# Analyse numérique des méthodes proximales : décomposition et parallélisme

Moulay Es-Saïd Oualibouch

Thèse préparée par :

OUALIBOUCH MOULAY ES-SAÏD

Pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER - GRENOBLE 1

(ARRETES MINISTERIELS DU 5 JUILLET 1984 ET DU 23 NOVEMBRE 1988)

Spécialité : MATHEMATIQUES APPLIQUEES
Option : RECHERCHE OPERATIONNELLE

Titre

# *ANALYSE NUMERIQUE DES METHODES PROXIMALES : DECOMPOSITION ET PARALLELISATION*

Soutenue le 18 décembre 1992

Devant le jury :

**Président :**

| | |
|---|---|
| P. J. Laurent | Professeur Université Joseph Fourier |

**Membres :**

| | |
|---|---|
| C. Michelot | Professeur (Rapporteur), Université Bourgogne-Dijon |
| Nguyen V. H. | Professeur (Rapporteur), Facultés Université Namur (Belgique) |
| Pham D. T. | Professeur (Examinateur), INSA - Rouen |
| P. Mahey | Maître de Conférence (Examinateur), Université Joseph Fourier |
| D. Trystram | Professeur (Invité), INPG |

Thèse préparée au sein du laboratoire ARTEMIS/IMAG

*... A mes parents ...*

## *Remerciements*

Je tiens à remercier tout particulièrement Monsieur **Philippe Mahey**, mon directeur de thèse, pour ses enseignements et ses conseils tout au long de ce travail de recherche. Nos discussions et échanges d'idées m'ont énormément apporté.

Je remercie Monsieur **Pierre-Jean Laurent** pour l'honneur qu'il me fait en acceptant de présider le jury de cette thèse.

J'exprime ma grande gratitude envers Messieurs **Christian Michelot** et **Van Hien Nguyen** de m'avoir fait l'honneur de rapporter ce travail, pour leurs conseils et suggestions qui m'ont beaucoup apporté.

Je voudrais exprimer ma profonde gratitude à Monsieur **Pham Dinh Tao**, non seulement d'avoir accepté d'examiner ce travail, mais aussi d'avoir été présent tout au long de ce travail pour me faire profiter de son savoir faire dans le domaine de l'optimisation.

J'exprime mes remerciements à Monsieur **Denis Trystram** pour l'honneur qu'il me fait en acceptant de participer à ce jury et pour ses encouragements. C'est grâce à lui et à son équipe de recherche, que je me suis initié au parallèlisme, en particulier messieurs **Denis Delesalle** et **Philippe Michallon**. Merci encore une fois à tous.

Je profite de cette occasion pour remercier tous les membres du laboratoire ARTEMIS pour leur sympathie avec une pensée aux thésards de l'équipe de Recherche Opérationnelle.

Merci à toute ma famille, en particulier mes parents, ainsi que Mademoiselle Paola Ghiandoni pour leur soutien moral et leurs encouragements tout au long de ce travail.

# Table des matières

# *Introduction générale*

Dans ce travail, nous présentons une analyse théorique et numérique de certains algorithmes de décomposition pour l'optimisation convexe basés sur la méthode du point proximal. Des versions nouvelles numériquement performantes de ces méthodes sont proposées et justifiées et, à la lumière de quelques modèles d'application de Recherche Opérationnelle, leur implémentation sur calculateur parallèle est envisagée et validée.

Notre contribution peut être située à trois niveaux :

- Au niveau théorique, nous proposons une vue d'ensemble d'opérateurs dits "proximaux" issus d'opérateurs monotones dont les propriétés contractantes (plus exactement, ce sont des opérateurs fermement non expansifs) sont utiles, des théories de point fixe (cf Goebel et Kirk(1990) ) à l'analyse convexe (cf Moreau(1965) ) et à l'optimisation (cf Rockafellar(1976a) ). En particulier, nous étudions le cas de la somme de deux opérateurs maximaux monotones qui, bien que non nécessairement maximale, engendre des itérations convergentes.

- Au niveau numérique, notre attention s'est portée sur l'accélération de la convergence de la méthode de l'inverse partiel (Spingarn(1983)). Cette méthode, dérivée de l'algorithme du point proximal étudié par Rockafellar(1976a) entre autres, s'applique en particulier au problème de la minimisation d'une fonction convexe non nécessairement différentiable sur un convexe fermé. Le calcul du point proximal étant lui-même un problème de minimisation convexe, il n'est numériquement intéressant que dans certaines situations. C'est le cas quand la fonction est séparable, la méthode devient alors une méthode de décomposition. En pratique, la convergence est très lente et le calcul du point proximal de l'inverse partiel doit être accéléré en introduisant un paramètre. Nous montrons comment ce paramètre doit intervenir et comment sa valeur optimale peut être évaluée. Ces résultats théoriques sont confirmés par les expériences numériques aussi bien pour la "nouvelle" méthode, appelée Décomposition Proximale Paramétrée sur un Graphe, que pour les méthodes proposées antérieurement dans la littérature.

- Au niveau des algorithmes et de leur implémentation informatique, la principale motivation de ce travail reste la construction de nouvelles méthodes de décomposition pour l'optimisation convexe. Nous montrons en particulier comment le couplage entre les différents sous-systèmes peut être représenté par un sous-espace qui rend la décomposition proximale particulièrement attrayante. De plus, l'effet régularisateur de la méthode proximale rend la coordination décentralisée, c.a.d. permet aux sous-problèmes

de calculer une solution unique à chaque itération qui converge vers la solution optimale. Ce dernier aspect est fondamental pour les applications aux équilibres économiques avec information décentralisée (cf Arrow et Hurwicz(1960) ou Luna(1978) ). Un autre aspect intéressant est l'obtention d'algorithmes de Lagrangien Augmenté séparables dans le cas dual. Dans ce sens, ce travail rejoint ceux de Cohen et Zhu(1982) et de Bertsekas et Tsitsiklis(1989).Finalement,comme dans ce dernier ouvrage, l'implémentation parallèle de ces méthodes de décomposition est envisagée. Là encore, la simplicité et la séparabilité de l'itération proximale permettent de distribuer les calculs de chaque sous-problème sur autant de processeurs, ouvrant plusieurs possibilités pour implémenter la coordination, qui revient à effectuer des projections sur des sous-espaces de structures simples. Dans certaines applications comme les modèles de transport, nous avons pu proposer une version massivement parallèle de l'algorithme de décomposition proximale et nous avons pu l'implémenter sur une Connection Machine CM-2 (comportant plus de 32K processeurs) rejoignant par là les travaux de Zenios(1990).

Ce document est organisé comme suit :
Dans un premier Chapitre, nous allons donner des résultats théoriques généraux portant sur l'analyse convexe et la théorie des opérateurs maximaux monotones qui nous seront utiles par la suite.

Le Chapitre 2 est dédié à la description et l'étude de l'Algorithme Proximal pour la résolution d'un problème sans contraintes ainsi que des résultats de convergence concernant certaines versions de cet algorithme ; notamment l'algorithme du point proximal (AP2), l'algorithme de Tikhonov et leurs versions perturbées respectives ; ainsi qu'une application à l'analyse convexe. On parlera aussi d'une version due à Gold'stein.

Dans le Chapitre 3, nous allons traiter les variantes de l'algorithme proximal dans le cas d'un problème avec contraintes, notamment l'algorithme de décomposition proximale sur le graphe d'un opérateur maximal monotone, qui est une des principales contributions de cette thèse.

Le Chapitre 4 sera consacré à une synthèse des méthodes classiques de décomposition et l'application aux méthodes de décomposition de certains algorithmes traités au Chapitre 3.

Dans le Chapitre 5, nous décrirons les problèmes d'application que nous avons traités, notamment les problèmes de localisation, (dont la première modélisation, adaptée à l'algorithme de l'inverse partiel (voir chapitre 2), a été faite par Idrissi et al(1989) et les problèmes de transport ainsi que les résultats numériques obtenus. Une brève description du matériel informatique que nous avons utilisé, en particulier la Connection Machine 2 (CM-2) sera donnée.

Dans la modélisation de certaines applications notamment l'application aux problèmes de transport, nous avons eu besoin de certaines notions sur la théorie de la dualité selon Rockafellar(1970c). Certains résultats de cette théorie sont repris dans l'annexe et étendus au cas où les perturbations doivent appartenir à un sous-espace.

# Chapitre I

# Généralités :

# Analyse convexe et opérateurs maximaux monotones

# 1-1 Introduction à l'analyse convexe

## 1-1-1 Quelques définitions :

Soit X un espace de Hilbert. Dans ce travail, nous nous limiterons au cas de la dimension finie ( $X=\mathbb{R}^n$) ; quelques remarques concernant le cas de la dimension infinie seront tout de même rajoutées en diverses occasions.

### *Notations :*

Le produit scalaire usuel de deux éléments x et y de X sera noté <x,y>, la norme $\mathfrak{L}2$ d'un élément x de X sera notée ||x||. Si $x_1$ et $x_2$ sont deux éléments de X, le segment liant x1 et x2 sera noté

$$[x_1,x_2] = \{x \in X \mid x=\lambda x_1 +(1-\lambda)x_2 \; ; \; \lambda \in [0,1] \}$$

On notera ri(C) l'interieur relatif d'un ensemble C de X, c'est à dire l'ensemble des points interieurs relativement à la norme induite dans le sous-espace affine engendré par C.

**Définition 1-1-1:**

i- Un sous-ensemble C de X est dit convexe si

$$\forall x_1,x_2 \in C : [x_1,x_2] \subset C$$

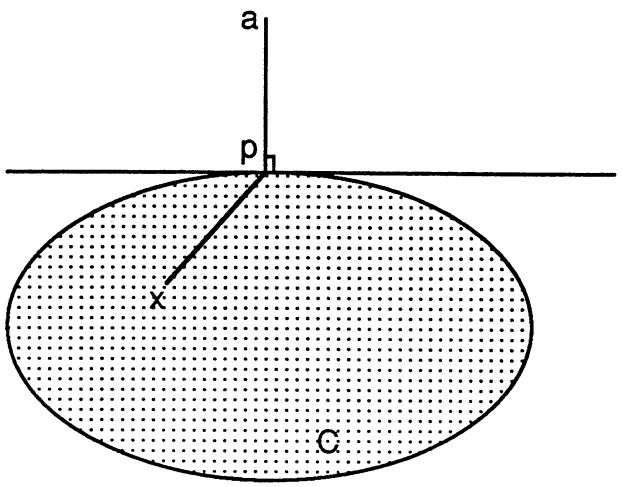
ii- Si $x \in C$, on appelle cône normal à C en x l'ensemble

$$N_C(x) = \{y \in X \text{ tq } \forall h \in C \;\; <h-x,y> \leq 0\}$$

iii- On appelle meilleure approximation de $a \in X$ sur un ensemble convexe fermé C, ou projection orthogonale de a sur C, notée $Proj_C a$, l'unique point de C tel que :

$$\forall x_1,x_2 \in C, <x - Proj_C a , a - Proj_C a> \leq 0$$

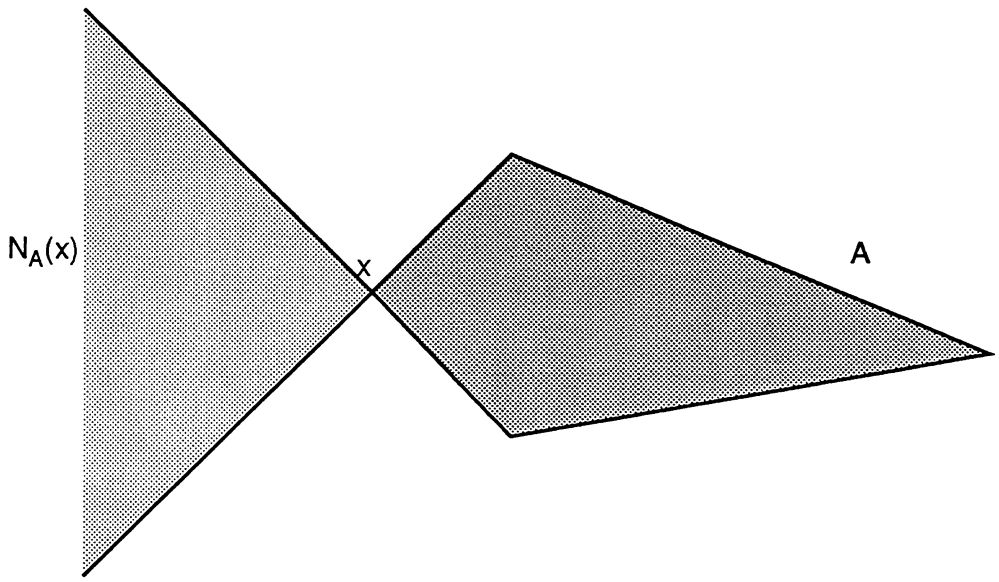
Nous représentons cette dernière propriété par la figure suivante :




$p = Proj_C(a)$

Dans le cas où C est un sous-espace vectoriel de X, $N_C(x)$ n'est autre que le sous-espace orthogonal de C. Dans la figure suivante, nous visualisons le cône normal d'un ensemble dans différentes situations.

Exemple




Exemple de cône normal d'un ensemble en un point

**Proposition 1-1-2 :**

Soit A un sous espace vectoriel de X et a un élément de X, alors pour tout x, élément de X, on a :

$$N_{A+\{a\}}(x) = A^{\perp} \quad \text{si } x \in A+\{a\}$$

$$= \emptyset \quad \text{sinon}$$

$$A+\{a\} = \{x+a \; ; \; x \in A\}$$

Où $A^{\perp}$ désigne le sous-espace orthogonal de A dans X.

Soit maintenant une fonction f sur X ($f : X \to \mathbb{R} \cup \{\infty\}$.)

**Définition 1-1-3 :**
On appelle respectivement le domaine, l'image et l'épigraphe de f, qu'on note respectivement par D(f), R(f) et epi(f) les ensembles suivants :

$$D(f)=\{ x\in X : f(x)\neq +\infty \}$$

$$R(f)=f(X) = \{\alpha\in \mathbb{R} \mid \exists x\in X ; f(x) = \alpha\}$$

$$epi(f)=\{(x,r)\in X\times\mathbb{R} : r\geq f(x)\}$$

**Définitions 1-1-4 :**
On dira qu'une fonction est convexe si et seulement si son épigraphe est convexe, ce qui est équivalent à :

**(1-1-1)** $$f(\lambda x_1 +(1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \quad \forall x_1,x_2 \in X, \forall\lambda\in(0,1)$$

Elle est dite fortement convexe de module a>0 si :

**(1-1-2)** $$f(\lambda x_1 +(1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) - (1/2)a.\lambda(1-\lambda)\|x_1 - x_2\|^2$$

$$\forall x_1,x_2 \in X, \forall\lambda\in(0,1)$$

Elle est dite semi-continue inférieurement (sci) si epi(f) est fermé.

