previous variants of CR.

4.4.4 Convergence analysis

In this section, we establish that a step computed by Algorithm 14 satisfies the sufficient-decrease condition (4.28), and therefore that convergence of Algorithm 13 to a stationary point is guaranteed under standard assumptions on (4.1).

Both CG and CR begin by performing a search along the steepest-descent direction $-g$. The CG steplength $\alpha_C > 0$ is determined by minimizing

$$q(-\alpha g) = -\alpha \|g\|^2 + \tfrac{1}{2}\alpha^2 g^T H g.$$

The CR steplength $\alpha_M > 0$ is determined by minimizing

$$R(-\alpha g) = \tfrac{1}{2}\|g - \alpha H g\|^2.$$

Both of the above minimizations must take the trust-region bound into account. By definition, the first CG iterate is precisely the Cauchy point. If $g^T H g \leq 0$, both methods step to the boundary and therefore achieve the same decrease. If $g^T H g > 0$, the unconstrained minimizers are

$$\alpha_C = \frac{\|g\|^2}{g^T H g}, \qquad \text{and} \qquad \alpha_M = \frac{g^T H g}{g^T H^2 g}.$$

In addition,

$$q_C := q(-\alpha_C g) = -\tfrac{1}{2}\frac{\|g\|^4}{g^T H g}, \tag{4.31}$$

$$q_M := q(-\alpha_M g) = \frac{g^T H g}{g^T H^2 g}\left(\tfrac{1}{2}\frac{(g^T H g)^2}{g^T H^2 g} - \|g\|^2\right). \tag{4.32}$$

Theorem 11 implies that $q_M \leq 0$, which yields $\frac{1}{2}\alpha_M \leq \alpha_C$. Lemma 3 is more precise.

Lemma 3. *Let H be symmetric and g such that $g^T H g > 0$. Then $\alpha_M \leq \alpha_C$.*

Proof. Let $y = Hg$. The result follows from the Cauchy-Schwartz inequality $(g^T y)^2 \leq \|g\|^2 \|y\|^2$. □

Assume first that the CR minimizer along $-g$ lies inside the trust region. Lemma 3 implies that

$$\frac{(g^T H g)^2}{g^T H^2 g} \leq \|g\|^2,$$

so that (4.32) yields $q_M \leq -\frac{1}{2}\alpha_M \|g\|^2$. But

$$\alpha_M \geq \frac{g^T H g}{(1 + \|H\|)\, g^T H g} = \frac{1}{1 + \|H\|},$$

and therefore

$$q_M \leq -\tfrac{1}{2} \frac{1}{1 + \|H\|} \|g\|^2.$$

Suppose now that the CG minimizer lies on the boundary or outside the trust region. In this case, we reset $\alpha_C = \Delta / \|g\|$ and obtain

$$q_C := q(-\alpha_C g) = \Delta \left(\tfrac{1}{2} \Delta \frac{g^T H g}{\|g\|^2} - \|g\| \right).$$

If the CR minimizer also lies on the boundary or outside the trust region, $q_M = q_C$. If on the other hand $\alpha_M < \Delta / \|g\|$, (4.32) yields

$$q_M < \frac{\Delta}{\|g\|} \left(\tfrac{1}{2} \frac{\Delta}{\|g\|}\, g^T H g - \|g\|^2 \right) = q_C.$$

By Theorem 11, in the event that $g^T H g > 0$ and $\alpha_M < \Delta / \|g\|$, subsequent CR iterations further reduce the value of $q(s)$. Should CR encounter a direction of negative curvature, Algorithm 14 guarantees that no increase in the value of $q(s)$ can result, but that further decrease may occur. Thus in all cases, CR improves upon its first iterate and therefore yields an approximate solution of (4.3) that satisfies the sufficient-decrease condition (4.28).

4.4.5 Numerical results

We use the same benchmark problems as in subsection 4.3.4. We eliminate problems *jimack*, *sbrybnd* and *scosine* because they did not solve within 1.5 hours. In total, we have 109 problems.

Algorithms 13 to 15 are implemented in Julia v0.6. Truncated CG is implemented as described by Steihaug (1983).

Algorithm 13 uses $\epsilon_a = \epsilon_r = 10^{-6}$, $\Delta_0 = 10$, $\eta_1 = 10^{-4}$, $\eta_2 = 0.99$, $\gamma_1 = \gamma_2 = \frac{1}{3}$. Line 14 is changed to $\Delta_{k+1} = 3\Delta$, and line 16 becomes $\Delta_{k+1} = \Delta_k$. We impose a maximum of $10,000$

iterations.

Truncated CG and CR have a maximum number of iterations equal to the number of variables in the problem. Algorithms 14 and 15 set ϵ to the machine precision for the detection of negative curvature. Finally, at iteration k of the trust-region algorithm, the tolerance for the inner iterations is $\tau_a + \tau_r \|g_k\|$ with $\tau_a = 0$ and $\tau_r = \min(0.1, \sqrt{\|g_k\|})$.

Both variants perform equivalently on convex problems in terms of function and gradient evaluations, as illustrated by the profiles of Figure 4.4. In particular, both variants solve all problems. However, CG turns out to be slightly better in terms of Hessian-vector products.

On nonconvex problems, Figure 4.5 shows that CR performs better for all types of evaluations, which is a surprise given the CG optimality property of minimizing the quadratic objective as long as it is convex. However, the gap between the performance curves remains small. Both variants solve all problems.

Figure 4.6 shows that the trend persists on the set of 109 problems, where CR has a more substantial advantage in terms of Hessian-vector products. The CG variant fails on 6 problems, while CR fails on 4 problems.

Overall, although CG is conceptually the best suited iterative method for (4.3) because of its optimality property, CR performs fewer Hessian-vector products per trust-region subproblem on average, which in turn results in savings in terms of objective and gradient evaluations.

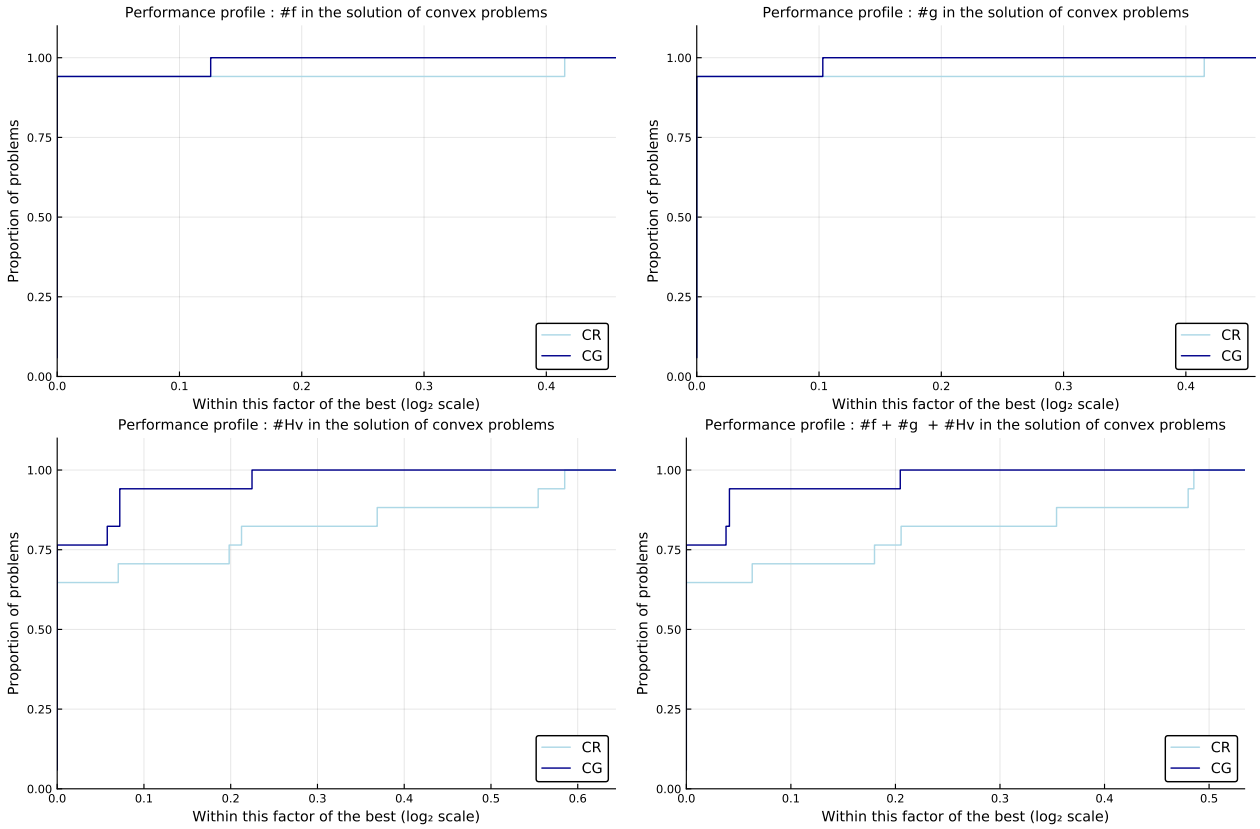

Figure 4.4 Performance of trust-region CR and CG on 17 convex problems in terms of evaluations of f, g and products with H.

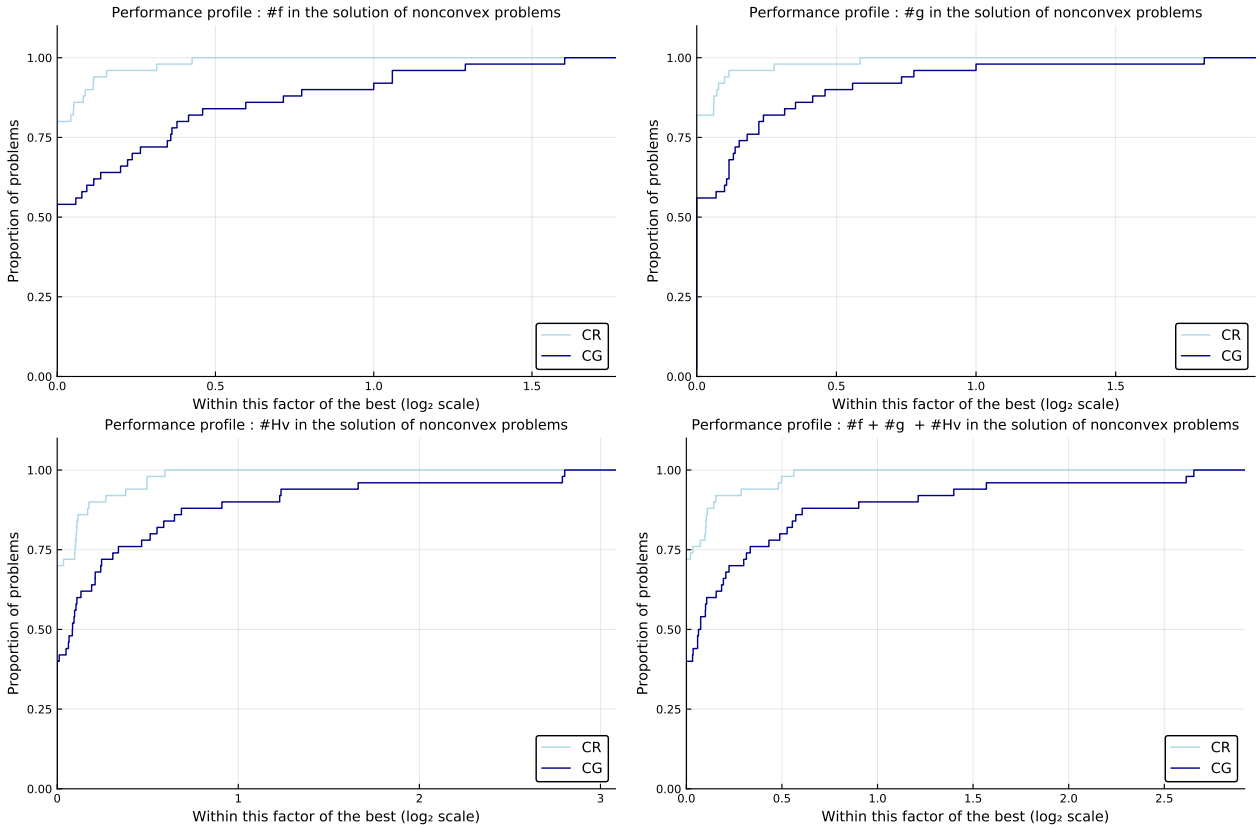

Figure 4.5 Performance of trust-region CR and CG on 50 nonconvex problems in terms of evaluations of f, g and products with H.

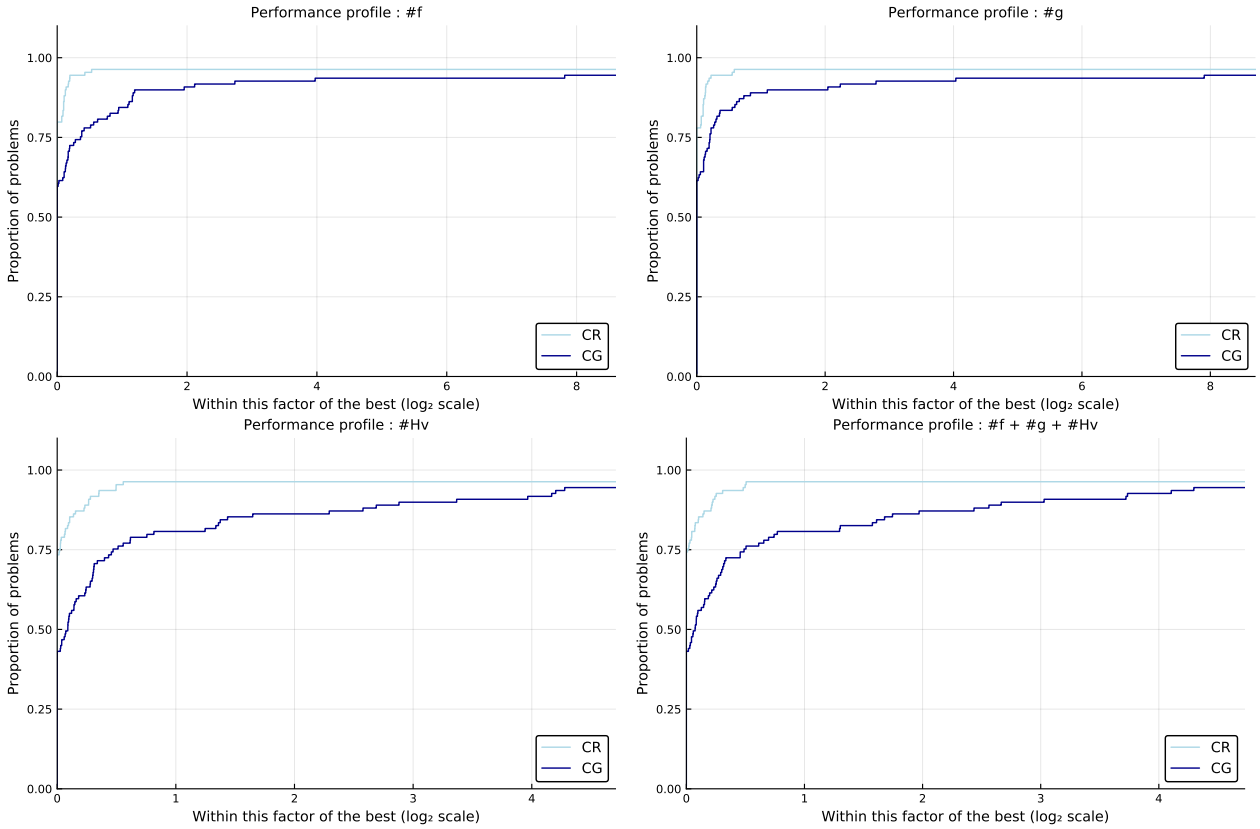

Figure 4.6 Performance of trust-region CR and CG on 109 problems in terms of evaluations of f, g and products with H.

Algorithm 16 : CRLS for (4.33)

Require: J, F, $\Delta > 0$, $\tau_a > 0$, $\tau_r > 0$, $\epsilon > 0$

Initialize: $k = 0$, $s_0 = 0$, $r_0 = -F$, $\tilde{r}_0 = J^T r_0$, $w_0 = J\tilde{r}_0$, $\zeta_0 = w^T w$, $p_0 = \tilde{r}_0$, $q_0 = w_0$
while $\|\tilde{r}_{k-1}\| > \tau_a + \tau_r \|F\|$ **do**
  $k \leftarrow k + 1$
  $v_k = J^T q_{k-1}$
  $\alpha_k = \zeta_{k-1} / \|v\|^2$
  Compute $\alpha_p > 0$ such that $\|s_{k-1} + \alpha_p p_{k-1}\| = \Delta$
  if $\|q_{k-1}\|^2 \leq \epsilon \|p_{k-1}\| \|v_k\|$ {(near) zero curvature detected} **then**
    $p_{k-1} \leftarrow \tilde{r}_{k-1}$
    $\alpha_k \leftarrow \min(\alpha_p, \|\tilde{r}_{k-1}\|^2 / \zeta_{k-1})$
    $s_k = s_{k-1} + \alpha_k p_{k-1}$
    return s_k
  else
    if $\alpha_k \geq \alpha_p$ **then**
      $\alpha_k \leftarrow \alpha_p$
      $s_k = s_{k-1} + \alpha_k p_{k-1}$
      return s_k
  $s_k = s_{k-1} + \alpha_k p_{k-1}$
  $r_k = r_{k-1} - \alpha_k q_{k-1}$
  $\tilde{r}_k = J^T r_k$
  $w_k = J\tilde{r}_k$
  $\zeta_k = w_k^T w_k$
  $\beta_k = \zeta_k / \zeta_{k-1}$
  $p_k = \tilde{r}_k + \beta_k p_{k-1}$
  $q_k = w_k + \beta_k q_{k-1}$
return s_k

4.5 Extension to nonlinear least squares

Suppose the objective of (4.1) is $f(x) = \frac{1}{2}\|F(x)\|^2$, where $F(x) = (f_1(x), \ldots, f_m(x))$. The classical approach of Levenberg (1944) and Marquardt (1963) may be implemented by replacing $q(s)$ in (4.3) with the Gauss-Newton model $q^{\text{GN}}(s) := \frac{1}{2}\|J(x)s + F(x)\|^2$, where $J(x)$ is the Jacobian of F at x, and solving (4.1) using Algorithm 13. Thus, at each iteration the

subproblem to solve is

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \; q^{\text{GN}}(s) \quad \text{subject to} \quad \|s\| \leq \Delta. \tag{4.33}$$

Both CG and CR can be used to solve (4.33). Products with $J(x)^T J(x)$ may be dissociated, and this gives rise to variants named CGLS [4] (Hestenes et Stiefel, 1952, Section 10) and CRLS (Fong, 2011) specific to linear least squares [5].

4.5.1 CRLS for trust region

For simplicity we use J and F to refer respectively to $J(x)$ and $F(x)$ in the description of the algorithm.

Algorithm 16 is a modification of CRLS adapted to a trust-region context, with radius Δ as input parameter. Note that $\nabla f(x) = J^T F = -\tilde{r}_0$.

At any iteration k, let $\alpha_p > 0$ be the step length to the trust-region boundary in the direction p_{k-1}. Because $J^T J$ is positive semi-definite, the curvature of q^{GN} can only be zero or positive. If $p_{k-1}^T J^T J p_{k-1} = 0$ then $J p_{k-1} = q_{k-1} = 0$, and q^{GN} is constant in the direction p_{k-1}. In that case, we select $-\nabla q^{\text{GN}}(s_{k-1}) = \tilde{r}_{k-1}$ as the new search direction. We compute the steplength to the minimizer of q^{GN} in the direction $\tilde{r}_{k-1}$, and either step to the minimizer or stop at the trust-region boundary if the minimizer lies outside.

4.5.2 Numerical results

In exact arithmetic, CGLS and CRLS are equivalent to LSQR (Paige et Saunders, 1982) and LSMR (Fong et Saunders, 2011), respectively, which are based on the Golub et Kahan (1965) process and numerically preferable. LSQR and LSMR require the same number of operator-vector products per iteration as CGLS and CRLS. Björck *et al.* (1998) analyze several versions of CGLS, notably using recurred residuals, and compare their numerical stability. They conclude that the standard version of CGLS, in which the optimality residual is not recurred, is more stable than LSQR and achieves similar accuracy. Björck et Saunders (2017) perform similar comparisons between CRLS and LSMR and conclude that in its default version, CRLS is inferior to LSMR and is unable to achieve comparable accuracy. They devise a version of CRLS that requires an extra operator-vector product per iteration that is competitive with LSMR. Kloek (2012) performs similar experiments and makes similar observations. For the

4. The name CGLS appears to have been coined by Paige et Saunders (1982).

5. The earliest reference mentioning CRLS, though not under that name, that we are aware of is Björck (1979).

above reasons, in our experiments, we use LSQR and LSMR as implemented in the package Krylov.jl [6], with appropriate changes to accomodate a trust-region constraint.

We use nonlinear least-squares problems implemented in Julia from the NLSProblems.jl [7] collection, together with those from CUTEst.jl. We eliminate problems with fewer than 10 variables. We exclude *ba-l16, ba-l21, ba-l49* and *ba-l52* as they require more than 1.5 hour to be solved. We further eliminate *mgh11* because its objective is not $\mathcal{C}^2$. In total, we run 47 problems of size 10 to 123, 200.

The trust-region parameters are as in subsection 4.4.5. The maximum number of LSQR and LSMR iterations is $n + m$, $\tau_a = 0$ and $\tau_r = \min(0.1, \sqrt{\|g_k\|})$.

Figure 4.7 gives the profiles using $\#F$, $\#Ju + \#J^T v$, and $\#F + \#Ju + \#J^T v$. The profiles show that LSQR and LSMR perform equivalently, with a slight advantage for LSMR in terms of residual evaluations. Both methods fail on a single problem: *ba-l73*.

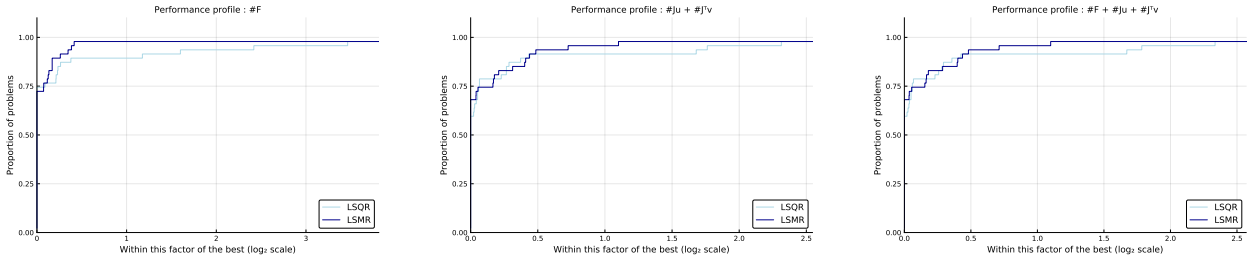

Figure 4.7 Performance of trust-region LSQR and LSMR on 101 nonlinear least-squares problems in terms of $\#F$, $\#Ju + \#J^T v$, and the sum of both.

4.6 Discussion

Most implementations of linesearch inexact Newton and trust-region methods with iterative step computation employ the conjugate gradient method. While CG is conceptually the correct method in a trust-region context due to it minimization property of the quadratic model, the CG residual is typically erratic and it is difficult to justify why it should be the method of choice in a linesearch inexact Newton context. When applied to a convex quadratic model, CR shares similarities with CG. Although CR does not minimize the quadratic model at each iteration, the value of the latter decreases monotonically. By construction, CR also produces a monotonic residual, which is in fact minimized at each iteration. Our adaptations of CR to handle nonpositive curvature in both linesearch and trust-region contexts show that

6. https://github.com/JuliaSmoothOptimizers/Krylov.jl
7. https://github.com/JuliaSmoothOptimizers/NLSProblems.jl

CR performs as well as or slightly better than CG, particularly in terms of Hessian-vector products. Our extension to nonlinear least-squares problems via the LSMR implementation of CRLS shows that it behaves comparably to the LSQR implementation of CGLS in a trust-region context.

Our implementations try to save computations and recur a number of quantities such as $p_k^T H p_k$ and $r_k^T H r_k$. It is conceivable that such recurrence formulae are subject to accumulation of rounding errors, especially on ill-conditioned problems, though we have not observed damaging results in our experiments. A finite-precision arithmetic analysis such as that of Björck *et al.* (1998) is warranted to shed light on the matter.