Rappelons également qu'une enveloppe supérieure de fonctions s.c.i est s.c.i.

**Exemples**

**1 -** La norme $L^2$ est convexe sur X.

**2 -** Toute fonction continue est sci.

**3 -** La fonction indicatrice d'un ensemble convexe fermé est convexe et sci.

### 1-1-2 Enveloppe supérieure de fonctions affines continues :

**Définition 1-1-5 :**
Les fonctions affines continues sur X sont les fonctions du type

$$x \mapsto l(x) + a$$

où l est une forme linéaire continue sur X et a est un réel.

On note $\Gamma^{\circ}(X)$ l'ensemble des fonctions f sur X ($f\neq+\infty$), qui sont l'enveloppe supérieure d'une famille quelconque de fonctions affines continues.

Il découle de cette définition que $\Gamma^\circ(X)$ est l'ensemble des fonctions propres convexes semi-continues inférieurement sur X. Pour f élément de $\Gamma^\circ(X)$, on remarque que epi(f) est un ensemble convexe fermé non vide dans $X\times\mathbb{R}$, la projection sur cet ensemble est donc bien définie. Cette classe de fonctions est d'un intérêt particulier en analyse convexe, grâce à sa généralité et surtout aux propriétés dont elle bénéficie.

### 1-1-3 L'application prox

Dans ce paragraphe, nous allons rappeler certaines propriétés et résultats dûs à Moreau(1965) qui a été le premier à étudier l'application $prox_f$ associée à une fonction f, où $f\in\Gamma^\circ(X)$, sans considérer l'aspect itératif. Cet aspect itératif a été abordé plus tard pour la première fois par Martinet(1972), et puis par Rockafellar(1976b).

Cette application est définie de X dans lui même par la relation suivante :

**(1-1-3)** $$prox_f(x) = \arg\min_u \{f(u) + (1/2)||u-x||^2\}$$

où arg min f(x) désigne une solution du problème de minimisation de f.

**Proposition 1-1-6 :**

Soient $x\in X$ et $f\in\Gamma^\circ(X)$ ;

**1 -** La fonction définie sur X par

$$u \mapsto g(x,u) = (1/2)||u-x||^2+f(u) \qquad \textbf{(1-1-4)}$$

est fortement convexe par rapport à u, donc elle possède une valeur minimale, qu'elle atteint en un unique point.

**2 -** la fonction $f_1$ définie sur X par

$$f_1(x) = \inf\{ g(x,u) \ ; \ u\in X\}$$

est une minorante différentiable de f.

Cette proposition sera démontrée dans un cadre plus général au Chapitre 2.

**Définition 1-1-7 :**

**1-** L'unique point qui réalise le minimum de (1-1-3) est appelé le point proximal de f en x et est noté $prox_f(x)$.

**2 -** la fonction $f_1$ est appelée l'approximation de Moreau-Yosida de f , de rapport 1, cf. Yosida(1968) .

### 1-1-4 Fonction conjuguée et sous-différentiel d'une fonction

Soient $f \in \Gamma^{\circ}(X)$ et h une fonction affine qui minore f

$$h(x) = \langle x,y \rangle - b.$$

Il est évident que :

$$\mathrm{Sup}_{x \in X} \{ \langle x,y \rangle - f(x) \} \leq b$$

ce qui nous amène à parler de la fonction suivante :

$$g(y) = \mathrm{Sup}_{x \in X} \{ \langle x,y \rangle - f(x) \}$$

**Définition 1-1-8 :**

1- Soit $f \in \Gamma^{\circ}(X)$. La fonction conjuguée de Fenchel de f, notée $f^c$ (appelée aussi fonction polaire de f), est définie pour tout $y \in X$ par :

$$f^c(y) = \sup_{x \in X} \{ \langle y,x \rangle - f(x) \} \qquad \textbf{(1-1-5)}$$

Soient x et y deux éléments de X; d'après la relation (1-1-5) ci-dessus, on peut voir que :

$$f(x) + f^c(y) \geq \langle x,y \rangle. \qquad \textbf{(1-1-6)}$$

On verra plus loin que :

$$f \in \Gamma^{\circ}(X) \Leftrightarrow (f^c)^c = f \qquad \textbf{(1-1-7)}$$

f et $f^c$ sont dites alors mutuellement conjuguées.

**Définition 1-1-9 :**

Soient f et g deux fonctions mutuellement conjuguées de $\Gamma^{\circ}(X)$. On dit que deux éléments x et y de X sont conjugués par rapport à f et g si et seulement si :

$$f(x) + g(y) = \langle x,y \rangle.$$

On peut facilement voir que, si $f^c$ est finie, la fonction affine

$$u \mapsto \langle u,y \rangle - f^c(y)$$

est une minorante affine de f. Si, de plus, x et y sont conjugués, on a : $\forall u \in X$

$$\langle u,y \rangle - f^c(y) = \langle u,y \rangle - \langle x,y \rangle + f(x) \leq f(u)$$

donc

$$\langle u-x,y \rangle + f(x) \leq f(u) \qquad \textbf{(1-1-8)}$$

**Définition 1-1-10 :**

Soit f une fonction élément de $\Gamma^{\circ}(X)$ et $x,y\in X$. On dira que y est un sous-gradient de f au point x si et seulement si, pour tout $u\in X$, on a l'inégalité (1-1-8). L'ensemble des sous-gradients est le sous-différentiel de f en x. En résumé :

$$\partial f(x)=\{y\in X : \forall u\in X \quad f(u)\geq \langle u-x,y\rangle+f(x)\}$$

f est dite sous-différentiable au point x si $\partial f(x)$ est non vide.

Grâce à la proposition suivante, Moreau(1965) a établi le lien entre la notion de prox et celle de fonction conjuguée d'une fonction de la classe $\Gamma^{\circ}(X)$.

**Proposition1-1-11 :**

Soit $f\in\Gamma^{\circ}(X)$. On a les assertions suivantes :

**1 -** La fonction $f^c$ est un élément de $\Gamma^{\circ}(X)$, sa fonction conjuguée est f.

**2 -** Tout élément z de X est décomposable de façon unique en la somme de deux éléments x et y conjugués par rapport à f et $f^c$ et qui sont donnés par

$$x=prox_f(z) \text{ et } y=prox_{f^c}(z). \qquad \textbf{(1-1-9)}$$

La deuxième partie de cette proposition sera prouvée dans le cas général dans le chapitre suivant.

En résumé, si x et y sont conjugués par rapport à f et $\partial f^c$, la fonction h définie par :

$$h(u) = \langle u,y\rangle - f^c(y) = \langle u-x,y\rangle + f(x)$$

est un support affine de f en u = x et on a :

$$y\in\partial f(x) \text{ et } x\in\partial f^c(y)$$

**Proposition1-1-12 :**

Soit f une fonction élément de $\Gamma^{\circ}(X)$. On a les deux propriétés suivantes :

**1 -** $\partial f = (\partial f^c)^{-1}$

**2 -** $(I + \partial f^c)^{-1} + (I + \partial f)^{-1} = I$

(où I est l'application identité sur X )

**Preuve :**

**2 -** Soit $x\in X$ ; posons

$$y = (I + \partial f^c)^{-1}(x)$$

$$\Rightarrow \quad x \in y + (\partial f)^{-1}(y)$$

$$\Rightarrow \quad x - y \in (\partial f)^{-1}(y)$$

$$\Rightarrow \quad y \in (\partial f)(x - y)$$

$$\Rightarrow \quad x - y = (I + \partial f)^{-1}(x)$$

d'où la deuxième partie de la proposition.

**Remarque 1-1-13 :**

$$x = \text{prox}_f(z) \quad \Leftrightarrow \quad z - x \in \partial f(x)$$

$$\Leftrightarrow \quad z \in x + \partial f(x) = (I + \partial f)(x)$$

$$\Leftrightarrow \quad x \in (I + \partial f)^{-1}(z)$$

nous allons ensuite voir qu'on peut écrire que : $x = (I + \partial f)^{-1}(z)$

Notons par $f^{cc}$ la fonction conjuguée de Fenchel de $f^c$. Par définition, on peut voir que si l est une fonction affine continue minorant f, alors l minore aussi $f^{cc}$. Si en outre

$$l(u) = f(u)$$

pour un u fixé, on obtient que :

$$l(u) \leq f^{cc}(u) \leq f(u)$$

donc

$$l(u) = f^{cc}(u)$$

Une conséquence immédiate de la définition 1-1-10 est l'équivalence :

$$f(x) = \min_{v \in X} f(v) \Leftrightarrow 0 \in \partial f(x) \qquad \textbf{(1-1-10)}$$

Cette relation présente un des principaux liens entre l'optimisation convexe et la théorie des opérateurs maximaux monotones (cf § 1-2), cf Brézis(1973), Minty(1964) :
La recherche d'un zéro d'un opérateur maximal monotone généralise le problème de minimisation convexe.

On peut déduire de la proposition 1-1-11 que l'image réciproque d'un élément a de X par l'application $\text{prox}_f$ est l'ensemble convexe fermé, éventuellement vide :

$$[\text{prox}_f]^{-1}(a) = a + \partial f(a)$$

il est donc clair que l'ensemble des points fixes de l'application $\text{prox}_f$ coïncide avec l'ensemble des zéros de $\partial f$.

La proposition suivante examine dans quelle mesure, la multiplication par un scalaire et l'additivité, dans le calcul différentiel ordinaire, peuvent s'étendre au calcul sous-différentiel.

**Proposition 1-1-14 : (Rockafellar(1970c)**

Soient f et g deux éléments de $\Gamma^{\circ}(X)$ et $\lambda$ un réel positif non nul. Pour tout élément u de X on a immédiatement :

$$\partial(\lambda f)(u) = \lambda \partial f(u)$$

$$\partial(f + g)(u) \supset \partial f(u) + \partial g(u)$$

Si de plus

$$ri(dom(f)) \cap ri(dom(g)) \neq \emptyset$$

alors

$$\partial(f + g)(u) = \partial f(u) + \partial g(u)$$

**Exemples de fonctions prox :**

*Exemple 1*

Soit C un ensemble convexe fermé non vide et $\chi_C$ sa fonction indicatrice, définie par :

$$\chi_C(x) = \begin{cases} 0 & \text{si } x \in C \\ +\infty & \text{sinon} \end{cases}$$

Cette fonction est un élément de $\Gamma^{\circ}(X)$. Il suffit de reprendre la définition 1-1-10 pour voir que

$$\partial \chi_C(x) = N_C(x)$$

( $\chi_C$ : fonction indicatrice de C )

Si on pose

$$f(x) = \chi_C(x)$$

on a

$$prox_f(x) = \arg \min_{u \in X}\{f(u) + (1/2)||u-x||^2\}$$

$$= \arg \min_{u \in C}\{(1/2)||u-x||^2\}$$

$$= Proj_C(x)$$

**Conséquence de la proposition 1-1-11 :**

Soient P et Q deux cônes polyhèdriques mutuellement polaires, ce qui revient à écrire :

$$P = \{ x \in X : \langle x,y \rangle \leq 0 \ \forall y \in Q \}$$

(si $f=\chi_P$ alors $f^c=\chi_Q$)

tout élément z de X est égal à la somme de x et y, donnés par :

$$x = Proj_P(z) \text{ et } y = Proj_Q(z) ;$$

$$\langle x,y \rangle = 0$$

et c'est l'unique décomposition de z en la somme d'un élément de P et d'un élément de Q qui sont orthogonaux.

*Exemple 2*
Si f est fonction affine i.e.

$$f(u) = \langle a,u \rangle - b.$$

$$g(x,u) = \langle a, u \rangle + \frac{1}{2}||x - u||^2 - b$$

Il suffit d'écrire la condition d'optimalité en u pour voir que cette fonction atteint son minimum en

$$u = x - a.$$

donc

$$prox_f(x) = x - a$$

ainsi, toute translation est une application prox.

## *1-2 Opérateurs Maximaux monotones*

Revenons encore une fois à la définition 1-1-10 et remarquons que

**(1*)** $y \in \partial f(x) \Leftrightarrow \forall u \in X \quad f(u) \geq \langle u-x, y\rangle + f(x)$

si x' et y' sont deux autres éléments de X tels que

**(2*)** $y' \in \partial f(x')$ [ $\Leftrightarrow \forall u \in X \quad f(u) \geq \langle u-x', y'\rangle + f(x')$ ]

Si on remplace u par x' dans (1*) on a :

**(3*)** $f(x') \geq \langle x'-x, y\rangle + f(x)$

et si on remplace u par x dans (2*) on a :

**(4*)** $f(x) \geq \langle x-x', y'\rangle + f(x')$

En reécrivant (3*) en tenant compte de (4*), on obtient :

**(5*)** $\langle x-x', y-y'\rangle \geq 0$

On dit alors que $\partial f$ est un opérateur monotone. On peut voir facilement que $\partial(\lambda f)$, (où $\lambda>0$) est aussi monotone.

### 1-2-1 Quelques définitions

La théorie des opérateurs maximaux monotones (et son lien avec l'analyse convexe), outil fondamental dans l'étude des méthodes proximales, doit beaucoup à plusieurs auteurs, notamment Brézis(1973), Minty(1961, 1962, 1964), Rockafellar (1966, 1969) etc. Ci-dessous, nous avons rapporté certaines propriétés et résultats qui seront utiles dans la suite de ce travail .

Soit T une application de X dans $2^Y$ ou une multiapplication de X dans Y. Son graphe, noté Gr(T), est l'ensemble des couples (x,y) de XxY tels que $y \in Tx$ ; par abus de langage, il nous arrivera parfois de confondre T et Gr(T). Dans la suite Y=X sauf si une indication signale le contraire.

**Définition 1-2-0 :**

Soit T un opérateur sur X, nous définissons les ensembles Im(T) (ou R(T)) et Zer(T) par :

$$Im(T) = R(T) = \{y \in X \mid \exists x \in X : y \in Tx\}$$

$$Zer(T) = \{x \in X \mid 0 \in Tx\}$$

**Définition 1-2-1:**

**1-** Un opérateur T sur X est dit *monotone* si

$$\forall\, x,x' \in X \quad \forall\, y \in Tx\,,\ \forall\, y' \in Tx' \qquad \langle x-x',\, y-y' \rangle \geq 0 \qquad \textbf{(1-2-1)}$$

On note la classe de ces opérateurs, sur X, par $\mathcal{M}on(X)$.

**2-** Un opérateur $T \in \mathcal{M}on(X)$ est *maximal monotone* s'il est maximal pour l'inclusion dans $\mathcal{M}on(X)$ ; i.e. Gr(T) est maximal pour l'inclusion dans l'ensemble {Gr(A) : A monotone}.