MINRES (Paige et Saunders, 1975) should be the preferred implementation of CR, and it generalizes CR to indefinite systems. We have not used MINRES in the present research because its implementation is substantially more involved and certain quantities of interest, such as $p_k^T H p_k$, are not readily available. However, corresponding variants of MINRES adapted to linesearch and trust-region contexts would be highly relevant.

We have deliberately left questions of preconditioning aside as they are typically application dependent. A study of the behavior of CR with generic—e.g., diagonal or incomplete Cholesky—preconditioners is left for future work.

The satisfactory performance of CR illustrated in this research raises the question of whether it would also be a worthwhile subproblem solver in constrained optimization, e.g., in projected direction methods for bound-constrained problems such as that of Lin et Moré (1998).

Finally, in trust-region methods, Yuan (2000, Theorem 2) establishes that the decrease in the quadratic model achieved by truncated CG is at least half of that obtained at a global solution of the trust-region subproblem. This is an important result and it is relevant to determine whether a similar result holds for CR.

Acknowledgements

The authors express their gratitude to Michael Saunders for comments on an earlier draft of the present research, and to Åke Björck for providing a copy of (Björck, 1979).

4.7 Appendix

4.7.1 Detailed results for the linesearch method

Detailed results for each problem are given in Table 4.1 and Table 4.2.

Table 4.1: Solution of 107 nonlinear problems with linesearch CR

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
arglina	200	2.000e+02	1.000e+03	5.8e−13	5.7e+01	2	2	2	1
arglinb	200	9.963e+01	8.651e+15	1.2e−01	1.4e+15	2	2	2	1
arglinc	200	1.011e+02	8.353e+15	7.8e−02	1.4e+15	2	2	2	1
arwhead	5000	1.110e−12	1.500e+04	5.9e−06	4.0e+04	6	6	12	5
bdqrtic	5000	2.001e+04	1.129e+06	9.2e−01	1.5e+06	10	10	40	9
box	10000	−1.865e+03	0.000e+00	2.7e−06	5.0e+01	12	5	14	4
boxpower	20000	4.716e−02	1.764e+05	2.1e−02	1.7e+05	8	7	18	6
brownal	200	2.037e−08	2.010e+06	2.8e−02	5.7e+05	2	2	2	1
broydn7d	5000	1.854e+03	1.760e+04	7.6e−04	1.1e+03	1977	1782	7610	1781
brybnd	5000	2.050e−06	1.249e+05	6.5e−03	7.8e+03	9	9	92	8
chainwoo	4000	2.388e+03	1.445e+07	1.7e−01	4.2e+05	5291	3454	85956	3453
chnrosnb	50	2.206e−06	7.636e+03	3.5e−03	3.6e+03	70	48	984	47
chnrosnb_mod	100	1.115e−05	1.764e+04	6.1e−03	6.4e+03	200	112	1195	111
chnrsnbm	50	2.275e−09	8.633e+03	2.7e−04	4.4e+03	126	70	1311	69
clplatea	5041	−1.259e−02	0.000e+00	1.1e−06	1.0e−01	8	7	646	6
clplateb	5041	−5.095e−03	0.000e+00	3.6e−07	1.2e−02	4	4	411	3
clplatec	5041	−5.021e−03	0.000e+00	7.3e−07	9.9e−02	5	5	10961	4
cosine	10000	−9.999e+03	8.775e+03	8.7e−07	7.2e+01	8	7	19	6
cragglvy	5000	1.688e+03	2.749e+06	1.1e−01	2.8e+05	13	13	92	12
curly10	10000	−1.003e+06	−6.306e−01	1.3e−04	1.3e+02	21	19	86127	18
curly20	10000	−1.003e+06	−1.344e+00	2.4e−05	3.0e+02	25	21	99683	20
curly30	10000	−1.003e+06	−2.190e+00	7.6e−05	5.1e+02	28	22	106049	21
deconvu	63	1.825e−07	1.104e+02	2.5e−05	1.1e+02	14	13	191	12
dixmaana	3000	1.000e+00	2.850e+04	2.6e−04	1.2e+03	10	8	20	7
dixmaanb	3000	1.000e+00	4.724e+04	2.1e−04	2.0e+03	8	8	14	7
dixmaanc	3000	1.000e+00	8.248e+04	2.8e−04	3.7e+03	9	9	16	8
dixmaand	3000	1.000e+00	1.586e+05	2.2e−03	7.6e+03	10	10	18	9
dixmaane	3000	1.000e+00	2.850e+04	2.6e−04	1.2e+03	10	8	20	7
dixmaanf	3000	1.000e+00	4.104e+04	2.7e−04	1.9e+03	12	12	252	11
dixmaang	3000	1.000e+00	7.607e+04	1.3e−03	3.6e+03	12	12	152	11
dixmaanh	3000	1.001e+00	1.517e+05	3.6e−03	7.4e+03	12	12	64	11
dixmaani	3000	1.000e+00	2.002e+04	3.3e−05	1.0e+03	11	11	1260	10
dixmaanj	3000	1.000e+00	3.900e+04	4.7e−04	1.8e+03	13	13	166	12
dixmaank	3000	1.000e+00	7.400e+04	9.7e−04	3.6e+03	13	13	122	12
dixmaanl	3000	1.000e+00	1.496e+05	2.4e−03	7.4e+03	13	13	86	12

Table 4.1 — Solution of 107 nonlinear problems with linesearch CR (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
dixmaanm	3000	1.001e+00	9.358e+03	3.1e−04	4.4e+02	9	9	524	8
dixmaann	3000	1.000e+00	2.018e+04	4.9e−04	1.0e+03	13	13	290	12
dixmaano	3000	1.001e+00	3.635e+04	1.1e−03	2.0e+03	13	13	198	12
dixmaanp	3000	1.002e+00	7.128e+04	2.5e−03	4.0e+03	13	13	136	12
dixon3dq	10000	9.661e−04	8.000e+00	1.8e−05	5.7e+00	10001	10001	42496	10000
dqdrtic	5000	3.143e−05	9.041e+06	1.1e−02	8.5e+04	5	5	16	4
dqrtic	5000	4.137e+09	6.241e+17	8.1e+06	1.3e+13	13	13	52	12
edensch	2000	1.200e+04	7.358e+06	2.8e−02	1.0e+05	14	14	48	13
eg2	5000	5.549e+03	2.949e+05	1.7e−03	8.8e+03	9	9	34	8
engval1	5000	5.549e+03	2.949e+05	1.7e−03	8.8e+03	9	9	34	8
errinros	50	4.022e+01	1.102e+05	4.1e−02	1.2e+05	32	27	341	26
errinros_mod	100	7.862e+01	3.140e+05	1.9e−01	1.9e+05	22	18	93	17
errinrsm	50	3.873e+01	1.529e+05	1.1e−01	1.3e+05	25	20	209	19
extrosnb	1000	8.403e−03	3.996e+05	3.8e−02	3.8e+04	48	30	216	29
fletbv3m	5000	−2.326e+05	1.982e+02	3.2e−05	4.4e+01	293	292	528	291
fletcbv2	5000	−5.003e−01	−5.003e−01	3.7e−06	4.4e−06	10001	10001	32266	10000
fletcbv3	5000	−2.286e+08	1.982e+02	5.0e+01	4.4e+01	10001	10001	13158	10000
fletcbv3_mod	100	−8.684e−02	−1.879e−02	2.8e−03	1.6e−03	10001	10001	10001	10000
fletchbv	5000	−1.916e+19	−2.302e+11	4.5e+09	2.8e+09	15817	10001	29997	10000
fletchcr	1000	7.199e−11	9.990e+02	2.2e−05	6.3e+01	2726	1634	37090	1633
fminsrf2	5625	1.000e+00	2.846e+01	2.5e−06	3.3e−01	10133	10001	27849	10000
fminsurf	5625	1.000e+00	2.859e+01	2.8e−07	3.3e−01	227	38	16029	37
freuroth	5000	6.082e+05	5.049e+06	4.4e−03	5.5e+04	28	10	43	9
genhumps	5000	1.735e−07	1.281e+08	3.0e−04	6.0e+03	7947	7530	28065	7529
genrose	500	1.000e+00	1.870e+03	2.2e−06	3.0e+02	915	457	7380	456
genrose_nash	100	1.000e+00	4.041e+02	8.7e−05	1.3e+02	196	110	790	109
hilbertb	10	2.216e−14	5.102e+02	6.8e−07	1.1e+02	6	6	12	5
indef	5000	−2.499e+07	4.603e+03	7.1e+01	8.0e+01	10001	10001	20062	10000
liarwhd	5000	1.671e−04	2.925e+06	2.6e−02	4.8e+05	13	13	36	12
mancino	100	1.059e−01	1.103e+12	9.1e+02	2.9e+09	6	6	12	5
modbeale	20000	1.695e−02	1.264e+07	2.4e−01	3.1e+05	14	13	150	12
morebv	5000	3.637e−11	1.597e−07	9.5e−08	8.0e−04	3	3	898	2
ncb20	5010	−1.463e+03	1.000e+04	2.6e−04	2.8e+02	238	218	1561	217
ncb20b	5000	7.351e+03	1.000e+04	7.2e−05	2.8e+02	14	10	203	9
noncvxu2	5000	1.182e+04	3.235e+11	2.6e+00	3.3e+06	3291	3287	13292	3286
noncvxun	5000	1.217e+04	3.335e+11	2.7e+00	3.6e+06	2343	2337	9987	2336
nondia	5000	2.570e−04	2.000e+06	1.3e−02	2.0e+06	3	3	4	2
nondquar	5000	4.830e−03	5.006e+03	1.9e−02	2.0e+04	17	15	112	14
osborneb	11	4.014e−02	3.166e+00	7.2e−06	6.5e+00	118	113	689	112
oscigrad	100000	1.062e−03	6.121e+08	2.7e+02	2.2e+09	10	10	66	9

Table 4.1 — Solution of 107 nonlinear problems with linesearch CR (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
oscipath	10	1.000e+00	1.000e+00	3.7e−06	1.0e+00	10001	10001	75635	10000
parkch	15	NaN	2.150e+03	NaN	2.5e+04	24	14	97	13
penalty1	1000	3.933e+08	1.114e+17	1.1e+07	2.4e+13	13	13	24	12
penalty2	200	4.712e+13	4.712e+13	3.6e+00	1.6e+07	12	12	232	11
penalty3	200	1.016e−03	1.584e+09	7.9e−01	2.2e+06	49	20	115	19
powellsg	5000	5.539e−04	2.688e+05	4.8e−03	1.6e+04	14	14	90	13
power	10000	9.819e+07	2.501e+15	8.7e+07	1.2e+14	13	13	76	12
quartc	5000	4.137e+09	6.241e+17	8.1e+06	1.3e+13	13	13	52	12
schmvett	5000	−1.499e+04	−1.429e+04	3.2e−05	7.5e+01	7	7	80	6
scosine	100	−4.591e+01	8.688e+01	9.1e+04	8.1e+02	10225	10001	25764	10000
scurly10	10000	3.636e+24	7.006e+31	8.3e+23	1.3e+30	13	13	116	12
scurly20	10000	4.752e+25	9.031e+32	1.1e+25	1.6e+31	13	13	108	12
scurly30	10000	1.914e+26	4.163e+33	4.7e+25	7.5e+31	13	13	104	12
sensors	100	−2.109e+03	−5.648e+01	6.2e−09	7.1e+01	33	18	49	17
sinquad	5000	−6.749e+06	6.561e−01	4.6e−03	5.1e+03	53	18	62	17
sparsine	5000	1.022e−01	5.173e+07	2.3e+00	3.0e+06	9	9	429	8
sparsqur	10000	9.566e−02	1.406e+07	7.3e−01	1.2e+06	13	13	70	12
spmsrtls	4999	1.341e−11	4.141e+03	1.0e−06	7.7e+01	16	15	554	14
srosenbr	5000	1.831e−08	4.850e+04	1.2e−04	1.1e+04	8	8	18	7
ssbrybnd	5000	6.737e−04	1.249e+05	3.0e−01	9.0e+05	18	17	16568	16
stratec	50	0.000e+00	0.000e+00	0.0e+00	0.0e+00	1	1	0	0
testquad	5000	4.741e+01	1.250e+09	1.5e+01	4.4e+07	7	7	748	6
tointgor	50	1.374e+03	5.074e+03	1.9e−05	6.0e+02	9	9	234	8
tointgss	5000	1.000e+01	4.499e+04	2.1e−04	4.2e+02	4	4	6	3
tointpsp	50	2.256e+02	1.828e+03	5.6e−05	1.1e+02	53	20	226	19
tointqor	50	1.175e+03	2.335e+03	1.9e−04	2.1e+02	6	6	66	5
tquartic	5000	2.749e−15	8.100e−01	1.5e−09	1.8e+00	43	33	71	32
tridia	5000	2.749e−15	8.100e−01	1.5e−09	1.8e+00	43	33	71	32
vardim	200	1.149e+08	3.257e+16	7.3e+09	1.6e+16	13	13	24	12
vareigvl	50	1.052e−13	1.126e+02	1.1e−06	4.2e+01	7	7	59	6
watson	12	1.602e−07	3.000e+01	4.8e−06	2.1e+02	14	14	112	13
woods	4000	7.877e+03	1.919e+07	1.1e−02	5.2e+05	10	10	38	9

Table 4.2: Solution of 107 nonlinear problems with linesearch CG

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
arglina	200	2.000e+02	1.000e+03	5.8e−13	5.7e+01	2	2	2	1
arglinb	200	9.963e+01	8.651e+15	1.2e−01	1.4e+15	2	2	2	1
arglinc	200	1.011e+02	8.353e+15	4.1e−01	1.4e+15	2	2	2	1
arwhead	5000	1.110e−12	1.500e+04	5.9e−06	4.0e+04	6	6	12	5

Table 4.2 — Solution of 107 nonlinear problems with linesearch CG (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
bdqrtic	5000	2.001e+04	1.129e+06	8.4e−01	1.5e+06	10	10	40	9
box	10000	−1.865e+03	0.000e+00	1.5e−06	5.0e+01	12	5	14	4
boxpower	20000	4.716e−02	1.764e+05	2.0e−02	1.7e+05	8	7	20	6
brownal	200	2.037e−08	2.010e+06	2.8e−02	5.7e+05	2	2	2	1
broydn7d	5000	1.875e+03	1.760e+04	4.2e−05	1.1e+03	1040	300	4326	299
brybnd	5000	1.510e−06	1.249e+05	7.2e−03	7.8e+03	8	8	78	7
chainwoo	4000	1.535e+02	1.445e+07	1.0e−01	4.2e+05	5324	1450	79310	1449
chnrosnb	50	1.877e−08	7.636e+03	1.9e−03	3.6e+03	72	45	978	44
chnrosnb_mod	100	6.582e−08	1.764e+04	2.3e−03	6.4e+03	213	99	1192	98
chnrsnbm	50	4.301e−09	8.633e+03	2.1e−03	4.4e+03	100	56	1176	55
clplatea	5041	−1.259e−02	0.000e+00	3.0e−07	1.0e−01	8	7	773	6
clplateb	5041	−5.095e−03	0.000e+00	2.2e−07	1.2e−02	4	4	456	3
clplatec	5041	−5.021e−03	0.000e+00	2.7e−08	9.9e−02	5	5	9370	4
cosine	10000	−9.999e+03	8.775e+03	9.1e−07	7.2e+01	8	7	19	6
cragglvy	5000	1.688e+03	2.749e+06	1.0e−01	2.8e+05	13	13	102	12
curly10	10000	−1.003e+06	−6.306e−01	6.9e−05	1.3e+02	27	14	116337	13
curly20	10000	−1.003e+06	−1.344e+00	1.2e−04	3.0e+02	29	15	138593	14
curly30	10000	−1.003e+06	−2.190e+00	3.2e−04	5.1e+02	35	15	127354	14
deconvu	63	1.773e−07	1.104e+02	7.4e−06	1.1e+02	23	15	254	14
dixmaana	3000	1.000e+00	2.850e+04	2.6e−04	1.2e+03	11	9	24	8
dixmaanb	3000	1.000e+00	4.724e+04	1.9e−04	2.0e+03	8	8	14	7
dixmaanc	3000	1.000e+00	8.248e+04	2.8e−04	3.7e+03	9	9	16	8
dixmaand	3000	1.000e+00	1.586e+05	2.1e−03	7.6e+03	10	10	18	9
dixmaane	3000	1.000e+00	2.850e+04	2.6e−04	1.2e+03	11	9	24	8
dixmaanf	3000	1.000e+00	4.104e+04	1.7e−03	1.9e+03	14	12	444	11
dixmaang	3000	1.000e+00	7.607e+04	1.8e−03	3.6e+03	15	13	202	12
dixmaanh	3000	1.000e+00	1.517e+05	4.1e−03	7.4e+03	18	13	225	12
dixmaani	3000	1.000e+00	2.002e+04	4.5e−05	1.0e+03	11	11	3696	10
dixmaanj	3000	1.000e+00	3.900e+04	1.4e−03	1.8e+03	12	12	128	11
dixmaank	3000	1.001e+00	7.400e+04	3.3e−03	3.6e+03	12	12	94	11
dixmaanl	3000	1.000e+00	1.496e+05	1.9e−03	7.4e+03	13	13	100	12
dixmaanm	3000	1.000e+00	9.358e+03	3.0e−04	4.4e+02	9	9	1382	8
dixmaann	3000	1.001e+00	2.018e+04	9.8e−04	1.0e+03	17	13	445	12
dixmaano	3000	1.000e+00	3.635e+04	1.9e−03	2.0e+03	20	15	395	14
dixmaanp	3000	1.000e+00	7.128e+04	1.9e−03	4.0e+03	17	15	326	14
dixon3dq	10000	5.207e−10	8.000e+00	6.4e−06	5.7e+00	6	6	21578	5
dqdrtic	5000	4.340e−09	9.041e+06	6.6e−04	8.5e+04	5	5	16	4
dqrtic	5000	2.888e+09	6.241e+17	6.8e+06	1.3e+13	13	13	56	12
edensch	2000	1.200e+04	7.358e+06	8.0e−03	1.0e+05	14	13	46	12
eg2	5000	5.549e+03	2.949e+05	1.5e−03	8.8e+03	9	9	34	8

Table 4.2 — Solution of 107 nonlinear problems with linesearch CG (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
engval1	5000	5.549e+03	2.949e+05	1.5e−03	8.8e+03	9	9	34	8
errinros	50	4.033e+01	1.102e+05	5.5e−02	1.2e+05	48	26	361	25
errinros_mod	100	7.838e+01	3.140e+05	5.8e−02	1.9e+05	41	25	148	24
errinrsm	50	3.855e+01	1.529e+05	5.6e−02	1.3e+05	34	24	280	23
extrosnb	1000	8.625e−03	3.996e+05	3.3e−02	3.8e+04	37	26	196	25
fletbv3m	5000	−2.492e+05	1.982e+02	2.7e−05	4.4e+01	66	50	119	49
fletcbv2	5000	−5.003e−01	−5.003e−01	1.7e−08	4.4e−06	2	2	9884	1
fletcbv3	5000	−1.302e+09	1.982e+02	3.3e+01	4.4e+01	10001	10001	15043	10000
fletcbv3_mod	100	−8.701e−02	−1.879e−02	2.8e−03	1.6e−03	10001	10001	10001	10000
fletchbv	5000	−3.044e+22	−2.302e+11	1.3e+10	2.8e+09	14970	10001	20057	10000
fletchcr	1000	4.824e−13	9.990e+02	2.3e−05	6.3e+01	1758	1488	33547	1487
fminsrf2	5625	1.000e+00	2.846e+01	9.7e−09	3.3e−01	465	47	101670	46
fminsurf	5625	1.000e+00	2.859e+01	5.3e−08	3.3e−01	349	43	72166	42
freuroth	5000	6.082e+05	5.049e+06	4.2e−03	5.5e+04	25	11	53	10
genhumps	5000	2.405e−06	1.281e+08	1.2e−03	6.0e+03	5593	4720	18566	4719
genrose	500	1.000e+00	1.870e+03	2.3e−04	3.0e+02	829	281	6893	280
genrose_nash	100	1.000e+00	4.041e+02	4.2e−06	1.3e+02	209	71	742	70
hilbertb	10	3.437e−14	5.102e+02	8.4e−07	1.1e+02	6	6	12	5
indef	5000	−2.322e+09	4.603e+03	3.2e+02	8.0e+01	10348	10001	29124	10000
liarwhd	5000	1.626e−04	2.925e+06	2.6e−02	4.8e+05	13	13	36	12
mancino	100	1.101e−01	1.103e+12	9.2e+02	2.9e+09	6	6	13	5
modbeale	20000	1.299e−03	1.264e+07	7.3e−02	3.1e+05	9	9	106	8
morebv	5000	1.053e−11	1.597e−07	1.0e−07	8.0e−04	3	3	5380	2
ncb20	5010	−1.458e+03	1.000e+04	9.4e−06	2.8e+02	76	38	734	37
ncb20b	5000	7.351e+03	1.000e+04	2.6e−04	2.8e+02	65	20	1992	19
noncvxu2	5000	1.161e+04	3.235e+11	1.9e+00	3.3e+06	2755	941	8218	940
noncvxun	5000	1.163e+04	3.335e+11	7.1e−01	3.6e+06	3127	942	11348	941
nondia	5000	2.570e−04	2.000e+06	1.3e−02	2.0e+06	3	3	4	2
nondquar	5000	1.301e−03	5.006e+03	1.3e−02	2.0e+04	24	18	196	17
osborneb	11	4.014e−02	3.166e+00	1.5e−08	6.5e+00	41	22	293	21
oscigrad	100000	1.146e−03	6.121e+08	3.9e+02	2.2e+09	10	10	64	9
oscipath	10	9.999e−01	1.000e+00	1.5e−03	1.0e+00	17132	10001	162076	10000
parkch	15	1.624e+03	2.150e+03	3.6e−03	2.5e+04	27	22	280	21
penalty1	1000	3.933e+08	1.114e+17	1.1e+07	2.4e+13	13	13	24	12
penalty2	200	4.712e+13	4.712e+13	1.3e+00	1.6e+07	12	12	250	11
penalty3	200	1.011e−03	1.584e+09	7.3e−01	2.2e+06	43	18	118	17
powellsg	5000	2.065e−03	2.688e+05	1.3e−02	1.6e+04	13	13	86	12
power	10000	4.738e+07	2.501e+15	8.0e+07	1.2e+14	13	13	80	12
quartc	5000	2.888e+09	6.241e+17	6.8e+06	1.3e+13	13	13	56	12
schmvett	5000	−1.499e+04	−1.429e+04	2.8e−05	7.5e+01	7	7	78	6

Table 4.2 — Solution of 107 nonlinear problems with linesearch CG (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
scosine	100	−8.359e+01	8.688e+01	2.7e+03	8.1e+02	10186	10001	1232544	10000
scurly10	10000	1.409e+24	7.006e+31	7.2e+23	1.3e+30	13	13	130	12
scurly20	10000	1.847e+25	9.031e+32	9.1e+24	1.6e+31	13	13	124	12
scurly30	10000	1.006e+26	4.163e+33	4.3e+25	7.5e+31	13	13	102	12
sensors	100	−2.109e+03	−5.648e+01	1.8e−11	7.1e+01	28	16	49	15
sinquad	5000	−6.749e+06	6.561e−01	4.3e−03	5.1e+03	37	16	54	15
sparsine	5000	5.750e−03	5.173e+07	1.1e+00	3.0e+06	47	23	3018	22
sparsqur	10000	6.964e−02	1.406e+07	6.5e−01	1.2e+06	13	13	64	12
spmsrtls	4999	2.706e−10	4.141e+03	1.8e−05	7.7e+01	47	17	981	16
srosenbr	5000	1.916e−08	4.850e+04	1.2e−04	1.1e+04	8	8	18	7
ssbrybnd	5000	3.273e−06	1.249e+05	1.7e−01	9.0e+05	349	76	35100	75
stratec	50	0.000e+00	0.000e+00	0.0e+00	0.0e+00	1	1	0	0
testquad	5000	1.768e+00	1.250e+09	1.5e+01	4.4e+07	7	7	1020	6
tointgor	50	1.374e+03	5.074e+03	7.5e−05	6.0e+02	8	8	218	7
tointgss	5000	1.000e+01	4.499e+04	3.8e−05	4.2e+02	4	4	6	3
tointpsp	50	2.256e+02	1.828e+03	1.3e−05	1.1e+02	31	14	182	13
tointqor	50	1.175e+03	2.335e+03	1.2e−05	2.1e+02	7	7	86	6
tquartic	5000	1.474e−15	8.100e−01	1.1e−09	1.8e+00	30	22	58	21
tridia	5000	1.474e−15	8.100e−01	1.1e−09	1.8e+00	30	22	58	21
vardim	200	1.149e+08	3.257e+16	7.3e+09	1.6e+16	13	13	24	12
vareigvl	50	9.344e−13	1.126e+02	4.7e−06	4.2e+01	8	8	66	7
watson	12	1.597e−07	3.000e+01	1.2e−05	2.1e+02	13	13	100	12
woods	4000	7.877e+03	1.919e+07	1.8e−01	5.2e+05	10	10	37	9

4.7.2 Detailed results for the trust-region method

Detailed results on individual problems are given in Table 4.3 and Table 4.4.