**3-** Un opérateur $T \in \mathcal{M}on(X)$ est dit *fortement monotone* de rapport $\alpha$ s'il existe $\alpha$, réel positif tel que

$$\forall\, x,x' \in X \quad \forall\, y \in Tx\,,\ \forall\, y' \in Tx' \qquad (x-x',\, y-y') \geq \alpha \|x-x'\|^2 \qquad \textbf{(1-2-2)}$$

**4-** Un opérateur T sur X est dit *cycliquement monotone* si :

$$\forall\, x_1,\ldots,x_n \in X \text{ on a:} \qquad (x_2-x_1, Tx_1)+\ldots+(x_n-x_{n-1}, Tx_{n-1}) +(x_1-x_n, Tx_n) \leq 0 \qquad \textbf{(1-2-3)}$$

**Théorème 1-2-2 :**

Soit T un opérateur monotone sur X, T est maximal monotone si et seulement si

$$Im(T) = U\{Tx,\, x \in X\} = X$$

La démonstration de ce théorème fait intervenir le lemme de Zorn ; on en trouve une version dans Minty(1962).

**Théorème 1-2-3:** (Rockafellar(1966, 1970c))

Si f est convexe alors $\partial f$ est monotone, si de plus $f \in \Gamma^\circ(X)$, $\partial f$ est maximal monotone. Dans le cas où $T=\partial f$, T est fortement monotone si et seulement si f est fortement convexe .

Ceci ne veut pas dire que tout opérateur monotone est la multiapplication sous-différentielle d'une fonction convexe. Seuls le sont les opérateurs cycliquement monotones.

En fait, l'ensemble des opérateurs cycliquement monotones coïncide avec l'ensemble des opérateurs sous-différentiels des fonctions de $\Gamma^\circ(X)$, résultat que l'on peut trouver

dans Brezis(1973). Par exemple, la rotation d'angle $\Pi/2$ est un opérateur monotone sur $\mathbb{R}^2$ mais elle n'est pas sous-différentiel d'une fonction convexe (sa matrice n'est pas symétrique).

Soit C un ensemble non vide convexe fermé ; alors l'opérateur $N_C$ qui fait correspondre à un point x de X le cône normal à C en x est un opérateur maximal monotone et si A est un sous-espace vectoriel de X et a un élément de X alors

$$Gr(N_{A+a}) = (A+a)xA^{\perp}$$

On peut donc voir que tous les opérateurs constants de X sont maximaux monotones.

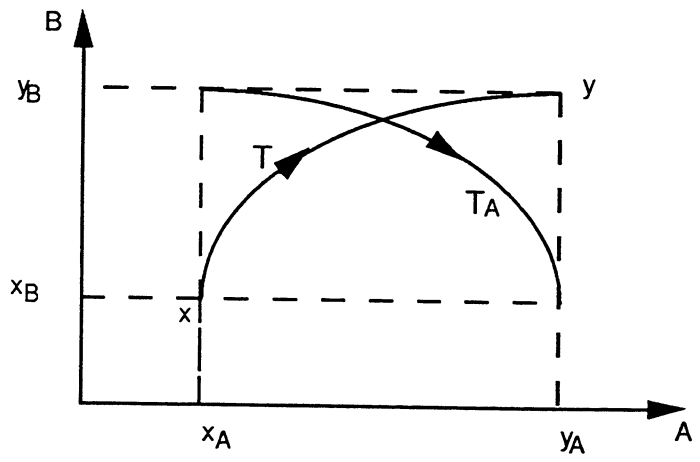
Ci-dessous, on va définir la notion d'inverse partiel d'un opérateur par rapport à un sous-espace vectoriel A de X (cf Spingarn(1983)) ; pour cela on adopte les notations suivantes : $B=A^{\perp}$, pour $x\in X$ on note

$$x_A=proj_A x \text{ et } x_B=proj_B x$$

**Définition 1-2-4 :**
Soient A un sous-espace vectoriel de X et T maximal monotone sur X. On définit l'inverse partiel de T par rapport à A, que l'on note $T_A$, comme suit :

$$(x,y)\in Gr(T) \Leftrightarrow (x_A+y_B, y_A+x_B)\in Gr(T_A) \qquad \textbf{(1-2-4)}$$

La figure suivante nous permet de visualiser cet opérateur




(figure visualisant l'inverse partiel d'un opérateur )

Cet opérateur est maximal monotone lorsque T l'est, cf Spingarn(1983).

**Proposition 1-2-5 :**
Soit T un opérateur monotone sur X. Les proprietés suivantes sont équivalentes :

0- $T_A$ est maximal monotone

i- T est maximal monotone

ii- $T^{-1}$ est maximal monotone

iii- Soit **a** élément de X, T+**a** est maximal monotone.

iv- Soit $\mathfrak{A}$ une matrice (n,n) inversible, l'opérateur $\mathfrak{A}^t T \mathfrak{A}$ est maximal monotone où

$$\mathfrak{A}^t T \mathfrak{A} = \{(\mathfrak{A}^{-1}x, \mathfrak{A}^t y) \mid (x,y) \in T\}.$$

v- I+T est surjectif.

**Preuve :**
L'équivalence entre 0 et i- est due à Spingarn(1983) et on peut la retrouver dans Lawrence et Spingarn(1986). Par contre, l'équivalence entre i-, ii-, iii- et iv est facile et on peut la trouver par exemple dans Eckstein(1989). L'équivalence avec v est due à Rockafellar(1966, 1970b)

**Proposition 1-2-6 :**
Soit T un opérateur maximal monotone.

**1 -** Le graphe de T est fermé dans XxX i.e.

$$(x_k \to x \,,\, y_k \to y \text{ et } (x_k,y_k) \in T) \Rightarrow ((x,y) \in T).$$

**2 -** Pour tout x élément de X, Tx est convexe fermé .

**preuve:**
**1 -** cf Eckstein(1989)
**2 -** Il suffit de voir que pour $y_1, y_2 \in Tx$, $T \cup \{(x, \lambda y_1 + (1-\lambda) y_2) : \lambda \in (0,1)\}$ est monotone. Ce résultat est également énoncé dans Aubin et Ekeland(1984b) [ prop.3, §6.7] et dans Eckstein(1989).

### 1-2-2 Opérateurs proximaux

**Définitions 1-2-7 :**
**1-** Soit N un opérateur sur X ; on dit que N est *non expansif* (ou *faiblement contractant*) si :

$$\| y' - y \| \leq \| x' - x \| \;;$$

$$\forall (x,y), (x',y') \in Gr(N) \;;$$

On note la classe de ces opérateurs sur X par $\mathcal{N}exp(X)$.

**2-** Un opérateur P sur X est dit *fermement non expansif* si :

$$\| y' - y\|^2 \leq \| x' - x\|^2 - \|(x'-y') - (x-y)\|^2 \ ; \qquad \textbf{(1-2-5)}$$

$$\forall\, (x,y)\ ,\ (x',y') \in Gr(P)\ ;$$

On note la classe de ces opérateurs, sur X, par $\mathcal{P}$rox(X).

**Remarque :**

La terminologie *fermement non expansif* a été utilisée par Lions et Mercier(1979). Le premier à utiliser cette notion est Martinet(1972) qui emploie le terme inexact de pseudocontraction. La terminologie "opérateur proximal" a été introduite par Lawrence et Spingarn(1986).

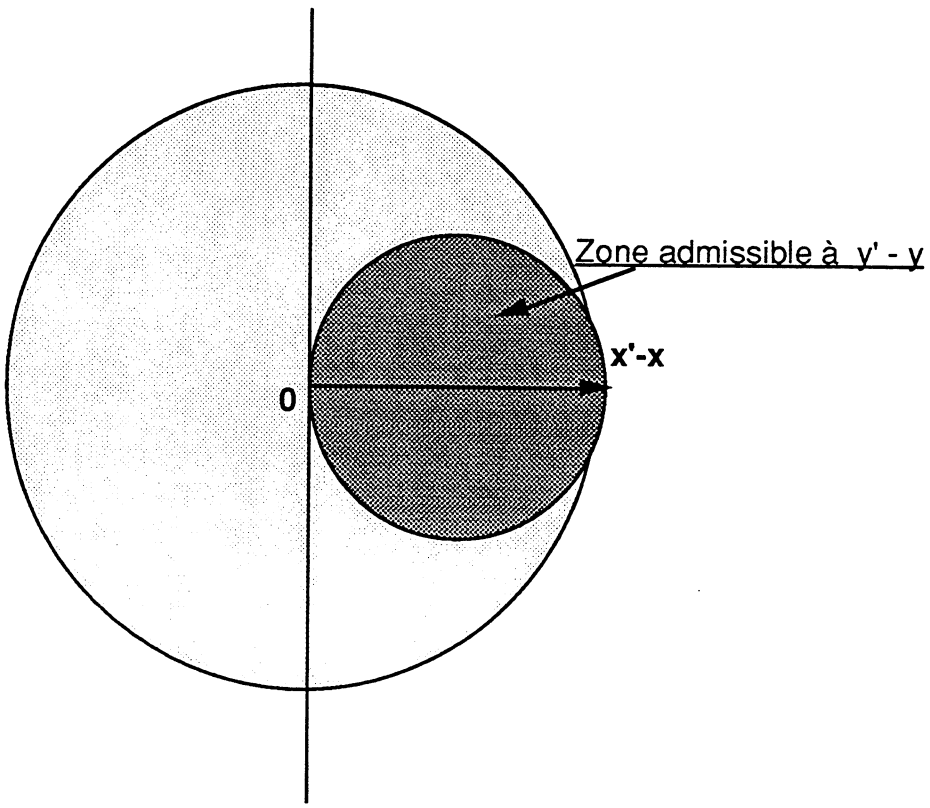
A partir de la définition précedente, nous pouvons déjà voir que tout opérateur fermement non expansif est non expansif. De plus, en posant :

$$b := y' - y \quad \text{et} \quad a := x' - x\ ;$$

l'inéquation (1-2-5) est équivalente à :

$$< b\ ,\ a - b > \geq 0$$

Cette inéquation nous permet de localiser l'ensemble auquel b peut appartenir, a étant donné. Cet ensemble est le cercle de centre a/2 et de rayon $\|a\|/2$, ce que nous pouvons voir graphiquement sur la figure ci-dessous :



Nous pouvons ainsi déduire facilement de ce qui précède que de tels opérateurs sont aussi monotones, donc univoques et uniformément continus (cf Eckstein(1989)). On a donc les inclusions suivantes :

$$\mathcal{P}\text{rox}(X) \subset \mathcal{N}\text{exp}(X) \cap \mathcal{M}\text{on}(X).$$

**Proposition 1-2-8 :**
Soit **J** une application de X dans lui même ; alors on a :

**1-** **J** est fermement non expansive si et seulement si l'application **2J-I** est non expansive. De façon équivalente, on a :

**2-** **J** est une application fermement non expansive si et seulement si elle peut être écrite sous la forme suivante :

$$\mathbf{J} = \frac{1}{2}(\mathbf{C} + \mathbf{I})$$

où C est non expansive sur X.

**Preuve :**
Cette démonstration a été partielment inspirée de Eckstein(1989).
Soit **J** une application proximale sur X. Alors pour tout x et y éléments de X, on a :

$$\|(2\mathbf{J} - \mathbf{I})x - (2\mathbf{J} - \mathbf{I})y\|^2 = 4\|\mathbf{J}x - \mathbf{J}y\|^2 - 4\langle \mathbf{J}x - \mathbf{J}y, x - y\rangle + \|x - y\|^2$$

donc

$$4(\|\mathbf{J}x - \mathbf{J}y\|^2 - \langle \mathbf{J}x - \mathbf{J}y, x - y\rangle) \leq 0$$

par conséquent, on a

$$\|(2\mathbf{J} - \mathbf{I})x - (2\mathbf{J} - \mathbf{I})y\|^2 \leq \|x - y\|^2$$

d'où (**2J - I**) est non expansive. Inversement, supposons que **C** = (**2J - I**) est non expansive, alors

$$\mathbf{J} = \frac{1}{2}(\mathbf{C} + \mathbf{I})$$

et soient x et y éléments de X, on a :

$$\|\mathbf{J}x - \mathbf{J}y\|^2 = (1/4)\|\mathbf{C}x - \mathbf{C}y\|^2 + (1/2)\langle \mathbf{C}x - \mathbf{C}y, x - y\rangle + (1/4)\|x - y\|^2$$

$$\leq (1/2)\|\mathbf{C}x - \mathbf{C}y\|^2 + (1/2)\langle \mathbf{C}x - \mathbf{C}y\ ,\ x - y\rangle$$

$$= (1/2)\langle (\mathbf{C}+\mathbf{I})x - (\mathbf{C}+\mathbf{I})y\ ,\ x - y\rangle$$

$$= \langle \mathbf{J}x - \mathbf{J}y, x - y\rangle$$

Donc **J** est fermement non expansive. ∎

**Proposition 1-2-9 :**

On peut déduire facilement de la relation (1-2-5) qu'une application **J** sur X est fermement non expansive si et seulement si pour tout x et y, éléments de X

$$\|\mathbf{J}x - \mathbf{J}y\|^2 \leq \langle \mathbf{J}x - \mathbf{J}y, x - y \rangle \qquad \textbf{(1-2-6)}$$

**Preuve :**

Soient (x,y) et (x',y') $\in$ Gr(**J**). Remarquons que

$$\|y'-y\|^2 \leq \|x'-x\|^2 - \|(x'-y')-(x-y)\|^2$$

$$\Leftrightarrow \quad \|y'-y\|^2 \leq \|x'-x\|^2 - (\|x'-x\|^2 - 2\langle x'-x, y'-y\rangle + \|y'-y\|^2)$$

$$\Leftrightarrow \quad 2\|y'-y\|^2 \leq 2\langle x'-x, y'-y\rangle.$$

On appelle opérateur **proximal** (ou résolvante) d'un opérateur maximal monotone T l'opérateur suivant :

$$J^T = (I+T)^{-1}$$

Par abus de langage, on appelle aussi résolvante de T, l'opérateur proximal de $\lambda T$ pour $\lambda>0$ et on le note par :

$$J_\lambda^T = (I+\lambda T)^{-1}$$

Un cas particulier important est le cas où $T=N_C$, où C est convexe fermé non vide. On a alors pour tout $\lambda>0$

$$J_\lambda^T = Proj_C$$

L'approximation de YOSIDA de l'opérateur T est définie par :

$$A_\lambda^T = \frac{I-J_\lambda^T}{\lambda}$$

**Théorème 1-2-10 :**

Soit T un opérateur monotone alors on a :

**1-** $J_\lambda^T$ est fermement non expansif et si de plus T est maximal alors, $dom(J_\lambda^T) = X$.

ce qui veut dire que $J_\lambda^T \in \mathcal{P}rox(X)$.

**2-** $$\langle x - x', A_\lambda^T x - A_\lambda^T x' \rangle \geq \lambda \|A_\lambda^T x - A_\lambda^T x'\|^2 \; ; \; \forall (x, x' \in X)$$

**Preuve :**

Cette démonstration sera faite pour le cas où $\lambda = 1$, le passage au cas $\lambda \neq 1$ étant facile (il suffit de considérer $\lambda T$ au lieu de T).

**1-** Soient (x,y) et (x',y') deux éléments de Gr(T) ; remarquons d'abord que :

$$y \in Tx \Leftrightarrow x = (I+T)^{-1}(x+y) \qquad \textbf{(1-2-7)}$$

par définition, on a :

$$\langle x'-x, y'-y \rangle \geq 0$$

$$\Leftrightarrow \quad \langle x'-x+y'-y, x'-x \rangle \geq \|x'-x\|^2$$

$\Leftrightarrow$ $(I+T)^{-1}$ est fermement non expansif (d'après la proposition 1-2-9 et la relation (1-2-7)).

La seconde partie de cette assertion peut être déduite de la proposition 1-2-5-v. Nous insitons sur le fait que la r´solvante n'est définie sur l'espace tout entier que si l'opérateur T est maximal, propriété qui n'est pas nécessaire pour que cette résolvante soit fermement nonxexpansive.

**2-** En réecrivant l'expression de l'approximation de Yosida de T, on obtient :

$$\langle x - x' , A_\lambda^T x - A_\lambda^T x' \rangle = \frac{1}{\lambda}\langle x - x' , x - x' - J_\lambda^T x - J_\lambda^T x' \rangle$$

$$= \frac{1}{\lambda}\|x-x'\|^2 - \frac{1}{\lambda}\langle x - x' , J_\lambda^T x - J_\lambda^T x' \rangle$$

par la relation (1-2-6), on obtient :

$$\langle x - x' , A_\lambda^T x - A_\lambda^T x' \rangle \geq \frac{1}{\lambda}\|x-x'\|^2 - \frac{1}{\lambda}\|J_\lambda^T x - J_\lambda^T x'\|^2$$

en utilisant la relation (1-2-5) on a :

$$\langle x - x' , A_\lambda^T x - A_\lambda^T x' \rangle \geq \frac{1}{\lambda}\|x-x'\|^2 - \frac{1}{\lambda}\left(\|x-x'\|^2 - \|(x-J_\lambda^T x) - (x'-J_\lambda^T x')\|^2\right)$$

d'où le résultat. ∎

**Proposition 1-2-11 (Lawrence et Spingarn(1986)) :**

Pour tout opérateur J fermement non expansif défini sur X tout entier, il existe un et un seul opérateur T maximal monotone tel que J soit la résolvante de T.

Comme dans Lawrence et Spingarn(1986), les opérateurs fermement non expansifs définis sur l'espace tout entier seront appelés proximaux dans la suite du texte.

Lawrence et Spingarn(1986) ont établi les liens précis entre les classes d'opérateurs $\mathcal{M}$on(X), $\mathcal{N}$exp(X) et $\mathcal{P}$rox(X). Le schéma ci-dessous représente ces liens.

Soient les applications $\alpha$, $\beta$ et $\gamma$, définies comme suit :

$$\alpha : \mathcal{P}\text{rox}(X) \to \mathcal{N}\text{exp}(X)$$

$$P \quad \mapsto (2P-I)$$

$$\gamma : \mathcal{M}\text{on}(X) \to \mathcal{P}\text{rox}(X)$$

$$T \quad \mapsto (I+T)^{-1}$$

et $\beta = \alpha o \gamma$

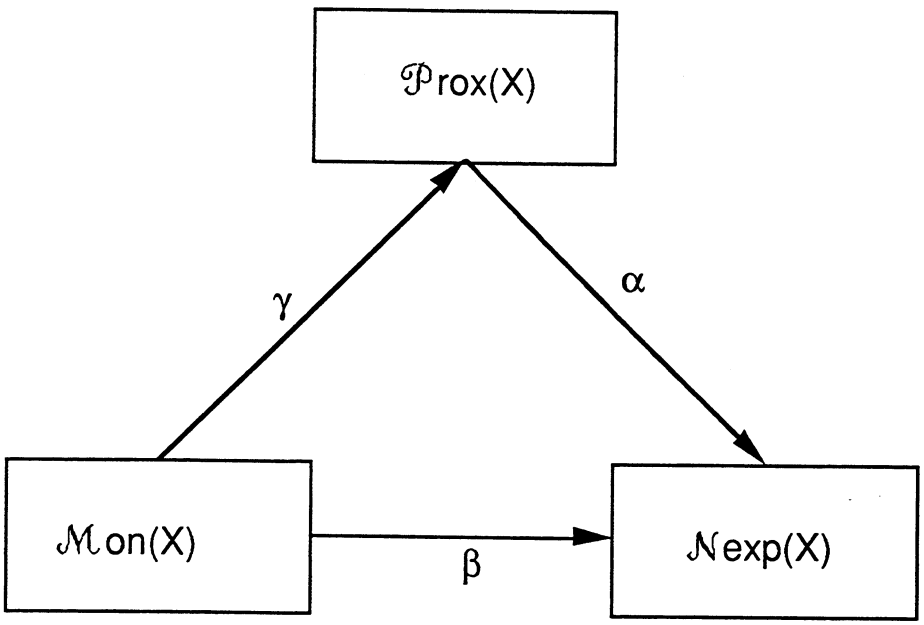



Figure représentant les liens entre les classes d'opérateurs

On reviendra sur ces relations au chapitre 3.

En ce qui concerne la résolvante d'un opérateur maximal monotone, signalons à titre d'exemple la proposition suivante :

**Proposition 1-2-12:**

Soit T un opérateur maximal monotone ; alors on a les propriétés suivantes :

i- $\forall\ z \in X\ \exists\ (x,y) \in Gr(T) : z=x+y$ , (x,y) unique **(1-2-8)**

ii- $J_{1+}^{T} + J_{1}^{T^{-}} = I$ (I : l'opérateur identité) **(1-2-9)**

iii- $A_{\lambda}^{T}$ est maximal monotone, univoque, défini sur X tout entier et Lipschitzien de rapport $1/\lambda$.

**Preuve:**

i- $x := J_1^T(z)$ ; $y := J_1^{T^{-1}}(z)$

donc à chaque élément de X on peut faire correspondre un élément unique du graphe de T. De plus, nous savons que si T est maximal monotone, alors sa résolvante est définie sur X tout entier. Ce qui veut dire que X est en bijection avec le graphe de T.

ii- $z = x+y = J_1^T(z) + J_1^{T^{-1}}(z)$

iii- Sachant que

$$A_\lambda^T(x) = J_{1/\lambda}^{T^{-1}}(x/\lambda)$$

et que $J_\lambda^T$ est faiblement contractant, le résultat est facilement obtenu. En ce qui concerne la maximale monotonie de la résolvante, nous faisons référence à Brezis(1973) prop(2-6). ∎

Un cas particulier de cette proposition est la proposition 1-1-12.

Dans la suite on suppose que T est un opérateur maximal monotone et on note $T^\circ x$ l'élément de norme minimale de Tx.

**Proposition 1-2-13 :**

i- $\overline{D(T)}$ est convexe et pour tout $x \in X$, on a

$$\lim_{\lambda \to 0} J_\lambda^T x = \mathrm{Proj}_{\overline{D(T)}} x$$

ii- Tx est convexe fermé et

$$A_\lambda^T x \to T^\circ x \text{ quand } \lambda \to 0$$

On peut trouver la preuve de cette proposition dans Brezis(1973).

Dans la proposition suivante, on établit la relation entre toutes les résolvantes d'un opérateur maximal monotone.

**Proposition 1-2-14 :**

SoientT un opérateur maximal monotone et $\alpha, \lambda > 0$. On a la relation suivante :

$$J_\lambda^T = J_{\alpha\lambda}^T \circ (\alpha I + (1-\alpha) J_\lambda^T)$$

**preuve :** Soient $x \in X$ et $\alpha, \lambda > 0$ ; posons

$$y = J_\lambda^T x \text{ et } z = J_{\alpha\lambda}^T \circ (\alpha I + (1-\alpha) J_\lambda^T) x$$

donc

$$\alpha x+(1-\alpha)y \in z+ \alpha.\lambda Tz$$

et $\alpha.\lambda(x-y)_{/\lambda} + (y-z) \in \alpha.\lambda Tz$

or $(x-y)_{/\lambda} \in Ty$

donc $(y-z) \in \alpha.\lambda(Tz - Ty)$

et par la monotonie de T on a

$$\langle y-z , z-y \rangle \geq 0$$

**Corollaire 1-2-15 :**

$$\forall\ \lambda>0,\ \{x : 0 \in Tx\} = \{y \in D(T)\ :\ J_{\lambda}^{T} y = y\}$$

**Définition 1-2-16 :**

Soient $\{T^n\}_n$ et T des opérateurs maximaux monotones. On dit que $T^n$ converge vers T au sens des graphes et on note $T^n \xrightarrow{G} T$ si

$$\forall\ (x,y) \in Gr(T),\ \exists\ \{(x_n,y_n)\}\ tq\left[(x_n,y_n) \in Gr(T^n)\right] et\left[x_n \to x \text{ et } y_n \to y\right]$$

Maintenant, nous allons énoncer une proposition faisant le lien entre la convergence, au sens des graphes, d'une suite $T_n$ d'opérateurs maximaux monotones vers un opérateur maximal monotone T et la convergence des suites des résolvantes et des approximations de YOSIDA de cet opérateur. Le lien est établi aussi avec la convergence au sens des graphes de la suite des opérateurs inverses partiels par rapport à un sous espace vectoriel donné.

**Proposition 1-2-17 :**

Soient $T^n$ (n>0) et T des opérateurs maximaux monotones de X. Les assertions suivantes sont équivalentes :

(i) $T^n \xrightarrow{G} T$ ;

(ii) $J_{\lambda}^{T^n} x \to J_{\lambda}^{T} x$ pour tout $x \in X$ et $\lambda>0$

(iii) $A_{\lambda}^{T^n} x \to A_{\lambda}^{T} x$ pour tout $x \in X$ et $\lambda>0$

(iv) $J_{\alpha}^{T^n} x \to J_{\alpha}^{T} x$ pour tout $x \in X$ et $\alpha>0$ fixé

(v) $A_{\alpha}^{T^n} x \to A_{\alpha}^{T} x$ pour tout $x \in X$ et $\alpha>0$ fixé

(vi) $T_A^n \xrightarrow{G} T_A$

Les cinq premières assertions peuvent être déduites de H.Attouch(1984) où on peut trouver aussi les définitions des différents types de convergence (au sens des graphes, convergence de Mosco, ...) ; l'équivalence avec l'assertion (vi) peut être déduite du travail de Tossings(1989).

Dans la suite de ce document, nous aurons besoin de certaines propriétés des opérateurs qui sont la composition d'opérateurs fermement non expansifs, ce qui nous facilitera la tâche dans les démonstrations des résultats de convergence de certains algorithmes que nous allons voir plus tard.

### 1-2-3 Somme de deux opérateurs maximaux monotones :

Dans ce paragraphe, on suppose que S et T sont deux opérateurs maximaux monotones. L'opérateur S+T est de toute évidence monotone, mais il n'est généralement pas maximal (par exemple, lorsque l'intersection de leurs domaines respectifs est vide). La question qui s'impose est de savoir sous quelles conditions cet opérateur est maximal monotone. Cette propriété a beaucoup d'importance dans l'étude de la convergence de certains algorithmes qui consistent à résoudre des problèmes équivalents à la recherche d'un zéro de la somme de deux opérateurs maximaux monotones.

**Exemple :**

On cherche à résoudre le problème suivant :

$$\text{Trouver } x\in X \mid x\in C_1\cap C_2 \qquad \textbf{(e1)}$$

où $C_1$ et $C_2$ sont deux sous-ensembles convexes fermés de X tels que

$$\text{int}(C_1\cap C_2)\neq\varnothing \qquad \textbf{(e2)}$$

Ce problème est équivalent à

$$\min_{x\in X} \quad \chi_{C_1}(x) + \chi_{C_2}(x)$$

ce qui revient à trouver $x^*\in X$ tel que :

$$0\in \partial(\chi_{C_1})(x^*) + \partial(\chi_{C_2})(x^*)$$

Pour obtenir la maximale monotonie de l'opérateur S+T, il a fallu imposer des conditions supplémentaires sur l'un des deux opérateurs S ou T (ou sur les deux). Par exemple, si l'un d'eux est Lipschitzien (donc univoque, défini sur X tout entier et localement borné), alors l'opérateur S+T est maximal monotone, cf Brezis(1973). C'est d'ailleurs un cas particulier du théorème suivant dû à R.T.Rockafellar(1970b).

**Théorème 1-2-18 :**

Si l'une des deux conditions suivantes est vérifiée

(a) $D(T)\cap D(S)\neq\varnothing$

(b) $\exists x\in \text{cl}(D(T))\cap\text{cl}(D(S))$

tel que S soit localement borné en x, alors (S+T) est maximal monotone.

Dans le Chapitre 3, et en particulier dans le paragraphe où nous traitons l'algorithme proximal projeté, nous considèrerons le problème de la recherche d'un zéro de la somme de deux opérateurs maximaux monotones. Certains algorithmes feront intervenir la composition des deux opérateurs proximaux associés. Nous observons tout d'abord que la composition de deux opérateurs proximaux n'est pas forcément proximale :

**Exemple :**

Soient les opérateurs linéaires A et B définis par

$$A = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \text{ et } B = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

alors

$$(2A-I) = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix} \text{ et } (2B-I) = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

et

$$C = AB = \begin{bmatrix} \frac{1}{3} & \frac{1}{9} \\ \frac{4}{9} & \frac{1}{3} \end{bmatrix}$$

Par un simple calcul on peut voir que pour tout x, élément de X, on a

$$||(2A-I)x|| \leq ||x||$$

et

$$||(2B-I)x|| \leq ||x||$$

alors que

$$\langle x'-x, Cx'-Cx \rangle \geq ||Cx'-Cx||^2$$

pour

$$x' = (-(2+\sqrt{2}),1) \text{ et } x = (0,0)$$

Ce qui veut dire que AB n'est pas fermement non expansif.

### 1-2-4 Itérations asymptotiquement régulières

Comme on le verra au Chapitre suivant, la méthode du point proximal consiste à appliquer l'itération

$$x^{k+1} = J_{\lambda_k}^T x^k$$

On sait déjà que la résolvante est fermement non expansive. Montrons que la suite

engendrée possède une propriété qui entraine la convergence.