Table 4.3: Solution of 109 nonlinear problems with trust-region CR

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
arglina	200	$2.000e+02$	$1.000e+03$	$1.8e-13$	$5.7e+01$	3	3	4	2
arglinb	200	$1.011e+02$	$8.353e+15$	$3.1e-03$	$1.4e+15$	3	3	4	2
arglinc	200	$1.011e+02$	$8.353e+15$	$3.1e-03$	$1.4e+15$	3	3	4	2
arwhead	5000	$1.110e-12$	$1.500e+04$	$5.9e-06$	$4.0e+04$	6	6	11	5
bdqrtic	5000	$2.001e+04$	$1.129e+06$	$9.2e-01$	$1.5e+06$	10	10	29	9
box	10000	$-1.865e+03$	$0.000e+00$	$3.5e-07$	$5.0e+01$	7	7	17	6
boxpower	20000	$4.736e-02$	$1.764e+05$	$6.9e-02$	$1.7e+05$	11	6	22	10
brownal	200	$2.037e-08$	$2.010e+06$	$2.8e-02$	$5.7e+05$	2	2	2	1
broydn7d	5000	$1.846e+03$	$1.760e+04$	$9.8e-04$	$1.1e+03$	428	422	2239	427

Table 4.3 — Solution of 109 nonlinear problems with trust-region CR (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
brybnd	5000	$6.074e-08$	$1.249e+05$	$1.1e-03$	$7.8e+03$	11	11	67	10
chainwoo	4000	$2.799e+03$	$1.445e+07$	$3.7e-01$	$4.2e+05$	5656	3545	68977	5655
chnrosnb	50	$3.192e-06$	$7.636e+03$	$2.6e-03$	$3.6e+03$	95	53	973	94
chnrosnb_mod	100	$2.487e-08$	$1.764e+04$	$6.7e-04$	$6.4e+03$	184	122	1901	183
chnrsnbm	50	$4.924e-08$	$8.633e+03$	$9.7e-04$	$4.4e+03$	101	68	965	100
clplatea	5041	$-1.259e-02$	$0.000e+00$	$5.2e-07$	$1.0e-01$	21	15	681	20
clplateb	5041	$-5.095e-03$	$0.000e+00$	$3.6e-07$	$1.2e-02$	4	4	414	3
clplatec	5041	$-5.021e-03$	$0.000e+00$	$7.3e-07$	$9.9e-02$	5	5	10965	4
cosine	10000	$-9.999e+03$	$8.775e+03$	$3.0e-05$	$7.2e+01$	12	12	26	11
agglvy	5000	$1.688e+03$	$2.749e+06$	$1.1e-01$	$2.8e+05$	13	13	57	12
curly10	10000	$-1.003e+06$	$-6.306e-01$	$3.9e-05$	$1.3e+02$	24	21	48883	23
curly20	10000	$-1.003e+06$	$-1.344e+00$	$2.6e-04$	$3.0e+02$	24	21	47039	23
curly30	10000	$-1.003e+06$	$-2.190e+00$	$7.5e-05$	$5.1e+02$	26	22	48385	25
deconvu	63	$5.824e-08$	$1.104e+02$	$1.6e-06$	$1.1e+02$	20	13	257	19
dixmaana	3000	$1.000e+00$	$2.850e+04$	$5.5e-04$	$1.2e+03$	9	9	19	8
dixmaanb	3000	$1.000e+00$	$4.724e+04$	$1.8e-03$	$2.0e+03$	8	8	14	7
dixmaanc	3000	$1.000e+00$	$8.248e+04$	$1.6e-04$	$3.7e+03$	10	10	18	9
dixmaand	3000	$1.000e+00$	$1.586e+05$	$5.0e-03$	$7.6e+03$	10	10	18	9
dixmaane	3000	$1.000e+00$	$2.209e+04$	$3.8e-05$	$1.1e+03$	11	11	204	10
dixmaanf	3000	$1.000e+00$	$4.104e+04$	$1.1e-03$	$1.9e+03$	12	12	91	11
dixmaang	3000	$1.001e+00$	$7.607e+04$	$3.2e-03$	$3.6e+03$	12	12	46	11
dixmaanh	3000	$1.001e+00$	$1.517e+05$	$2.0e-03$	$7.4e+03$	13	13	60	12
dixmaani	3000	$1.000e+00$	$2.002e+04$	$2.4e-04$	$1.0e+03$	11	11	297	10
dixmaanj	3000	$1.000e+00$	$3.900e+04$	$9.4e-04$	$1.8e+03$	13	13	79	12
dixmaank	3000	$1.001e+00$	$7.400e+04$	$2.5e-03$	$3.6e+03$	13	13	56	12
dixmaanl	3000	$1.001e+00$	$1.496e+05$	$5.7e-03$	$7.4e+03$	13	13	46	12
dixmaanm	3000	$1.000e+00$	$9.358e+03$	$6.1e-05$	$4.4e+02$	10	10	507	9
dixmaann	3000	$1.001e+00$	$2.018e+04$	$9.3e-04$	$1.0e+03$	13	13	123	12
dixmaano	3000	$1.000e+00$	$3.635e+04$	$6.3e-04$	$2.0e+03$	14	14	134	13
dixmaanp	3000	$1.001e+00$	$7.128e+04$	$1.7e-03$	$4.0e+03$	14	14	89	13
dixon3dq	10000	$1.195e-04$	$8.000e+00$	$4.9e-06$	$5.7e+00$	6	6	10799	5
dqdrtic	5000	$6.563e-06$	$9.041e+06$	$5.1e-03$	$8.5e+04$	8	8	18	7
dqrtic	5000	$1.741e+09$	$6.241e+17$	$4.2e+06$	$1.3e+13$	21	21	54	20
edensch	2000	$1.200e+04$	$7.358e+06$	$2.4e-02$	$1.0e+05$	13	13	30	12
eg2	1000	$-9.989e+02$	$-8.406e+02$	$6.0e-09$	$5.4e+02$	4	4	6	3
engval1	5000	$5.549e+03$	$2.949e+05$	$7.9e-04$	$8.8e+03$	11	11	29	10
errinros	50	$4.034e+01$	$1.102e+05$	$1.1e-01$	$1.2e+05$	42	26	382	41
errinros_mod	100	$7.858e+01$	$3.140e+05$	$1.4e-01$	$1.9e+05$	30	18	220	29
errinrsm	50	$3.866e+01$	$1.529e+05$	$8.4e-02$	$1.3e+05$	36	20	300	35
extrosnb	1000	$4.800e-03$	$3.996e+05$	$1.5e-02$	$3.8e+04$	61	34	343	60

Table 4.3 — Solution of 109 nonlinear problems with trust-region CR (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
fletbv3m	5000	$-2.394e+05$	$1.982e+02$	$5.0e-06$	$4.4e+01$	16	12	30	15
fletcbv2	5000	$-5.003e-01$	$-5.003e-01$	$1.7e-08$	$4.4e-06$	2	2	4843	1
fletcbv3	5000	$-1.941e+09$	$1.982e+02$	$3.6e+01$	$4.4e+01$	10001	9728	25199	10000
fletcbv3_mod	100	$-2.047e+00$	$-1.879e-02$	$1.6e-08$	$1.6e-03$	43	39	103	42
fletchbv	5000	$-2.372e+17$	$-2.302e+11$	$3.7e+09$	$2.8e+09$	10001	10000	25863	10000
fletch	1000	$8.100e-11$	$9.990e+02$	$5.0e-05$	$6.3e+01$	2347	1564	28041	2346
fminsrf2	5625	$1.000e+00$	$2.846e+01$	$2.2e-09$	$3.3e-01$	221	212	1522	220
fminsurf	5625	$1.000e+00$	$2.859e+01$	$5.0e-07$	$3.3e-01$	710	703	2689	709
freuroth	5000	$6.082e+05$	$5.049e+06$	$4.2e-03$	$5.5e+04$	12	12	34	11
genhumps	5000	$5.468e-06$	$1.281e+08$	$1.4e-03$	$6.0e+03$	6453	6222	25419	6452
genrose	500	$1.000e+00$	$1.870e+03$	$4.1e-05$	$3.0e+02$	378	308	4362	377
genrose_nash	100	$1.000e+00$	$4.041e+02$	$4.1e-05$	$1.3e+02$	96	77	931	95
hilbertb	10	$2.216e-14$	$5.102e+02$	$6.8e-07$	$1.1e+02$	6	6	11	5
indef	5000	$-2.024e+13$	$4.603e+03$	$7.1e+01$	$8.0e+01$	10001	7246	30578	10000
indefm	100000	$-1.005e+07$	$9.207e+04$	$1.3e-06$	$3.6e+02$	24	20	69	23
liarwhd	5000	$1.825e-02$	$2.925e+06$	$2.7e-01$	$4.8e+05$	13	13	31	12
mancino	100	$5.784e-02$	$1.103e+12$	$6.6e+02$	$2.9e+09$	12	11	24	11
modbeale	20000	$1.041e-02$	$1.264e+07$	$2.1e-01$	$3.1e+05$	10	10	53	9
morebv	5000	$3.637e-11$	$1.597e-07$	$9.5e-08$	$8.0e-04$	3	3	451	2
ncb20	5010	$-1.459e+03$	$1.000e+04$	$2.6e-05$	$2.8e+02$	64	52	526	63
ncb20b	5000	$7.351e+03$	$1.000e+04$	$8.7e-05$	$2.8e+02$	33	21	1214	32
noncvxu2	5000	$1.326e+04$	$3.235e+11$	$3.2e+00$	$3.3e+06$	935	864	4710	934
noncvxun	5000	$1.335e+04$	$3.335e+11$	$3.5e+00$	$3.6e+06$	906	824	4605	905
nondia	5000	$2.570e-04$	$2.000e+06$	$1.3e-02$	$2.0e+06$	3	3	4	2
nondquar	5000	$1.519e-03$	$5.006e+03$	$5.7e-03$	$2.0e+04$	31	19	229	30
osborneb	11	$4.014e-02$	$3.166e+00$	$5.8e-07$	$6.5e+00$	26	21	186	25
oscigrad	100000	$1.062e-03$	$6.121e+08$	$2.7e+02$	$2.2e+09$	10	10	42	9
oscipath	10	$9.997e-01$	$1.000e+00$	$1.5e-02$	$1.0e+00$	10001	6302	86167	10000
parkch	15	$1.624e+03$	$2.150e+03$	$1.8e-02$	$2.5e+04$	28	20	234	27
penalty1	1000	$8.186e+08$	$1.114e+17$	$1.9e+07$	$2.4e+13$	18	18	34	17
penalty2	200	$4.712e+13$	$4.712e+13$	$3.6e+00$	$1.6e+07$	12	12	127	11
penalty3	200	$1.002e-03$	$1.584e+09$	$1.9e-01$	$2.2e+06$	32	25	117	31
powellsg	5000	$2.085e-03$	$2.688e+05$	$1.3e-02$	$1.6e+04$	14	14	57	13
power	10000	$6.432e+07$	$2.501e+15$	$6.0e+07$	$1.2e+14$	14	14	54	13
quartc	5000	$1.741e+09$	$6.241e+17$	$4.2e+06$	$1.3e+13$	21	21	54	20
schmvett	5000	$-1.499e+04$	$-1.429e+04$	$2.5e-06$	$7.5e+01$	8	8	55	7
scosine	100	$-9.900e+01$	$8.688e+01$	$7.7e-04$	$8.1e+02$	262	242	38423	261
scurly10	10000	$1.602e+24$	$7.006e+31$	$4.1e+23$	$1.3e+30$	15	15	74	14
scurly20	10000	$2.013e+25$	$9.031e+32$	$5.3e+24$	$1.6e+31$	15	15	72	14
scurly30	10000	$8.866e+25$	$4.163e+33$	$2.4e+25$	$7.5e+31$	15	15	71	14