**Définition 1-2-19 :**

Soit J un opérateur de X dans X, défini sur un convexe fermé C tel que $J(C) \subset C$. On dit que J est asymptotiquement régulier si pour tout $x \in X$, la suite $\{J^n x\}$ satisfait :

$$\lim_{n \to +\infty} \left\{J^n x - J^{n+1} x\right\} = \lim_{n \to +\infty} \left\{(I - J)J^n x\right\} = 0$$

**Théorème 1-2-20 :**

Soit J un opérateur non expansif sur X.

**i-** Si J est asymptotiquement régulier et admet un point fixe, alors la suite $\{J^n x\}$ converge vers un point fixe de J.

**ii-** Si J admet un point fixe alors l'opérateur $\lambda I + (1-\lambda)J$ est asymptotiquement régulier, pour tout $\lambda$ ; $0<\lambda<1$.

**Preuve :**

1- * cf Opial(1967)

** cf Krasnosel'skii(1955) pour $\lambda=1/2$ et Schaefer(1957) pour $0<\lambda<1$ ; Browder et Petryshyn(1966)

Donc, l'affirmation ii- implique en particulier qu'un opérateur proximal admettant un point fixe est asymptotiquement régulier.

**Théorème 1-2-21 :**

Soient $J_1, ..., J_n$ des opérateurs fermement non expansifs sur X (donc univoques) tels que $B = J_1 \circ ... \circ J_n$ admette un point fixe. Alors, si la suite $\{x^k\}$ définie par $x^{k+1}=B\,x^k$ avec $x^0 \in X$ est bornée, elle converge vers un point fixe de B.

**Preuve :** cf Martinet(1972).

# Chapitre II

# L'algorithme du point proximal et ses variantes :

# Cas sans contraintes

## 2-1 L'Algorithme du Point Proximal

### 2-1-1 Introduction :

Dans ce chapitre, nous nous intéressons à l'algorithme qui est en quelque sorte la base de notre travail : l'**A**lgorithme du **P**oint **P**roximal (AP2) qui, comme nous l'avons dit dans l'introduction, a été étudié de façon assez générale, dans le cadre des opérateurs maximaux monotones, par R. T. Rockafellar(1976). Notons que, dans toute la suite, T désignera un opérateur maximal monotone. Le problème général est :

Trouver $x \in X$ tel que

$$0 \in Tx \qquad \textbf{(2-1-1)}$$

c'est à dire trouver un zéro de l'opérateur T .

**Proposition 2-1-1:**

x est un zéro de T si et seulement si x est un point fixe de $J_\lambda^T$

**Preuve :**

$$\begin{aligned} & 0 \in Tx \\ \Leftrightarrow \quad & 0 \in \lambda Tx \\ \Leftrightarrow \quad & x \in (I+\lambda T)x \\ \Leftrightarrow \quad & x = J_\lambda^T x \end{aligned}$$

(cette dernière egalité est une conséquence du théorème 1-2-10) ∎

D'où l'idée d'itérer la suite appelée *itération proximale* :

$$x^{k+1} = J_{\lambda_k}^T x^k \qquad \textbf{(2-1-2)}$$

L'algorithme résultant **(AP2)** est dû à Rockafellar (1976), qui a étudié en détail sa convergence.

### 2-1-2 L'algorithme du point proximal :

***L'Algorithme (AP2)***

*1) **Initialisation :***

$$x_0 \in X,\ k=0\ ;$$

*2) **Itération Proximale :***

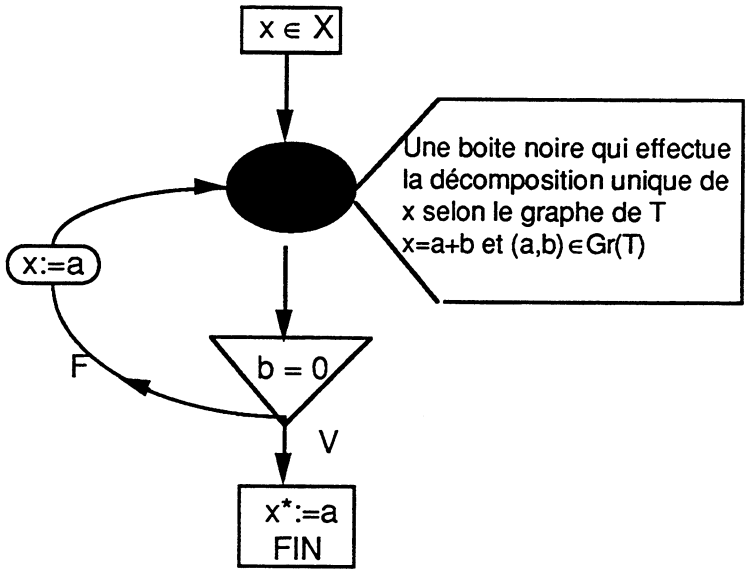
$$x^{k+1} = J_{\lambda_k}^T x^k \qquad \textbf{(2-1-3)}$$

*3) **Test***

**Si** $0 \in Tx^k$ alors fin et on a une solution

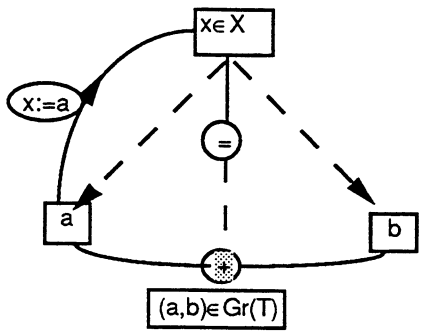
**Sinon** k:=k+1et retour à 2)

Cet algorithme peut être représenté par l'organigramme suivant :




Schema équivalent à une itération proximale

Une itération de cet algorithme peut être représentée par le schéma suivant :




- figure 3-
représentation de l'itération proximale

Cette représentation met en évidence le fait qu'une itération proximale peut être considérée comme la décomposition unique selon le graphe de T (resp. $\lambda T$) quand $\lambda=1$ (resp. $\lambda \neq 1$) ; pour comprendre cette interprétation nous avons choisi l'exemple suivant :

**Exemple 1 :**

Soit A un sous-espace vectoriel de X et B le sous-espace orthogonal à A ($B = A^{\perp}$). On sait que tout élément de X est décomposable de façon unique en la somme d'un élément

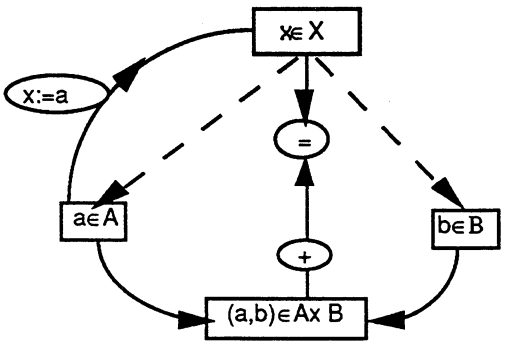
de A et d'un élément de B. Ceci est équivalent à une itération proximale appliquée au point à décomposer. En effet, lorsque $T = \partial x_A$, alors

$$Gr(T) = AxB$$

et on peut montrer facilement que :

$$J_\lambda^T x = Proj_A x = x_A$$

et dans ce cas le schéma précédent sera représenté comme suit :




- figure 4-

représentation de l'itération proximale
dans un cas particulier

## 2-2 Interprétation de l'algorithme (AP2) :

### 2-2-1 Analogie entre (AP2) et la méthode d'Euler Rétrograde :

L'algorithme (AP2) peut être vu comme la discrétisation de la méthode d'*Euler implicite* (ou progressif) appliquée à l'inclusion différentielle ci-dessous :

$$\frac{dx}{dt} \in Tx(t), \quad x(0)= x^0 \qquad \textbf{(2-2-1)}$$

On sait que tout opérateur maximal monotone T engendre un semi-groupe non linéaire contractant $\{S(t), t \geq 0\}$ d'applications sur la fermeture topologique du domaine de T, cl(D(T)) (cf Bruck(1975)). S(t)x=x(t), où x(t) est l'unique solution de l'inclusion différentielle (2-2-1). Cette dernière peut être discrétisée de deux manières différentes ; la première est une discrétisation d'Euler explicite :

$$\frac{x^k - x^{k+1}}{\lambda_k} \in T(x^k), \qquad \textbf{(2-2-2)}$$

la deuxième est une discrétisation d'Euler implicite :

$$\frac{x^{k-1}-x^k}{\lambda_k} \in T(x^k) \qquad \textbf{(2-2-3)}$$

Selon la discrétisation choisie, on obtient deux itérations différentes. La première est :

$$x^{k+1} = x^k - \lambda_k y^k \qquad \text{où } y^k \in Tx^k \text{ et } \lambda_k>0 \qquad \textbf{(2-2-4)}$$

qui n'est rien d'autre que l'algorithme du sous-gradient lorsque $T = \partial f$ ; $f \in \Gamma^\circ(X)$. Par contre, la discrétisation (2-2-3) nous donne l'itération suivante :

$$x^{k-1} \in (I + \lambda_k T)x^k$$

ce qui est équivalent à :

$$x^{k+1}=J^T_{\lambda_{k+1}} x^k$$

Par conséquent l'algorithme (AP2) correspond à une discrétisation *d'Euler implicite* de l'inclusion différentielle (2-1-5).

En considérant toujours le cas où $T=\partial f$, $f \in \Gamma^0(X)$, l'algorithme s'écrit :

$$x^{k+1}=x^k - \lambda_{k+1}\, y^{k+1}$$

où $y^{k+1} \in \partial f(x^{k+1})$

Schéma dont la convergence peut être déduite directement du théorème de Bruck (1975) suivant (on y considère que $T=\partial f$, $f\in\Gamma^0(X)$) :

**Théorème 2-2-1 :**

Soit $f\in\Gamma^0(X)$ admettant au moins un minimum. Alors pour tout $x_0$ élément de cl(Dom(f)), il existe une et une seule trajectoire,

$$x(t) : [0,\infty) \rightarrow X$$

qui est absolument continue sur $[\delta,\infty)$ pour tout $\delta>0$ et qui satisfait :

(i) $x(t) \in \mathrm{Dom}(\partial f)$ pour tout $t > 0$.

(ii) $\dfrac{\partial x(t)}{\partial t} \in -\partial f(x(t))$ pour tout $t > 0$

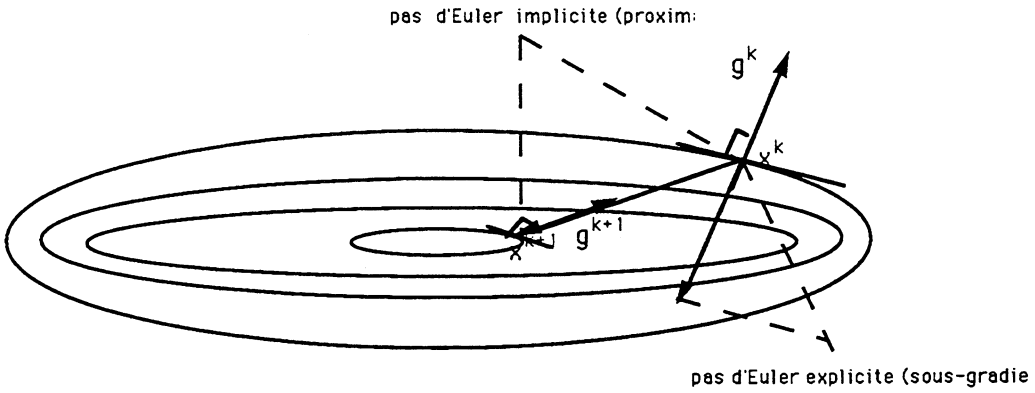
(iii) $x(0) = x_0$

De plus, la limite de x(t), quand t tend vers l'infini, existe et est un minimum de f.

Lorsqu'on discrétise ce chemin x(t), on obtient une suite de points $x^k = x(t_k)$ qui converge vers une solution du problème de minimisation de f sur X. Il se trouve que cette suite coïncide avec la suite engendrée par l' algorithme (AP2) quand x(0) est le point de départ de cette dernière. Elle s'écrit, dans le cas où f est différentiable

$$x^{k+1}=x^k - \lambda_k \nabla f(x^{k+1})$$

On peut alors visualiser, géométriquement, une itération de l'algorithme (AP2). Dans la figure ci-dessous, nous considérons le cas d'une fonction différentiable f, dont le gradient en $x^k$ est noté $g^k$.




une itération de algorithme du point proximal : Euler rétrograde

### 2-2-2 Interprétation géométrique de l'algorithme (AP2) :

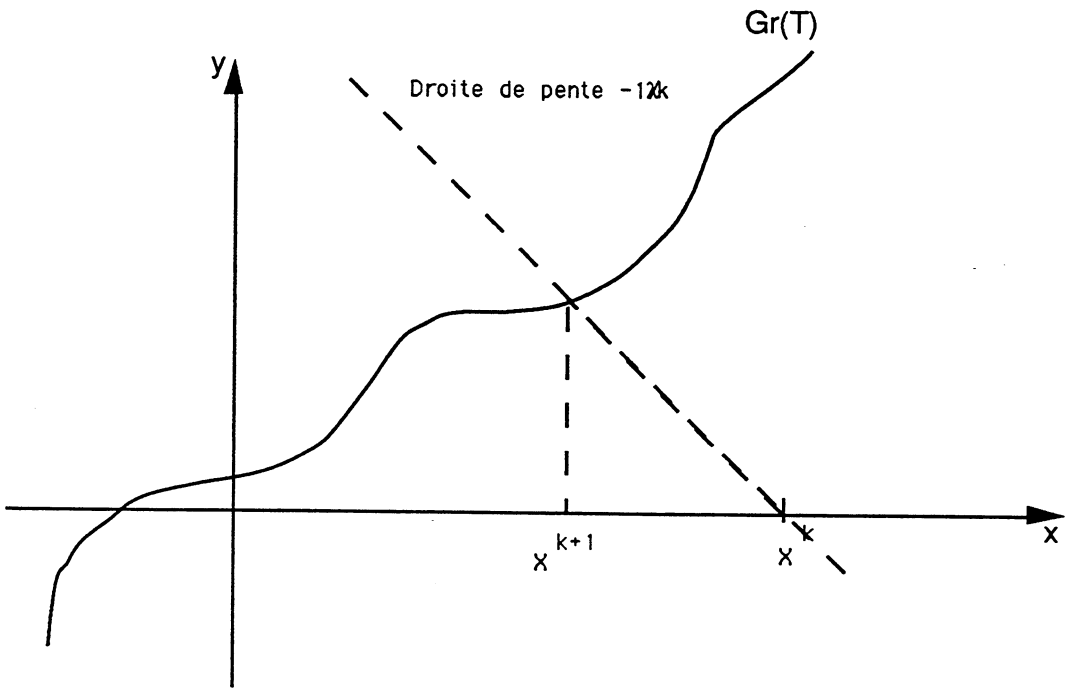
Dans ce paragraphe, nous supposons que T est un opérateur maximal monotone et surtout univoque. Dans ce cas, la relation (2-1-3) est équivalente à :

$$Tx^{k+1} = \frac{x^k - x^{k+1}}{\lambda_k}$$

On peut ainsi voir facilement que $x^{k+1}$ est le point d'intersection du graphe de T et de la droite D d'équation

$$y = \frac{x^k - x}{\lambda_k}$$

Par conséquent, on peut schématiser une itération de l'algorithme du point proximal comme suit :




Cet algorithme reste intéressant dans la mesure où l'on sait calculer le prox ( i.e $(I+\lambda T)^{-1}(x^k)$) ou bien une valeur approchée à chaque itération.

## 2-3 Convergence de l'algorithme (AP2) :

### 2-3-1 Introduction :

Dans ce paragraphe, nons allons parler de certains résultats de convergence de l'algorithme (AP2) en commençant par ceux de Rockafellar(1976a), sans doute la référence la plus importante parue en 1976. On considère le cas général des opérateurs maximaux monotones, le paramètre étant considéré non constant, et tenant compte des erreurs numériques éventuelles. Ces résultats ont été une source d'inspiration pour d'autres auteurs, notamment Spingarn(1983) pour la convergence de l'algorithme de l'inverse partiel, et Ha(1990) pour la convergence de l'algorithme dit proximal généralisé.

Notons que la convergence de l'algorithme (AP2) fut étudiée pour la première fois par Martinet(1972) dans le cas particulier des inéquations variationelles avec un paramètre $\lambda_k>0$ constant.

### 2-3-2 Résultats de convergence (R.T.Rockafellar) :

Comme nous l'avons dit plus haut, les résultats généraux dûs à Rockafellar(1976) sont d'une importance primordiale. Ces derniers tiennent compte des erreurs numériques en laissant une tolérance de calcul respectant à chaque itération un des deux critères suivants :

$$(A) \qquad \|x^{k+1} - J_{\lambda_k}(x^k)\| \le \varepsilon_k \text{ et } \varepsilon_k \to 0$$

$$(B) \qquad \|x^{k+1} - J_{\lambda_k}(x^k)\| \le \delta_k\|x^{k+1} - x^k\| \text{ et } \delta_k \to 0$$

ou plus généralement :

$$x^{k+1} = J^T_{\lambda_k}x^k + e_k \quad \text{et} \quad \|e_k\| \to 0 \qquad \textbf{(2-3-1)}$$

C'est la version perturbée de l'algorithme du point proximal. Les deux critères (A) et (B) sont garantis quand on a les deux conditions suivantes :

$$(A') \qquad \text{dist}(0,S_k(x^{k+1})) \le \frac{\varepsilon_k}{\lambda_k} \text{ et } \varepsilon_k \to 0$$

$$(B') \qquad \text{dist}(0,S_k(x^{k+1})) \le \frac{\delta_k}{\lambda_k}\|x^{k+1} - x^k\| \text{ et } \delta_k \to 0$$

où $\qquad S_k(x) = Tx + (1/\lambda_k)(x - x^k)$ .

**Proposition 2-3-1:** Rockafellar(1976)

On a l'estimation suivante :

$$\|x^{k+1} - J_{\lambda_k}(x^k)\| \le \lambda_k \text{dist}(0,S_k(x^{k+1}))$$

**Conséquence 2-3-2 :**

$$[\,(A') \Rightarrow (A)\,] \text{ et } [\,(B') \Rightarrow (B)\,]$$

Si on suppose que

$$\sum_{k=0}^{\infty} \delta_k < \infty \qquad (*)$$

$$\sum_{k=0}^{\infty} \varepsilon_k < \infty \qquad (**)$$

Ces conditions , (*) et (**), sont satisfaites par la version exacte de (AP2)

$$x^{k+1} = J_{\lambda_k} x^k,\ \lambda_k \geq c > 0$$

**Remarques 2-3-3 :**

**1-** Soit $x^*$ une solution du problème (2-1-1). Alors, la suite $\{x^k\}$ engendrée par l'algorithme (AP2) est contenue dans la boule compacte de centre $x^*$ et de rayon $||x^0-x^*||$.

**2-** Dans X (espace de Hilbert de dimension finie) les notions de faible et forte convergence coïncident.

**3 -** La convergence de l'algorithme (AP2) est Fejér-monotone :

$$\forall\ k>0 \qquad ||x^{k+1} - x^*|| \leq ||x^k - x^*||$$

ce qui renforce la stabilité de cet algorithme.

**Théorème 2-3-4:**

Soit $\{x^k\}$ une suite générée par l'algorithme (AP2) telle que (A) [ou (A')] soit vérifiée et $\lambda_k \geq c>0$ pour tout entier k. Supposons $\{x^k\}$ bornée, condition qui est assurée par exemple si (**) est vérifiée, et que (2-1-1) admet une solution. Alors :

**1-** $\{x^k\}$ converge vers un point $x^\infty$ satisfaisant (2-1-1) et

**2 -** $$\lim_{k\to\infty} \text{dist}(0,Tx^k) = 0$$

ce qui est équivalent à

$$\lim_{k\to\infty} ||Q_k(x^k)|| = \lim_{k\to\infty} ||x^{k+1}-x^k||=0$$

où

$$Q_k = \lambda_k.A^T_{\lambda_k}$$

**Preuve :**
1- (cf Rockafellar(1976a))

2- D'après la relation (2-1-2) et la définition de l'approximation de Moreau-Yosida d'un opérateur on a :

$$A^T_{\lambda_k} x_k = \frac{x_k - J^T_{\lambda_k} x_k}{\lambda_k} = \frac{x_k - x_{k+1}}{\lambda_k}$$

Donc

$$\|A^T_{\lambda_k} x_k\| \leq \frac{\|x_k - x_{k+1}\|}{\lambda_k}$$

**Remarque 2-3-5 :**
Dans le cas général (X de dimension infinie) cette convergence est forte si T est fortement monotone et $\lambda_k \geq c > 0$ ; $k \geq 0$

On peut se demander si, dans le cas général, la convergence de ce type d'algorithme peut être améliorée en une convergence forte (remarque ci-dessus). Güler(1991) a affirmé le contraire ; pour cela, il s'est servi de la liaison entre les semi-groupes et les opérateurs maximaux monotones et de certaines propriétés dont bénéficie le semi-groupe contractant S(t). Il a utilisé en particulier un résultat connu de Bruck(1975), qui montre que S(t)x converge faiblement vers un minimum de f (si $T=\partial f$ ; $f \in \Gamma^0(X)$ ). D'autre part, il s'est servi de l'exemple proposé par Baillon (1978) dans lequel il construit une fonction de $\mathfrak{L}^2$ propre convexe fermée et un point x de cl(D(f)) tel que S(t)x converge faiblement et pas fortement vers un minimum de f.

Revenons sur le coefficient $\lambda_k$ et signalons qu'il a une grande influence sur la vitesse de convergence de cet algorithme : En effet cet algorithme converge d'autant plus vite que $\lambda_k$ est plus grand. Sous certaines conditions, cette convergence devient superlinéaire, bien que l'algorithme soit du "premier ordre", quand $\lambda_k$ tend vers l'infini. L'inconvénient est que le calcul du prox devient mal conditionné. Ces remarques sont confirmées par les tests numériques que nous avons effectués (voir Chapitre V du présent document).

**Definition 2-3-6 :**
Nous dirons que $T^{-1}$ est Lipschitz continue en 0 (avec un module a) s'il existe une solution unique $z^*$ de $0 \in Tz$ (i.e. $T^{-1}(0) = \{z^*\}$), et pour un certain $\tau > 0$ on a :

$$\|z - z^*\| \leq a\|w\| \text{ pour tout } z \in T^{-1}(w) \text{ et } \|w\| \leq \tau.$$

**Théorème 2-3-6:** (Rockafellar(1976))
Soit $\{x^k\}$ une suite générée par l'algorithme (AP2) respectant le critère (B) ( ou (B') ) avec $\{\lambda_k\}$ croissante ( $\lambda_k \uparrow \lambda^\infty \leq \infty$). Supposons que $\{x^k\}$ est bornée et que $T^{-1}$ est Lipschitz continue en 0 de module a ; soit

$$\mu_k = \{a/(a^2 + c_k^2)^{1/2}\} < 1$$

alors $\{x^k\}$ converge vers un point $x^*$, solution unique de (2-1-1) ; de plus, il existe un entier $k_o$ tel que

$$\|x^{k+1} - x^*\| \leq \theta_k \|x^{k+1} - x^k\| \quad \text{pour tout } k \geq k_o ;$$

où

$$1 > \theta_k = (\mu_k + \delta_k)/(1-\delta_k) \geq 0 \quad \text{pour tout } k \geq k_o ;$$

$$\theta_k \rightarrow \mu_\infty \quad (\text{où } \mu_\infty = 0 \text{ si } \lambda^\infty = \infty)$$

**Remarques 2-2-7 :**

**1-** Sous les mêmes hypothèses on a l'équivalence suivante :

(i) $\{x^k\}$ est bornée

(ii) (2-1-1) admet au moins une solution

**2-** La condition de Lipschitz du théorème précèdent est satisfaite si T est fortement monotone de constante $\alpha>0$ (dans ce cas, $a = \alpha^{-1}$ ).

### 2-3-3 Convergence finie de l'algorithme (AP2) :

Sous certaines hypothèses, on peut obtenir une convergence de l'algorithme (AP2) en un nombre fini d'itérations, cf Rockafellar(1976a), Lefebvre et Michelot(1988), Ferris(1991). C'est l'objet du théorème suivant :

**Théorème 2-3-8:**

Soit $\{x^k\}$ une suite générée par l'algorithme (AP2). Supposons que pour tout $k\geq 0$ $\lambda_k \geq c>0$, que (A) [ou (A')] soit vérifiée et qu'il existe un point $x^*$ de X tel que

$$0 \in \text{int}(Tx^*) . \qquad \textbf{(2-3-2)}$$

Alors $x^*$ est l'unique solution de (2-1-1), de plus il existe $N>0$ tel que

$$J_{\lambda_k}^T x^k = x^* \quad \forall \ k \geq N$$

et donc

$$\|x^k - x^*\| \leq \varepsilon_k \quad \forall \ k \geq N$$

en particulier l'algorithme (AP2) dans sa forme exacte converge en un nombre fini d'itérations.

On peut trouver la démonstration de ce théorème dans Tossings(1990), et dans Ferris(1991).

**Remarque 2-3-9 :**

Cette propriété de convergence finie a été étudiée aussi par Lefebvre et Michelot (1988) dans le cas où T est le sous-différentiel d'une fonction localement polyhèdrale, cas où on n'a pas besoin de l'hypothèse (2-3-2).

Un exemple mettant en évidence la convergence finie dans le cas de la remarque ci-dessus sera donné à la fin de ce Chapitre.

### 2-3-4 Quelques observations

Dans le cas où le problème n'a pas de solution, la suite $\{x^k\}$ engendrée par l'algorithme (AP2) diverge ($||x^k||\to\infty$). Mais nous pouvons tout de même tirer intérêt de cet algorithme : en supposant que la suite $\{\lambda_k\}$ n'est pas sommable (la série correspondante est divergente). Cet algorithme permet de trouver un point v qui a la particularité d'être la projection de zéro sur cl(Im(T)), cf S.Reich(1977) ; autrement dit, nous avons :

$$\lim_{k\to\infty} \frac{x^k}{\sum_{i=1}^{k}\lambda_i} = -v$$

où v est l'élément de norme minimale dans R(T). La non sommabilité de la suite $\{\lambda_k\}$ n'étant pas suffisante pour la convergence de la suite $\{x^k\}$, nous pouvons tout de même avoir un renseignement pour la suite définie par :

$$z_n = \frac{\sum_{i=1}^{n}\lambda_i x^i}{\sum_{i=1}^{n}\lambda_i}$$

Cette suite converge si le problème (2-1-1) admet une solution. Dans ce cas, on dit que la suite $\{x_k\}$ *converge ergodiquement*, ou qu'elle converge en moyenne. Dans le cas contraire (i. e. (2-1-1) n'a pas de solution), la suite $\{||z_k||\}$ tend vers l'infini, cf H.Brezis et P.L.Lions1978a). Cette situation est résumée dans la deuxième assertion du théorème qui suit.

**Théorème 2-3-10 :**

Soit T un opérateur maximal monotone sur X, $\{x_n\}_{n\geq 0}$ une suite d'éléments de X générée par les règles (A) et (**), et $\{z_n\}_{n\geq 0}$ la suite définie ci-dessus. Si T admet un zéro et :

(a) Si $\lambda_k > c > 0$, $k\geq 0$, alors $\{x_n\}_{n\geq 0}$ converge vers $x^*\in Zer(T)$

(b) Si $\sum_{k=1}^{\infty}\lambda_k = +\infty$ alors $\{z_n\}$ converge vers $z^*\in Zer(T)$

(c) Si $\sum_{k=1}^{\infty}\lambda_k^2 = +\infty$ alors $x_n \to x^*\in Zer(T)$

(d) Si $\sum_{k=1}^{\infty}\lambda_k = +\infty$ et $T=\partial f$, $f\in\Gamma^0(X)$, alors $x_n \to x^*\in Zer(T)$

(a) est dû à Rockafellar(1976a), (b), (c), et (d) sont dûs à H.Brezis et P.L.Lions (1978a). Jusqu'ici, nous avons parlé de la version de l'algorithme du point proximal qui met en jeu l'opérateur T lui même. Les résultats de convergence que nous avons cités tiennent compte d'une perturbation "ponctuelle" qui représente l'erreur qui peut éventuellement être commise au cours du calcul de l'itéré. Dans le paragraphe suivant, nous allons présenter une version perturbée qui fait appel à une suite d'opérateurs $\{T^n\}$ au lieu de T.

## 2-4 L'algorithme du point proximal perturbé (AP3) :

### 2-4-1 Introduction :

Dans son travail, Rockafellar(1976a) a tenu compte d'une perturbation ponctuelle tolérant certaines erreurs numériques éventuelles quand on calcule le prox. Alart et Lemaire(1991) ont tenu compte aussi bien de la perturbation ponctuelle que de celle de l'opérateur T lui même (dans le cas $T=\partial f$ ). Ce dernier s'est servi pour cela de certains résultats de Attouch(1984) sur la convergence des opérateurs au sens des graphes. Ces résultats ont beaucoup d'utilité, vu que, dans la plupart des cas, le calcul du prox n'est pas facile. En d'autres termes, on peut approcher l'opérateur T par une suite d'opérateurs $\{T^k\}$ pour lesquels le calcul du prox est plus simple.