Table 4.3 — Solution of 109 nonlinear problems with trust-region CR (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
sensors	100	$-2.055e+03$	$-5.648e+01$	$1.9e-10$	$7.1e+01$	17	15	51	16
sinquad	5000	$-6.757e+06$	$6.561e-01$	$6.8e-05$	$5.1e+03$	14	14	38	13
sparsine	5000	$3.184e-02$	$5.173e+07$	$8.0e-01$	$3.0e+06$	10	10	262	9
sparsqur	10000	$1.693e-01$	$1.406e+07$	$1.1e+00$	$1.2e+06$	13	13	39	12
spmsrtls	4999	$1.647e-08$	$4.141e+03$	$5.0e-05$	$7.7e+01$	14	14	207	13
srosenbr	5000	$1.831e-08$	$4.850e+04$	$1.2e-04$	$1.1e+04$	8	8	16	7
ssbrybnd	5000	$2.227e-03$	$1.249e+05$	$4.9e-01$	$9.0e+05$	27	19	7783	26
sscosine	5000	$-4.997e+03$	$4.387e+03$	$5.4e-03$	$5.9e+03$	441	316	1022191	440
stratec	10	$2.212e+03$	$2.818e+03$	$1.4e-02$	$4.7e+04$	49	39	302	48
testquad	5000	$7.180e+00$	$1.250e+09$	$5.6e+00$	$4.4e+07$	9	9	456	8
tointgor	50	$1.374e+03$	$5.074e+03$	$2.3e-05$	$6.0e+02$	9	9	125	8
tointgss	5000	$1.000e+01$	$4.499e+04$	$1.0e-05$	$4.2e+02$	9	9	24	8
tointpsp	50	$2.256e+02$	$1.828e+03$	$6.7e-05$	$1.1e+02$	29	24	126	28
tointqor	50	$1.175e+03$	$2.335e+03$	$1.8e-05$	$2.1e+02$	7	7	47	6
tquartic	5000	$2.424e-21$	$8.100e-01$	$1.4e-08$	$1.8e+00$	11	11	24	10
tridia	5000	$4.377e-04$	$1.250e+07$	$1.0e-01$	$4.1e+05$	9	9	580	8
vardim	200	$1.149e+08$	$3.257e+16$	$7.3e+09$	$1.6e+16$	13	13	24	12
vareigvl	50	$6.242e-11$	$1.126e+02$	$2.7e-05$	$4.2e+01$	8	7	33	7
watson	12	$1.602e-07$	$3.000e+01$	$4.8e-06$	$2.1e+02$	14	14	69	13
woods	4000	$7.877e+03$	$1.919e+07$	$4.4e-02$	$5.2e+05$	9	9	25	8

Table 4.4: Solution of 109 nonlinear problems with trust-region CG

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
arglina	200	2.000e+02	1.000e+03	1.8e−13	5.7e+01	3	3	4	2
arglinb	200	1.011e+02	8.353e+15	4.6e−02	1.4e+15	3	3	4	2
arglinc	200	1.011e+02	8.353e+15	4.6e−02	1.4e+15	3	3	4	2
arwhead	5000	1.110e−12	1.500e+04	5.9e−06	4.0e+04	6	6	11	5
bdqrtic	5000	2.001e+04	1.129e+06	8.4e−01	1.5e+06	10	10	29	9
box	10000	−1.865e+03	0.000e+00	1.9e−07	5.0e+01	10	9	25	9
boxpower	20000	4.736e−02	1.764e+05	6.9e−02	1.7e+05	12	7	29	11
brownal	200	2.037e−08	2.010e+06	2.8e−02	5.7e+05	2	2	2	1
broydn7d	5000	1.825e+03	1.760e+04	4.2e−05	1.1e+03	480	472	2345	479
brybnd	5000	6.154e−07	1.249e+05	4.3e−03	7.8e+03	10	10	57	9
chainwoo	4000	2.856e+03	1.445e+07	9.9e−02	4.2e+05	4939	3223	46889	4938
chnrosnb	50	2.153e−07	7.636e+03	2.7e−03	3.6e+03	95	58	810	94
chnrosnb_mod	100	4.134e−07	1.764e+04	3.3e−03	6.4e+03	173	117	1768	172
chnrsnbm	50	4.507e−10	8.633e+03	4.7e−04	4.4e+03	102	69	955	101
clplatea	5041	−1.259e−02	0.000e+00	3.2e−07	1.0e−01	28	22	1150	27
clplateb	5041	−5.095e−03	0.000e+00	2.2e−07	1.2e−02	4	4	459	3

Table 4.4 — Solution of 109 nonlinear problems with trust-region CG (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
clplatec	5041	−5.021e−03	0.000e+00	2.7e−08	9.9e−02	5	5	9374	4
cosine	10000	−9.999e+03	8.775e+03	2.5e−05	7.2e+01	12	12	26	11
cragglvy	5000	1.688e+03	2.749e+06	1.1e−01	2.8e+05	13	13	62	12
curly10	10000	−1.003e+06	−6.306e−01	5.1e−05	1.3e+02	24	20	57435	23
curly20	10000	−1.003e+06	−1.344e+00	1.7e−04	3.0e+02	22	18	64559	21
curly30	10000	−1.003e+06	−2.190e+00	4.2e−04	5.1e+02	18	15	59948	17
deconvu	63	4.700e−08	1.104e+02	4.1e−06	1.1e+02	26	16	319	25
dixmaana	3000	1.000e+00	2.850e+04	4.7e−04	1.2e+03	9	9	19	8
dixmaanb	3000	1.000e+00	4.724e+04	1.9e−03	2.0e+03	8	8	14	7
dixmaanc	3000	1.000e+00	8.248e+04	1.6e−04	3.7e+03	10	10	18	9
dixmaand	3000	1.000e+00	1.586e+05	4.7e−03	7.6e+03	10	10	18	9
dixmaane	3000	1.000e+00	2.209e+04	5.7e−05	1.1e+03	11	11	251	10
dixmaanf	3000	1.000e+00	4.104e+04	9.8e−04	1.9e+03	23	12	587	22
dixmaang	3000	1.000e+00	7.607e+04	2.3e−03	3.6e+03	27	13	892	26
dixmaanh	3000	1.000e+00	1.517e+05	4.8e−03	7.4e+03	29	14	294	28
dixmaani	3000	1.000e+00	2.002e+04	2.7e−04	1.0e+03	11	11	762	10
dixmaanj	3000	1.000e+00	3.900e+04	4.5e−04	1.8e+03	28	14	815	27
dixmaank	3000	1.000e+00	7.400e+04	1.8e−03	3.6e+03	13	13	68	12
dixmaanl	3000	1.001e+00	1.496e+05	5.0e−03	7.4e+03	13	13	49	12
dixmaanm	3000	1.000e+00	9.358e+03	5.7e−05	4.4e+02	10	10	1589	9
dixmaann	3000	1.000e+00	2.018e+04	5.9e−05	1.0e+03	29	15	1922	28
dixmaano	3000	1.000e+00	3.635e+04	1.4e−03	2.0e+03	27	14	987	26
dixmaanp	3000	1.002e+00	7.128e+04	3.2e−03	4.0e+03	13	13	85	12
dixon3dq	10000	5.207e−10	8.000e+00	6.4e−06	5.7e+00	6	6	10794	5
dqdrtic	5000	1.111e−09	9.041e+06	4.3e−04	8.5e+04	8	8	18	7
dqrtic	5000	6.837e+09	6.241e+17	1.2e+07	1.3e+13	20	20	53	19
edensch	2000	1.200e+04	7.358e+06	7.5e−02	1.0e+05	14	14	37	13
eg2	1000	−9.989e+02	−8.406e+02	6.0e−09	5.4e+02	4	4	6	3
engval1	5000	5.549e+03	2.949e+05	5.8e−04	8.8e+03	11	11	31	10
errinros	50	4.009e+01	1.102e+05	1.0e−01	1.2e+05	62	40	586	61
errinros_mod	100	7.846e+01	3.140e+05	1.2e−01	1.9e+05	39	23	314	38
errinrsm	50	3.856e+01	1.529e+05	1.1e−01	1.3e+05	41	24	331	40
extrosnb	1000	5.872e−03	3.996e+05	3.0e−02	3.8e+04	54	31	269	53
fletbv3m	5000	−2.394e+05	1.982e+02	1.3e−06	4.4e+01	18	14	34	17
fletcbv2	5000	−5.003e−01	−5.003e−01	1.7e−08	4.4e−06	2	2	4943	1
fletcbv3	5000	−7.478e+12	1.982e+02	3.7e+01	4.4e+01	10001	9993	24984	10000
fletcbv3_mod	100	−2.034e+00	−1.879e−02	5.0e−07	1.6e−03	32	26	73	31
fletchbv	5000	−1.074e+21	−2.302e+11	3.8e+09	2.8e+09	10001	9999	24970	10000
fletchcr	1000	1.308e−12	9.990e+02	4.4e−05	6.3e+01	2390	1415	29831	2389
fminsrf2	5625	1.000e+00	2.846e+01	3.1e−07	3.3e−01	1473	1469	3612	1472

Table 4.4 — Solution of 109 nonlinear problems with trust-region CG (cont'd)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
fminsurf	5625	1.000e+00	2.859e+01	4.0e−09	3.3e−01	1510	1506	3631	1509
freuroth	5000	6.082e+05	5.049e+06	5.4e−02	5.5e+04	11	11	28	10
genhumps	5000	2.154e+01	1.281e+08	3.1e+00	6.0e+03	10001	9726	32001	10000
genrose	500	1.000e+00	1.870e+03	7.0e−05	3.0e+02	582	465	4275	581
genrose_nash	100	1.000e+00	4.041e+02	6.4e−05	1.3e+02	164	128	1100	163
hilbertb	10	3.437e−14	5.102e+02	8.4e−07	1.1e+02	6	6	11	5
indef	5000	−4.959e+07	4.603e+03	9.8e+02	8.0e+01	10001	10001	29258	10000
indefm	100000	−9.865e+06	9.207e+04	1.3e−05	3.6e+02	377	328	1240	376
liarwhd	5000	1.535e−02	2.925e+06	2.5e−01	4.8e+05	14	14	33	13
mancino	100	2.340e−02	1.103e+12	4.2e+02	2.9e+09	12	11	23	11
modbeale	20000	3.158e−04	1.264e+07	2.8e−02	3.1e+05	11	11	65	10
morebv	5000	1.053e−11	1.597e−07	1.0e−07	8.0e−04	3	3	2692	2
ncb20	5010	−1.460e+03	1.000e+04	6.0e−05	2.8e+02	71	60	412	70
ncb20b	5000	7.351e+03	1.000e+04	7.9e−05	2.8e+02	31	21	1535	30
noncvxu2	5000	1.159e+04	3.235e+11	9.5e−01	3.3e+06	3628	3558	12209	3627
noncvxun	5000	1.227e+04	3.335e+11	2.8e+00	3.6e+06	3930	3882	12459	3929
nondia	5000	2.570e−04	2.000e+06	1.3e−02	2.0e+06	3	3	4	2
nondquar	5000	1.581e−03	5.006e+03	1.0e−02	2.0e+04	31	19	246	30
osborneb	11	4.014e−02	3.166e+00	5.8e−08	6.5e+00	31	24	230	30
oscigrad	100000	1.146e−03	6.121e+08	3.9e+02	2.2e+09	10	10	41	9
oscipath	10	9.997e−01	1.000e+00	1.7e−02	1.0e+00	10001	6194	83375	10000
parkch	15	1.624e+03	2.150e+03	4.6e−03	2.5e+04	32	23	223	31
penalty1	1000	8.186e+08	1.114e+17	1.9e+07	2.4e+13	18	18	34	17
penalty2	200	4.712e+13	4.712e+13	1.3e+00	1.6e+07	12	12	136	11
penalty3	200	1.002e−03	1.584e+09	3.5e−01	2.2e+06	35	26	129	34
powellsg	5000	2.277e−03	2.688e+05	1.4e−02	1.6e+04	14	14	57	13
power	10000	3.252e+07	2.501e+15	5.6e+07	1.2e+14	14	14	55	13
quartc	5000	6.837e+09	6.241e+17	1.2e+07	1.3e+13	20	20	53	19
schmvett	5000	−1.499e+04	−1.429e+04	7.3e−06	7.5e+01	7	7	50	6
scosine	100	−9.900e+01	8.688e+01	5.8e−04	8.1e+02	336	301	27244	335
scurly10	10000	2.554e+24	7.006e+31	1.2e+24	1.3e+30	14	14	76	13
scurly20	10000	3.493e+25	9.031e+32	1.5e+25	1.6e+31	14	14	71	13
scurly30	10000	1.703e+26	4.163e+33	7.1e+25	7.5e+31	14	14	66	13
sensors	100	−2.041e+03	−5.648e+01	2.1e−05	7.1e+01	17	14	48	16
sinquad	5000	−6.757e+06	6.561e−01	6.5e−05	5.1e+03	17	17	45	16
sparsine	5000	1.499e−03	5.173e+07	7.4e−01	3.0e+06	10	10	461	9
sparsqur	10000	1.322e−01	1.406e+07	9.9e−01	1.2e+06	13	13	41	12
spmsrtls	4999	2.434e−10	4.141e+03	4.2e−05	7.7e+01	32	25	523	31
srosenbr	5000	1.916e−08	4.850e+04	1.2e−04	1.1e+04	8	8	16	7
ssbrybnd	5000	2.774e−04	1.249e+05	5.6e−01	9.0e+05	6086	4554	143194	6085