Le travail de Lemaire(1988) a été généralisé au cas d'un opérateur maximal monotone quelconque par P.Tossings(1990). L'algorithme résultant est appelé Algorithme du Point Proximal Perturbé (AP3)

### 2-4-2 Description de l'algorithme (AP3) :

Ici, l'itération du calcul du prox s'écrit :

$$x^{k+1} = J_{\lambda_k}^{T^k} x^k + e_k \quad \forall\ k \geq 0 \qquad \textbf{(2-4-1)}$$

où $\{T^k\}$ est une suite d'opérateurs maximaux monotones, qui converge vers T au sens que l'on précisera dans les résultats qui vont suivre. $\{e_k\}$ est une suite d'éléments de X destinée à tendre vers zéro en respectant certaines règles qui seront précisées. Elle représente l'erreur numérique éventuelle au moment du calcul du prox et $\{\lambda_k\}$ est une suite de réels strictement positifs.

***L'Algorithme (AP3)***

*1) **Initialisation** :*

$x_0 \in X$ , $k=0$ ;

*2) **Itération Proximale** :*

$$x^{k+1} = J_{\lambda_k}^{T^k} x^k + e_k$$

*3) **Test***

*Si* $0 \in Tx^k$ *alors fin et on a une solution ;*

***Sinon*** $k:=k+1$; *retour à 2)*

### 2-4-3 Convergence de l'algorithme (AP3) :

Les résultats de convergence obtenus sont similaires à ceux de Rockafellar(1976a) et sont résumés dans les deux théorèmes ci-dessous. Il sont dus à Tossings(1990), qui s'est inspirée des résultats de B.Lemaire(1988).

**Théorème 2-4-1 :**

Si le problème (2-1-1) admet une solution et si

(i) $0<c<\lambda_k$ , $k\geq 0$ ;

(ii) $\sum_{k>0} ||e_k|| < +\infty$

(iii) $\forall \rho \geq 0, \sum_{k>0} \lambda_k \delta_{c,\rho}(T^k,T) < +\infty$

alors la suite $\{x^k\}$ engendrée par la relation (2-4-1) converge vers une solution du problème (2-1-1) et

$$\lim_{k \to \infty} ||x^k - x^{k-1}|| = 0$$

Nous rappelons que la distance $\delta_{c,\rho}$ est définie dans Tossings(1990).

**Théorème 2-4-2 :**

Supposons que le problème (2-1-1) admette une solution et que

(i) $0<c<\lambda_k$ , $k\geq 0$ ;

(ii) la suite $\{x^k\}$ générée par (AP3) soit bornée

(iii) $\forall \rho \geq 0, \sum_{k>0} \lambda_k \delta_{c,\rho}(T^k,T) < +\infty$

(iv) $||e_k|| \leq \theta_k ||x^k - x^{k-1}||$ , $\forall k>0$ , avec $\sum_{k>0} \theta_k < +\infty$

(v) La suite $\{(T^k)^{-1}\}$ vérifie une condition de Lipschitz locale uniforme, i.e.

$\exists (a \geq 0 , \tau > 0)$ tq , $\forall k \geq 0$ ; on ait

$||\omega_1||, ||\omega_2|| < t \Rightarrow ||z_1 - z_2|| \leq a||\omega_1 - \omega_2||$

$\forall ( z_1 \in (T^k)^{-1}\omega_1 , z_2 \in (T^k)^{-1}\omega_2 ).$

*Alors* le problème (2-1-1) possède une solution unique $x^*$ et la suite $\{x^k\}$ engendrée par (AP3) converge vers $x^*$.
En plus il existe $N>0$ tel que :

$$||x^k - x^{k-1}|| \leq \eta ||x^k - x^*|| + C_k \delta_{\lambda_k, ||x^*||}(T^k,T) , \forall k \geq N$$

avec

$0 \leq \eta < 1$ et $\lim_{k \to \infty} C_k = 1$ .

Pour une application de cette version de l'algorithme du point proximal, nous citons à titre d'exemple le travail de Alart et Lemaire(1991).

## 2-5 Applications

### 2-5-1 Application à l'optimisation convexe :

Dans la suite, on suppose que f est un élément de $\Gamma^{\circ}(X)$. On veut dans un premier temps résoudre le problème suivant :

$$\text{Minimiser } f(x) \qquad x \in X \qquad \textbf{(2-5-1)}$$

la condition d'optimalité étant de trouver x* tel que:

$$0 \in \partial f(x^*)$$

qui est un cas particulier de (2-1-1) (il suffit de poser $T=\partial f$). Ainsi on peut itérer la suite donnée par (2-1-2) : c'est l'algorithme (AP2) appliqué au problème (2-5-1). Dans ce cas l'itération (2-1-2) peut être interprétée comme étant une *régularisation proximale*. Comme nous l'avons dit au début, elle a été évoquée pour la première fois par Moreau (1965) dans le cas particulier ($\lambda=1$) et a été appliquée en premier par Martinet(1972). Cette itération proximale prend la forme suivante :

$$x^{k+1}=\text{argmin}\{f(y) +(1/2\lambda_k)\|y-x^k\|^2 \ , y\in X\} \qquad \textbf{(2-5-2)}$$

Si on pose

$$f_\lambda(x)=\min\{f(y) +(1/2\lambda)\|x-y\|^2 \ , y\in X\}$$

ce minimum est atteint en un point unique qu'on notera $\text{prox}_{\lambda f}x$. Nous rappelons que $f_\lambda$ est appelée la fonction *régularisée* de f ou approximation de Moreau-Yosida de f.

**Proposition 2-5-1 :**

Soit f un élément de $\Gamma^{\circ}(X)$ et $\lambda>0$. La fonction régularisée $f_\lambda$ de f est convexe et différentiable ; son gradient est donné par

$$\nabla f_\lambda(x) = \frac{x - \text{prox}_{\lambda f}x}{\lambda}$$

De plus, on a

$$\|\nabla f_\lambda(x)\| \le \inf \{ \|y\| : y\in \partial f(x) \}$$

**Preuve** :

Montrons d'abord que $f_\lambda$ est différentiable. Pour cela, Soit $A_\lambda$ l'approximation de Yosida de $\partial f$ et $J_\lambda$ sa résolvante de paramètre $\lambda$. En réécrivant les deux opérateurs précédents et par un simple calcul on montre que pour tout x on a :

$$A_\lambda x \in \partial f(J_\lambda x) \qquad (*)$$

Notons que pour tout x et y éléments de X on a :

$$f_\lambda(x) = f(J_\lambda x) + (\lambda/2)||A_\lambda x||^2$$

d'où

$$f_\lambda(y) - f_\lambda(x) = f(J_\lambda y) - f(J_\lambda x) + (\lambda/2)\langle A_\lambda x - A_\lambda y , A_\lambda x + A_\lambda y\rangle$$

en effectuant les calculs pour le second membre nous obtenons :

$$f_\lambda(x) - f_\lambda(y) = f(J_\lambda x) - f(J_\lambda y) - \langle A_\lambda y , J_\lambda x - J_\lambda y\rangle + (\lambda/2)||A_\lambda x - A_\lambda y|| - \langle A_\lambda y , y-x\rangle$$

D'après la relation (*), ci-dessus, on peut écrire que :

$$f_\lambda(x) - f_\lambda(y) \geq \langle A_\lambda y , x - y\rangle$$

$$\geq - \langle A_\lambda x - A_\lambda y , x - y\rangle + \langle A_\lambda x , x - y\rangle$$

or $A_\lambda$ est Lipschitzien de rapport $\lambda^{-1}$ (cf prop. 1-2-12) donc

$$f_\lambda(x) - f_\lambda(y) \geq - \lambda^{-1}||x - y||^2 + \langle A_\lambda x , x - y\rangle$$

donc

$$0 \leq f_\lambda(y) - f_\lambda(x) + \langle A_\lambda x , x - y\rangle \leq \lambda^{-1}||x - y||^2$$

et on a finalement

$$| f_\lambda(y) - f_\lambda(x) + \langle A_\lambda x , x - y\rangle| \leq \lambda^{-1}||x - y||^2$$

En ce qui concerne la convexité de $f_\lambda$, on peut se référer aux résultats sur l'inf-convolution de deux fonction convexes, du moment que $f_\lambda$ est l'inf convolution de f et de la fonction $||.||^2$. D'où la première partie du résultat. La deuxième inéquation peut être montrée de façon générale lorsqu'on a un opérateur maximal monotone au lieu de $\partial f$. On notera par $(\partial f)^\circ(x)$ l'élément de $\partial fx$ de norme minimale. $\partial f$ est maximal monotone (Rockafellar (1966, 1970c)), donc pour tout x élément de X, $\partial f(x)$ est fermé, d'où :

$$(\partial f)^\circ x \in \partial f(x) \qquad (**)$$

tenant compte de la relation (*) ci-dessus, on peut écrire que

$$\langle (\partial f)^\circ x - A_\lambda x , \lambda^{-1}(x - J_\lambda x)\rangle \geq 0$$

donc

$$\langle (\partial f)^\circ x - A_\lambda x , A_\lambda x\rangle \geq 0$$

et par conséquent

$$||(\partial f)^\circ x|| \geq ||A_\lambda x||$$

**Proposition 2-5-2 :**

Soit f un élément de $\Gamma^{\circ}(X)$, $0<\lambda_1<\lambda_2$ et $\alpha=\inf f$. Alors pour tout $x \in Dom(f)$ on a

$$1- \quad \alpha \le f_{\lambda_2}(x) \le f_{\lambda_1}(x) \le f(x)$$

$$2- \quad \lim_{\lambda \to 0} f_\lambda(x) = f(x)$$

$$3- \quad \lim_{\lambda \to +\infty} f_\lambda(x) = \alpha = \inf f$$

de plus, si $x^*$ est le minimum de f(x) sur X, $x^*$ minimise $f_\lambda(x)$ sur X.

**Preuve :**

**1-** Soient $\alpha=\inf f(x)$, $\lambda_1 > \lambda_2 > 0$ et $x \in X$ fixé ;
Pour tout y élément de X on a :

$$\alpha \le f(y) + (2\lambda_1)^{-1}||y-x||^2 \le f(y) + (2\lambda_2)^{-1}||y-x||^2$$

ainsi on obtient les deux premières inégalités.
Si on pose

$$g(y) = f(y) + (2\lambda_2)^{-1}||y-x||^2$$

alors

$$g(x) = f(x) \ge \inf_y g(y)$$

d'où la troisième inégalité

**2-** D'après l'assertion 1 on a que pour tout $\lambda > 0$

$$f(x) \ge f_\lambda(x)$$

Soit y un élément du sous différentiel de f en x ;

$$f(x) - f_\lambda(x) = f(x) - f(J_\lambda x) - (\lambda/2)||A_\lambda x||^2$$

$$= f(x) - f(J_\lambda x) - \langle y , x-J_\lambda x \rangle + \lambda \langle y , A_\lambda x \rangle - (\lambda/2)||A_\lambda x||^2$$

$$\le \lambda \langle y , A_\lambda x \rangle - (\lambda/2)||A_\lambda x||^2$$

$$\le \lambda||y||.||A_\lambda x|| - (\lambda/2)||A_\lambda x||^2 = \lambda||A_\lambda x||.(||y|| - (1/2)||A_\lambda x||)$$

Tenant compte de la proposition précédente, cette quantité tend vers zéro quand $\lambda$ tend vers zéro.

**3 -** Soit $x^*$ un point optimal de f (i.e $\alpha = f(x^*)$ ) et x quelconque

$$f_\lambda(x) - f(x^*) = f(J_\lambda x) - f(x^*) + \langle A_\lambda x , x^* - J_\lambda x \rangle$$

$$- \langle A_\lambda x , x^* - J_\lambda x \rangle + (\lambda/2)\|A_\lambda x\|^2$$

$$\leq \langle A_\lambda x^* - A_\lambda x , x^* - x \rangle - \langle A_\lambda x , x - J_\lambda x \rangle + (\lambda/2)\|A_\lambda x\|^2$$

$$\leq (1/\lambda)\|x^* - x\|^2 - (\lambda/2)\|A_\lambda x\|^2$$

$$\leq (1/\lambda)\|x^* - x\|^2$$

d'où le résultat cherché . ∎

**Exemples :**

**1-** Pour la fonction f définie par

$$f(x) = \|x\|$$

son approximation de Moreau-Yosida $f_\lambda$ est définie pour $\lambda>0$ par

$$f_\lambda(x) = \begin{cases} \dfrac{x^2}{2\lambda} & ; \text{si } |x| \leq \lambda \\ |x| - \dfrac{\lambda}{2} & ; \text{ si } |x| \geq \lambda \end{cases}$$

Ci-dessous la figure 1 représente les courbes de $f_\lambda$ pour certaines valeurs du paramètre $\lambda>0$.

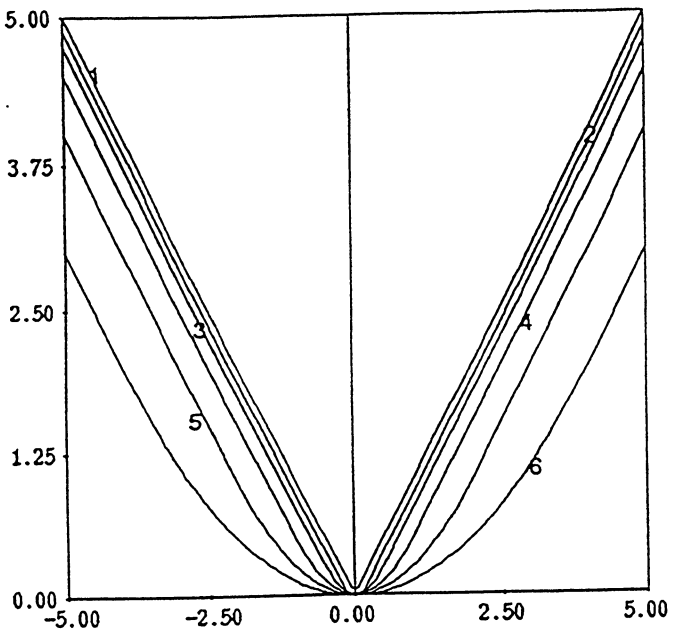
**2-** Pour la fonction f définie par

$$f(x) = x^2$$

son approximation de Moreau-Yosida $f_\lambda$ est définie pour $\lambda>0$ par

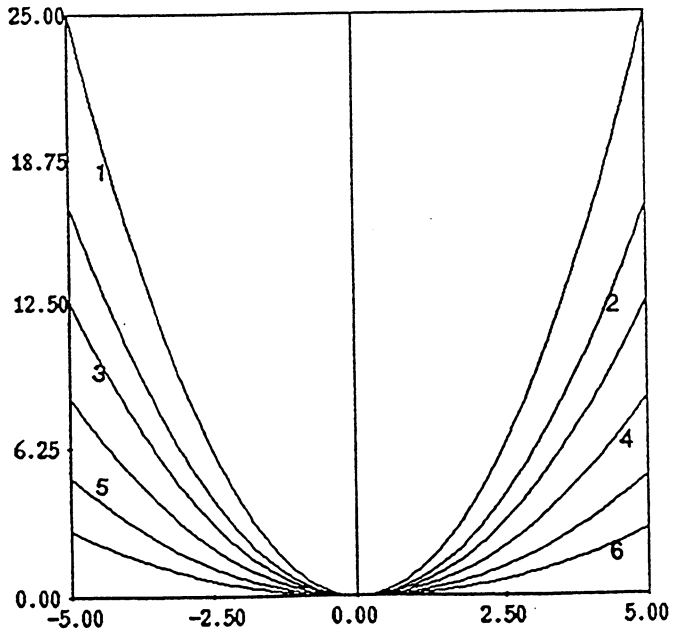
$$f_\lambda(x) = \frac{x^2}{1+2\lambda}$$

Ci-dessous la figure 2 représente les courbes de $f_\lambda$ pour certaines valeurs du paramètre $\lambda>0$.