Table 4.4 — Solution of 109 nonlinear problems with trust-region CG (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
sscosine	5000	−3.992e+03	4.387e+03	6.0e+03	5.9e+03	10001	9847	1955112	10000
stratec	10	2.212e+03	2.818e+03	2.7e−02	4.7e+04	52	40	310	51
testquad	5000	4.323e−01	1.250e+09	7.4e+00	4.4e+07	9	9	632	8
tointgor	50	1.374e+03	5.074e+03	3.8e−04	6.0e+02	8	8	104	7
tointgss	5000	1.000e+01	4.499e+04	4.5e−07	4.2e+02	9	9	25	8
tointpsp	50	2.256e+02	1.828e+03	1.0e−06	1.1e+02	51	38	193	50
tointqor	50	1.175e+03	2.335e+03	4.9e−05	2.1e+02	7	7	44	6
tquartic	5000	2.439e−21	8.100e−01	1.4e−08	1.8e+00	11	11	24	10
tridia	5000	1.432e−05	1.250e+07	9.7e−02	4.1e+05	9	9	648	8
vardim	200	1.149e+08	3.257e+16	7.3e+09	1.6e+16	13	13	24	12
vareigvl	50	6.049e−14	1.126e+02	9.6e−07	4.2e+01	9	8	40	8
watson	12	1.597e−07	3.000e+01	1.2e−05	2.1e+02	13	13	62	12
woods	4000	7.877e+03	1.919e+07	5.9e−02	5.2e+05	9	9	25	8

4.7.3 Detailed results for nonlinear least squares trust-region method

Detailed results for individual problems are given in Table 4.5 and Table 4.6.

Table 4.5: Solution of 47 nonlinear least-squares problems with LSMR

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
argtrig	200	8.9e−09	3.3e+01	1.6e−04	1.3e+03	9	631	631	8
ba-l1	57	2.3e−10	6.4e+04	2.2e−02	3.1e+05	7	34	34	6
ba-l16	200	8.9e−09	3.3e+01	1.6e−04	1.3e+03	9	631	631	8
ba-l73	33753	2.2e+07	1.2e+08	7.7e+06	3.1e+08	10001	32592	32592	10000
broydn3d	5000	1.5e−11	2.5e+03	2.0e−05	2.8e+02	7	36	36	6
broydnbd	5000	1.5e−08	6.2e+04	3.7e−04	3.9e+03	12	77	77	11
dmn15103	99	2.4e+02	2.0e+06	3.5e+01	5.3e+07	77	1725	1725	76
dmn15332	66	1.1e+02	2.5e+05	4.6e+00	5.8e+06	179	3298	3298	178
dmn15333	99	7.5e+01	2.4e+05	3.8e+00	5.5e+06	171	6922	6922	170
dmn37142	66	1.1e+02	1.3e+05	3.1e+00	3.8e+06	145	2190	2190	144
dmn37143	99	1.7e+02	7.5e+04	3.3e+00	3.8e+06	32	137	137	31
eigena	2550	1.8e−07	2.0e+04	2.9e−04	4.5e+02	100	4095	4095	99
eigenb	2550	6.3e−06	5.0e+01	3.7e−06	1.9e+01	1192	74133	74133	1191
eigenc	2652	2.4e−06	5.6e+03	8.2e−05	2.4e+02	430	53610	53610	429
inteqne	12	2.4e−17	3.2e−02	7.1e−09	3.1e−01	4	14	14	3
lukšan-vlček5.1	20	8.1e−08	2.3e+03	2.4e−04	1.5e+03	61	529	529	60
lukšan-vlček5.11	20	2.7e−08	4.5e+00	3.3e−06	6.2e+00	8	43	43	7
lukšan-vlček5.12	21	1.9e−08	4.2e+01	4.3e−06	2.7e+01	9	89	89	8
lukšan-vlček5.13	20	1.8e−07	2.5e+02	2.8e−05	6.9e+01	8	37	37	7

Table 4.5 — Solution of 47 nonlinear least-squares problems with LSMR (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
lukšan-vlček5.14	20	1.1e−03	1.6e+05	1.0e−02	5.7e+04	9	37	37	8
lukšan-vlček5.15	21	4.2e−01	6.4e+06	3.3e−01	4.1e+05	17	68	68	16
lukšan-vlček5.16	21	2.0e−07	5.6e+01	3.1e−05	5.2e+01	8	43	43	7
lukšan-vlček5.17	21	2.0e−07	1.4e+02	3.0e−05	7.0e+01	8	44	44	7
lukšan-vlček5.18	21	1.9e−09	1.5e+01	1.3e−06	7.1e+00	9	33	33	8
lukšan-vlček5.2	20	3.5e+01	7.8e+03	6.2e−03	6.9e+03	12	92	92	11
lukšan-vlček5.3	20	1.5e−05	2.2e+03	1.5e−03	1.5e+03	8	48	48	7
lukšan-vlček5.4	20	1.7e+00	2.4e+03	5.5e−03	5.7e+03	25	214	214	24
mgh19	11	4.4e−02	1.6e+01	7.1e−06	2.2e+01	57	507	507	56
mgh21	20	2.2e−11	1.2e+02	3.0e−06	3.7e+02	31	110	110	30
mgh22	20	5.5e−06	5.4e+02	4.0e−04	5.1e+02	9	44	44	8
mgh25	10	3.7e−06	1.1e+06	5.3e−02	2.2e+06	9	25	25	8
mgh26	10	1.4e−05	3.5e−03	6.4e−07	5.0e−02	29	245	245	28
mgh27	10	3.1e−07	1.4e+02	7.8e−05	1.7e+02	5	14	14	4
mgh28	10	2.0e−16	3.9e−04	4.0e−09	2.0e−02	4	35	35	3
mgh29	10	2.4e−17	3.2e−02	7.1e−09	3.1e−01	4	14	14	3
mgh30	10	1.2e−11	1.1e+01	2.1e−05	2.5e+01	6	33	33	5
mgh31	10	1.4e−12	1.8e+02	9.9e−06	4.1e+02	7	27	27	6
mgh32	10	5.0e+00	2.5e+01	7.0e−16	6.3e+00	2	4	4	1
mgh33	10	2.3e+00	4.3e+06	1.4e−09	3.1e+06	2	4	4	1
mgh34	10	3.1e+00	2.0e+06	4.3e−10	1.6e+06	2	4	4	1
mgh35	10	3.3e−03	1.7e−02	1.1e−06	6.7e−01	30	228	228	29
msqrta	1024	5.5e−07	4.0e+03	7.4e−05	1.7e+02	47	16071	16071	46
msqrtb	1024	2.2e−08	4.0e+03	1.6e−05	1.7e+02	34	6992	6992	33
nzf1	13	3.9e−15	5.0e+03	3.4e−07	9.3e+02	7	36	36	6
spmsqrt	4999	1.7e−10	2.1e+03	2.6e−05	3.9e+01	13	242	242	12
yatp1sq	123200	1.6e−01	1.3e+09	2.3e+00	8.1e+06	20	60	60	19
yatp2sq	123200	3.0e−08	3.8e+09	1.3e−02	3.8e+05	35	102	102	34

Table 4.6: Solution of 47 nonlinear least-squares problems with LSQR

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
argtrig	200	1.1e−11	3.3e+01	7.6e−05	1.3e+03	8	654	654	7
ba-l1	57	1.7e−09	6.4e+04	7.1e−02	3.1e+05	7	35	35	6
ba-l16	200	1.1e−11	3.3e+01	7.6e−05	1.3e+03	8	654	654	7
ba-l73	33753	7.4e+07	1.2e+08	5.3e+07	3.1e+08	10001	37754	37754	10000
broydn3d	5000	1.1e−10	2.5e+03	6.6e−05	2.8e+02	7	35	35	6
broydnbd	5000	2.4e−07	6.2e+04	1.8e−03	3.9e+03	12	62	62	11
dmn15103	99	1.7e+02	2.0e+06	3.8e+01	5.3e+07	174	2075	2075	173
dmn15332	66	5.5e+02	2.5e+05	5.5e+00	5.8e+06	957	11189	11189	956
dmn15333	99	4.7e+01	2.4e+05	5.4e+00	5.5e+06	1888	34421	34421	1887

Table 4.6 — Solution of 47 nonlinear least-squares problems with LSQR (cont'd and end)

model	nvar	$f(x)$	$f(x_0)$	$\|g(x)\|$	$\|g(x_0)\|$	$\#f$	$\#g$	$\#Hv$	$\#it$
dmn37142	66	1.1e+02	1.3e+05	3.7e+00	3.8e+06	114	1324	1324	113
dmn37143	99	1.4e+02	7.5e+04	5.1e−01	3.8e+06	97	439	439	96
eigena	2550	4.7e−08	2.0e+04	2.3e−05	4.5e+02	95	3943	3943	94
eigenb	2550	9.6e−07	5.0e+01	4.3e−06	1.9e+01	995	75269	75269	994
eigenc	2652	1.5e−08	5.6e+03	1.5e−04	2.4e+02	408	39554	39554	407
inteqne	12	2.3e−17	3.2e−02	7.0e−09	3.1e−01	4	14	14	3
lukšan-vlček5.1	20	8.2e−11	2.3e+03	1.7e−04	1.5e+03	61	468	468	60
lukšan-vlček5.11	20	2.1e−08	4.5e+00	2.7e−06	6.2e+00	8	45	45	7
lukšan-vlček5.12	21	2.2e−07	4.2e+01	2.5e−05	2.7e+01	8	67	67	7
lukšan-vlček5.13	20	1.3e−07	2.5e+02	2.1e−05	6.9e+01	8	37	37	7
lukšan-vlček5.14	20	8.0e−04	1.6e+05	7.5e−03	5.7e+04	9	36	36	8
lukšan-vlček5.15	21	2.1e−02	6.4e+06	7.8e−02	4.1e+05	20	88	88	19
lukšan-vlček5.16	21	1.4e−07	5.6e+01	2.3e−05	5.2e+01	8	45	45	7
lukšan-vlček5.17	21	2.0e−07	1.4e+02	3.0e−05	7.0e+01	8	44	44	7
lukšan-vlček5.18	21	1.9e−09	1.5e+01	1.3e−06	7.1e+00	9	33	33	8
lukšan-vlček5.2	20	3.5e+01	7.8e+03	6.1e−03	6.9e+03	11	82	82	10
lukšan-vlček5.3	20	5.6e−06	2.2e+03	6.4e−04	1.5e+03	8	49	49	7
lukšan-vlček5.4	20	1.7e+00	2.4e+03	4.4e−03	5.7e+03	29	223	223	28
mgh19	11	4.4e−02	1.6e+01	1.0e−05	2.2e+01	66	618	618	65
mgh21	20	1.8e−11	1.2e+02	2.7e−06	3.7e+02	33	112	112	32
mgh22	20	4.0e−06	5.4e+02	3.5e−04	5.1e+02	9	44	44	8
mgh25	10	3.7e−06	1.1e+06	5.3e−02	2.2e+06	9	25	25	8
mgh26	10	1.4e−05	3.5e−03	3.4e−07	5.0e−02	34	212	212	33
mgh27	10	3.1e−07	1.4e+02	7.8e−05	1.7e+02	5	14	14	4
mgh28	10	4.8e−16	3.9e−04	6.2e−09	2.0e−02	3	25	25	2
mgh29	10	2.3e−17	3.2e−02	7.0e−09	3.1e−01	4	14	14	3
mgh30	10	7.1e−12	1.1e+01	1.8e−05	2.5e+01	6	33	33	5
mgh31	10	1.3e−12	1.8e+02	1.0e−05	4.1e+02	7	27	27	6
mgh32	10	5.0e+00	2.5e+01	2.5e−16	6.3e+00	2	4	4	1
mgh33	10	2.3e+00	4.3e+06	1.1e−09	3.1e+06	2	4	4	1
mgh34	10	3.1e+00	2.0e+06	6.0e−10	1.6e+06	2	4	4	1
mgh35	10	3.3e−03	1.7e−02	1.1e−06	6.7e−01	36	267	267	35
msqrta	1024	2.3e−09	4.0e+03	1.7e−05	1.7e+02	36	7492	7492	35
msqrtb	1024	8.0e−11	4.0e+03	4.3e−06	1.7e+02	31	5290	5290	30
nzf1	13	1.8e−09	5.0e+03	2.3e−04	9.3e+02	7	32	32	6
spmsqrt	4999	1.6e−13	2.1e+03	3.3e−07	3.9e+01	12	291	291	11
yatp1sq	123200	1.6e−01	1.3e+09	2.3e+00	8.1e+06	26	81	81	25
yatp2sq	123200	2.3e−08	3.8e+09	1.1e−02	3.8e+05	35	102	102	34

CHAPITRE 5 DISCUSSION GÉNÉRALE

Dans le cadre de l'optimisation non linéaire continue sans contraintes, CG est quasi-systématiquement utilisée pour calculer les directions de mise à jour des approximations successives de la solution. Des extensions de CR ont déjà été proposées, telle que celle de Luenberger (1970), mais elles ne s'inscrivent pas dans des contextes de recherche linéaire et région de confiance où on souhaite que la direction retournée par CR soit une direction de descente pour l'objectif au point courant. L'absence de documentation sur CR en recherche linéaire et région de confiance, ainsi que la propriété de décroissance monotone du résidu sont les principales motivations de notre étude. L'article présenté au chapitre 4 constitue le cœur de ce mémoire et la base de la discussion qui suit.

Les modifications de CR proposées dans ce mémoire sont applicables pour résoudre à la fois des problèmes convexes mais aussi non convexes. Elles se basent sur la détection de courbure négative ou nulle dans les directions de recherche et dans les directions du résidu. Dans le cas d'un objectif convexe, les algorithmes sont équivalents à la version de base de CR, excepté qu'en régions de confiance la solution est tronquée si elle se trouve à l'extérieur de la région de confiance.

Les tests numériques effectués révèlent que CR est essentiellement équivalente à CG et parfois meilleure. CR se montre avantageuse en termes de nombre de produits hessien-vecteur effectués. Avec CR on observe en particulier une meilleure résolution des problèmes non convexes en recherche linéaire et région de confiance, mais aussi une performance légèrement supérieure sur les problèmes convexes en recherche linéaire. L'utilisation de CR semble alors tout à fait justifiée pour le calcul des directions en optimisation continue. Ces résultats soulignent davantage l'intérêt de l'étude menée et constituent un apport pour la communauté scientifique.

Les tests sur les problèmes aux moindres carrés non linéaires ont montré une équivalence des deux méthodes et un léger gain en termes d'évaluations du résidu, avec LSMR. Ils ont été effectués en utilisant les implémentations numériquement plus stables de CG et CR, qui sont LSQR et LSMR, et qu'il est recommandé d'utiliser plutôt que CGLS et CRLS pour une étude expérimentale.

Une discussion pourrait également avoir lieu concernant les critères utilisés pour la détection de courbure négative ou nulle. En effet, dans le cadre de la recherche linéaire, la courbure nulle est détectée en testant $p_{k-1}^T H p_{k-1} \leq \varepsilon \|p_{k-1}\|^2$ ou $r_{k-1}^T H r_{k-1} \leq \varepsilon \|r_{k-1}\|^2$ en référence à ce que proposent Dembo et Steihaug (1983). Cette condition vérifie également si la courbure

est proche de 0.

Cependant, dans un contexte de région de confiance, on souhaite tester séparément la courbure nulle de la courbure strictement négative. La courbure nulle est identifiée en s'inspirant de l'inégalité de Cauchy-Schwarz : on teste $|p_{k-1}^T H p_{k-1}| \leq \varepsilon \|p_{k-1}\| \|H p_{k-1}\|$. Une raison est le fait que $\|p_{k-1}\|^2$ n'est pas directement calculé dans les CG et CR de base, tandis qu'on dispose de Hp_{k-1}.

Ce critère s'inspirant de l'inégalité de Cauchy-Schwarz est également utilisé dans CR pour les régions de confiance afin de vérifier que la direction de recherche est une direction de descente pour la quadratique. On teste $p_{k-1}^T r_{k-1} \leq \varepsilon \|p_{k-1}\| \|r_{k-1}\|$.

De plus, le choix de ε a été fixé à 10^{-6} car ce paramètre doit avoir une petite valeur. Néanmoins, une autre valeur aurait pu être choisie et on aurait pu choisir des ε différents en fonction de la condition testée par exemple.

Le nombre maximal d'itérations majeures a été fixé à 10000 en recherche linéaire et région de confiance, afin d'éviter des temps de calcul trop longs, mais une autre valeur aurait pu être choisie. Les critères d'arrêt choisis sont des critères couramment utilisés.

Concernant les tests numériques, les méthodes itératives étant préférables pour des problèmes de grande taille, nous avons décidé de ne pas traiter ceux de dimension inférieure à 10. Ce choix nous permet de ne pas exclure trop de problèmes et d'avoir alors un nombre suffisant de problèmes à traiter. Cependant, nous ne disposons alors que de 17 problèmes non convexes de dimension supérieure à 10. Une bibliothèque proposant plus de problèmes convexes de grande taille serait utile pour la comparaison des méthodes.

Enfin, plusieurs critères de performance sont possibles pour tracer les profils. On aurait par exemple pu choisir de juger la performance en se basant sur la précision de la solution obtenue, c'est-à-dire regarder la méthode qui donne la plus faible valeur de l'objectif. Mais nous choisissons de comparer le nombre d'évaluations de l'objectif, du gradient et le nombre de produits hessien-vecteur effectués. Il s'agit de critères régulièrement utilisés en optimisation continue pour déterminer le coût d'un algorithme. De plus, étant donné le critère d'optimalité utilisé, qui est le même qu'on utilise CG ou CR, s'il est vérifié on peut considérer que l'optimalité a été atteinte.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

6.1 Synthèse des travaux

Dans ce mémoire, nous avons discuté des propriétés intéressantes de CR et des limites de CG qui justifient l'intérêt de l'étude. Nous avons présenté deux variantes de CR pour les méthodes de recherche linéaire et de région de confiance, ainsi qu'une adaptation de sa variante pour les problèmes aux moindres carrés linéaires, CRLS, pour traiter les cas de la courbure négative ou nulle.

La mise au point de ces méthodes a permis une comparaison de la performance de CR et CG par la résolution de problèmes de bibliothèques d'optimisation. Ces tests montrent que CR est tout aussi efficace que CG, parfois plus performante. La différence est notamment marquée lorsqu'on considère le nombre de produits effectués entre les hessiens et des vecteurs, et pour la résolution de problèmes non convexes. D'après ces tests, CR est également préférable pour résoudre des problèmes convexes dans un contexte de recherche linéaire. En ce qui concerne la résolution des problèmes aux moindres carrés en région de confiance, les deux méthodes semblent équivalentes, bien que LSMR effectue un peu moins d'évaluations du résidu que LSQR.

Nous établissons de plus des propriétés de convergence pour les méthodes proposées. En recherche linéaire on montre une convergence locale et globale de CR.

6.2 Limitations de la solution proposée

Les méthodes proposées présentent néanmoins certaines limitations. Tout d'abord, comme mentionné ci-dessus, le caractère itératif des algorithmes les rend particulièrement adaptés à la résolution de problèmes de grande taille. De plus, l'utilisation de formules récursives de mise à jour pour ρ_k, δ_k, π_k et μ_k, pourrait entraîner une accumulation d'erreurs numériques au fil des itérations et une déviation par rapport à l'algorithme analytique, en particulier dans le cas de problèmes mal conditionnés.

Enfin, malgré la modification de CRLS, il reste numériquement préférable d'utiliser LSMR, de par l'utilisation du processus de Lanczos (1950) qui rend cette dernière méthode plus stable. En effet, Björck et Saunders (2017) montrent que CRLS offre des résultats de moindre précision que LSMR.

6.3 Améliorations futures

Ci-dessus, nous mentionnons une possibilité de rendre la méthode plus stable numériquement. Il serait alors possible de faire appel à un préconditionneur pour éviter les effets liés au mauvais conditionnement. L'utilisation de l'itération de Lanczos est également envisageable pour améliorer la stabilité. MINRES, construit à partir de ce processus, est analytiquement équivalent à CR pour H définie positive et il est sans doute possible de le modifier de façon semblable à CR. De plus, une étude de la propagation d'erreurs liés aux formules récursives de mise à jour peut également être envisagée afin d'évaluer dans quels cas il pourrait être préférable de calculer les quantités directement.

Pour finir, la relation (2.16) démontrée par Yuan (2000, Théorème 2) montre un résultat important de décroissance obtenue par CG (Steihaug, 1983) dans un contexte de région de confiance. Il serait intéressant de trouver une relation similaire vérifiée par CR.

RÉFÉRENCES

A. Björck, T. Elfving et Z. Strakos : Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM Journal on Matrix Analysis and Applications*, 19(3):720–736, 1998.

Å. Björck : Use of conjugate gradients for solving linear least squares problems. *In* I. S. Duff, éditeur : *Conjugate Gradient Methods and Similar Techniques*, pages 49–71, Harwell, Didcot, UK, 1979. AERE Harwell.

Å. Björck et M. A. Saunders : Krylov subspace algorithms for overdetermined and underdetermined linear systems. Technical report, Stanford University, Stanford, CA, USA, 2017.

A. R. Conn, N. I. M. Gould et Ph. L. Toint : *Trust-Region Methods*, volume 1 de *MPS/SIAM Series on Optimization*. SIAM, Philadelphia, USA, 2000.

R. S Dembo et T. Steihaug : Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26(2):190–212, 1983.

J. E. Dennis, Jr et J. J. Moré : Quasi-Newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

D. C.-L. Fong : *Minimum-Residual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization*. Thèse de doctorat, Stanford University, 2011.

D. C.-L. Fong et M. A. Saunders : LSMR : An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

D. C.-L. Fong et M. A. Saunders : CG versus MINRES : An empirical comparison. *SQU Journal for Science*, 17(1):44–62, 2012.

R. Fourer, C. Maheshwari, A. Neumaier, D. Orban et H. Schichl : Convexity and concavity detection in computational graphs. *INFORMS Journal on Computing*, 22:26–43, 2010.

G. H. Golub et W. Kahan : Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2(2):205–224, 1965.

N. I. M. Gould, D. Orban et Ph. L. Toint : CUTEst : a Constrained and Unconstrained Testing Environment with safe threads for Mathematical Optimization. *Computational Optimization and Applications*, 60:545–557, 2015.

M. R. Hestenes et E. Stiefel : Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

T. Kloek : Conjugate gradients and conjugate residuals type methods for solving least squares problems from tomography. Bachelor's thesis, TU Delft, Delft, Netherlands, 2012.

C. Lanczos : An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards B*, 45:225–280, 1950.

K. Levenberg : A method for the solution of certain problems in least squares. *Quarterly Journal on Applied Mathematics*, 2:164–168, 1944.

C.-J. Lin et J. J. Moré : Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9:1100–1127, 1998.

D. G. Luenberger : The Conjugate Residual Method for Constrained Minimization Problems. *SIAM Journal on Numerical Analysis*, 7(3):390–398, 1970.

L. Lukšan, C. Matonoha et J. Vlček : Modified CUTE problems for sparse unconstrained optimization. Technical Report 1081, Institute of Computer Science, Academy of Science of the Czech Republic, Pod Vodàrenskou věžì 2, 182 07 Prague 8, 2010.

D. Marquardt : An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

J. Nocedal et S. Wright : Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

C. C. Paige et M. A. Saunders : Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

C. C. Paige et M. A. Saunders : LSQR : An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

T. Steihaug : The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

E. Stiefel : Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme. *Commentarii Mathematici Helvetici*, 29(1):157–179, 1955.

Y. Yuan : On the truncated conjugate gradient method. *Mathematical Programming*, 87 (3):561–573, 2000.