1- $f(x)=|x|$
2- $f_\lambda(x)$ $\lambda=0.25$
3- $f_\lambda(x)$ $\lambda=0.5$
4- $f_\lambda(x)$ $\lambda=1$
5- $f_\lambda(x)$ $\lambda=2$
6- $f_\lambda(x)$ $\lambda=4$

*Figure 1*




1- $f(x)=x^2$
2- $f_\lambda(x)$ $\lambda=0.25$
3- $f_\lambda(x)$ $\lambda=0.5$
4- $f_\lambda(x)$ $\lambda=1$
5- $f_\lambda(x)$ $\lambda=2$
6- $f_\lambda(x)$ $\lambda=4$

*Figure 2*

Au cours des démonstrations des deux dernières propositions, nous avons pu remarquer que la suite $\{f(x^k)\}$ est décroissante, où $\{x^k\}$ est une suite générée par l'algorithme (AP2) et $T=\partial f$. Lorsque la suite des itérés $\{x^k\}$ converge, $\{f(x^k)\}$ converge vers la valeur optimale $f^*$ de f. Un des intérêts de l'algorithme du point proximal réside dans le fait que, même si la suite $\{x^k\}$ ne converge pas ( le problème n'admet pas de solution ), on peut tout de même tirer des informations : la suite $\{f(x^k)\}$ converge vers la valeur optimale $f^*$ de f, Guler(1991).

**Proposition 2-5-3 :**

$x^0$ étant donné, si $\{x^k\}$ est une suite générée par la relation (2-5-1) i.e par l'algorithme (AP2), avec $\lambda_k>\lambda>0$, alors $\{x^k\}$ est une suite minimisante de f c'est à dire :

$$\lim_k f(x^k) = f^* = \inf f(x)$$

**Preuve :**

Nous savons que $(x^k - x^{k+1})/\lambda_k \in \partial f(x^{k+1})$, donc pour tout $x\in X$, on a :

$$2\lambda_k f(x^{k+1}) \le 2\lambda_k f(x) - 2\langle x^k - x^{k+1}, x - x^{k+1}\rangle$$

$$\le 2\lambda_k f(x) - \{ \|x^k - x^{k+1}\|^2 + \|x - x^{k+1}\|^2 - \|x - x^k\|^2 \}$$

par conséquent, on a :

$$\lambda_k f(x^k) \le \lambda_k f(x) + (1/2)\{\|x - x^k\|^2 - \|x^{k+1} - x^k\|^2 - \|x - x^k\|^2 \}$$

et donc

$$\sum_{k=0}^{n} \lambda_k f(x^{k+1}) \le \sum_{k=0}^{n} \lambda_k f(x) + \frac{1}{2}\left\{ \|x^0 - x\| - \sum_{k=0}^{n} \|x^{k+1} - x^k\|^2 - \|x - x^{k+1}\| \right\}$$

Notons $t_n$ la somme suivante :

$$t_n = \sum_{k=0}^{n} \lambda_k$$

alors

$$t_n.f(x^{n+1}) \le t_n.f(x) + (1/2).\|x^0 - x\|^2$$

par conséquent nous avons :

$$\inf f(x) \le f(x^{n+1}) \le f(x) + (1/2t_n)\|x^0 - x\|^2 \quad ; \ \forall x\in X$$

et en faisant tendre n vers l'infini nous obtenons le résultat cherché. ∎

### 2-5-2 Application au Lagrangien augmenté

On considère le problème suivant :

$$(\mathbf{E2}) \qquad \begin{cases} \mathbf{min}\ f(x) \\ g_j(x) = 0\ ;\ 1 \le j \le m \end{cases}$$

Le problème dual associé est :

$$(\mathbf{E3}) \qquad \max\ G(u_1, ..., u_m)$$
$$u = (u_1, ..., u_m)$$

où

$$G(u_1, ..., u_n) = \inf_x \left\{ f(x) + \sum_{j=1}^{m} u_j.g_j(x) \right\} \qquad \textbf{(L0)}$$

pour u tel que le inf dans (L0) existe.

On se propose d'appliquer la méthode proximale au problème (E3). Pour cela diminuons le terme à minimiser dans l'expression (L0) d'un terme quadratique correspondant, prenant en compte la variable duale u seulement. Le but est analogue : remplacer la fonction G, concave par une fonction fortement concave et différentiable. On retrouve alors la méthode dite des multiplicateurs initialement proposée par Hestenes(1969) et Powell(1969) (avant que le lien n'ait été établi avec l'algorithme proximal).

Cet algorithme est exprimé en terme d'une fonction *Lagrangien augmenté* qui dépend d'un paramètre positif $\lambda$.

A l'itération k, on calcule $u^k$ à partir du problème suivant :

$$\max \left\{ G(u) - \frac{1}{2\lambda} ||u-u^{k-1}||^2 \right\}$$

Tenant compte de (L0), on obtient le problème suivant :

$$\max\ \min_x \left\{ f(x) + \sum_{j=1}^{m} u_j.g_j(x) \right\} - \frac{1}{2\lambda} ||u-u^{k-1}||^2$$

Soit x(u) la solution du sous-problème en x ci-dessus. L'itéré $u^k$ peut être calculé explicitement :

$$u_i^k = u_i^{k-1} + \lambda g_j(x(u^{k-1}))\ ,\ 1 \le i \le m$$

En remplaçant dans le sous-problème on obtient le problème suivant :

$$\inf_x \left\{ f(x) + \sum_{j=1}^{m} u_j^{k-1}.g_j(x) + \frac{\lambda}{2} \left|\left| \sum_{j=1}^{m} g_j(x) \right|\right|^2 \right\}$$

Par conséquent, les sous-problèmes que nous obtenons à chaque itération sont en x seulement. Pour le cas des contraintes d'inégalités, il suffit d'ajouter des variables d'écart et on retrouve le cas du problème (E2). Par un calcul similaire à celui de

Rockafellar(1976a), le lagrangien augmenté peut être exprimé comme suit :

$$L(x,u,\lambda) = f(x) + \sum_{j=1}^{m} \Gamma(g_i(x),y_i,\lambda) \qquad \textbf{(L1)}$$

où

$$\Gamma(g_i(x),y_i,\lambda) = y_i g_i(x) + \frac{\lambda}{2} g_i(x)^2 \quad \text{si } g_i(x) \geq -\frac{y_i}{\lambda}$$

$$= -\frac{1}{2\lambda} y_i^2 \quad \text{si } g_i(x) < -\frac{y_i}{\lambda}$$

La manipulation primal-duale consiste à son tour à faire une régularisation proximale primale et une autre duale. Elle bénéficie des avantages de chaque méthode. Le nouveau lagrangien est :

$$L_{PD}(x,u,\lambda) = f(x) + \frac{1}{2\lambda} ||x-x^k||^2 + \sum_{j=1}^{m} \Gamma(g_i(x),y_i,\lambda) \qquad \textbf{(L2)}$$

La méthode obtenue est appelée *Méthode Proximale des multiplicateurs*.. Pour plus de détails, nous renvoyons le lecteur au travail de Rockafellar(1976b).

### 2-5-3 Remarques :

Dans ce paragraphe, nous allons résumer une interprétation de l'algorithme du point proximal due à B.Lemaire(1991).

Soient $f \in \Gamma^0(X)$, $\alpha$ et $\lambda$ deux réels positifs et $x \in X$. On suppose que f admette une valeur optimale $f^*$ et

$$\alpha < f^* = \inf f \text{ et } \lambda(\alpha) = f(y) - \alpha$$

Soit y défini par :

$$y = \text{prox}_{\lambda f}(x)$$

alors on a dans $X \times \mathbb{R}$

$$(y,f(y)) = \text{argmin}_{(u,r)\in epi(f)}\{ ||(u,r)-(x,\alpha)||^2_{X\times\mathbb{R}}\} = \text{proj}_{epi(f)}(x,\alpha)$$

En effet, il existe un hyperplan support de epi(f) en (y,f(y)) orthogonal au vecteur $(x-y,\alpha-f(y))$.

Donc le le vecteur $(1/f(y)-\alpha)(x-y)$ est un sous-gradient de f en y, i.e. :

$$0 \in y-x+(f(y)-\alpha)\partial f(y)$$

ce qui veut dire que

$$y = \text{prox}_{\lambda f}(x), \text{ avec } \lambda = \lambda(\alpha).$$

Dans la figure suivante, nous visualisons le calcul du prox d'une part et l'effet du paramètre $\lambda$, qui est une fonction décroissante de $\alpha$ , sur la convergence d'autre part : plus $\lambda$ est grand, plus vite on va vers la solution du problème.

Il y a des cas où il est plus facile de projeter sur le graphe de f que de calculer le prox autrement, notamment, si f est polyhèdrale, par exemple.

$$f(x) = \max g_i(x) \; ; \; 1 \leq i \leq n$$

où les fonctions $g_i$ ($1 \leq i \leq n$) sont linéaires.

Nous signalons tout de même qu'il y a une légère différence entre cette interprétation et l'algorithme du point proximal. Cette différence réside dans le paramètre $\lambda$ qui dépend, comme sa forme l'indique, de la valeur de la fonction au point suivant. On a ainsi une anticipation aussi bien sur la direction que sur le paramètre.

La figure suivante montre l'effet de $\alpha$ sur la convergence de l'algorithme. En effet, pour deux valeurs $\alpha > \beta$ du paramètre, nous construisons le point proximal de $\lambda(\alpha)f$ (resp. de $\lambda(\beta)f$) en z, $y(\alpha)$ (resp.$y(\beta)$). On constate que $y(\beta)$ est plus proche de la solution que $y(\alpha)$.

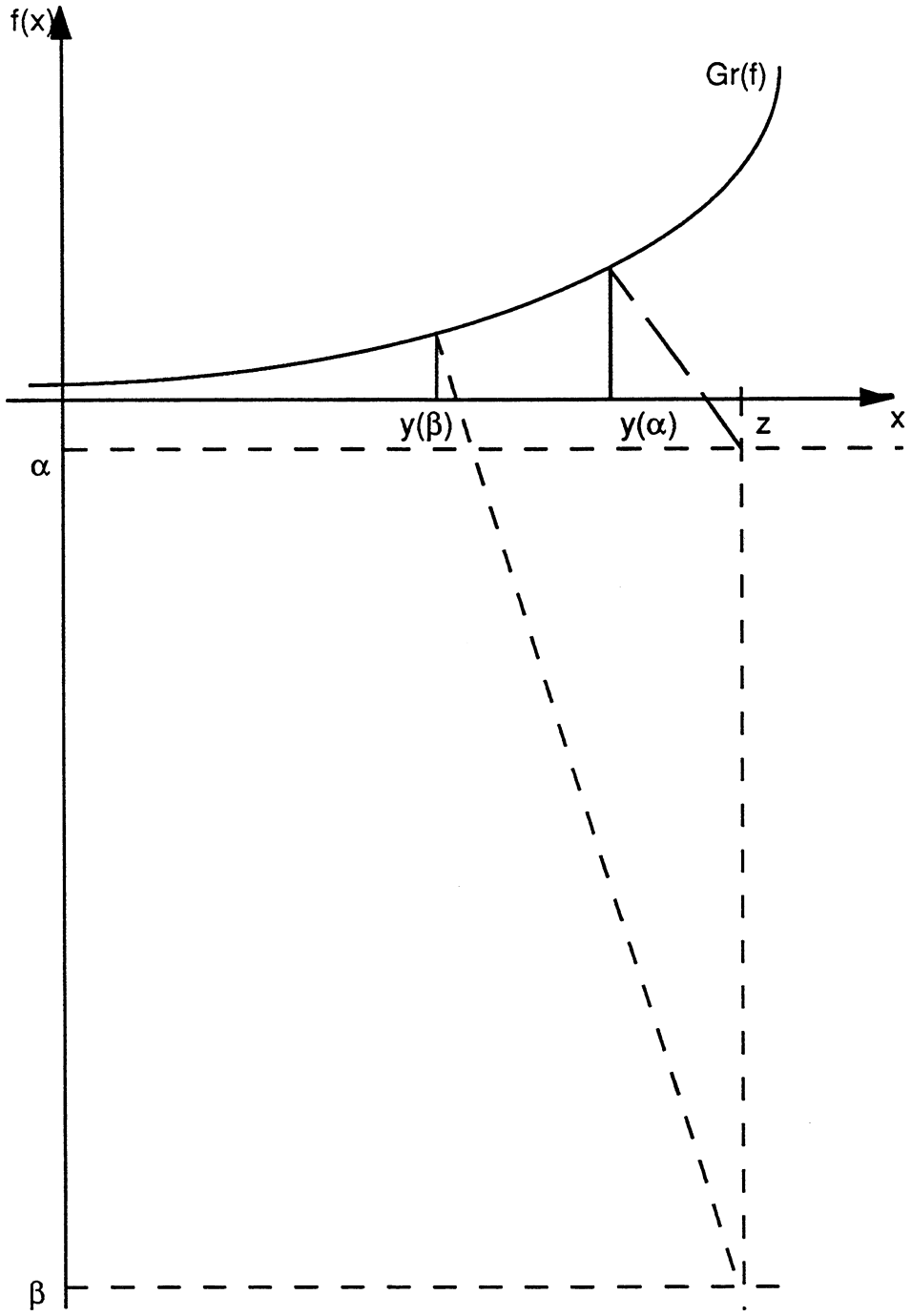



Figure mettant en évidence l'effet de α sur la convergence de l'algorithme de Lemaire

Nous donnons sur la figure suivante un exemple où la convergence de (AP2) est finie : le cas où f est polyhèdrale.

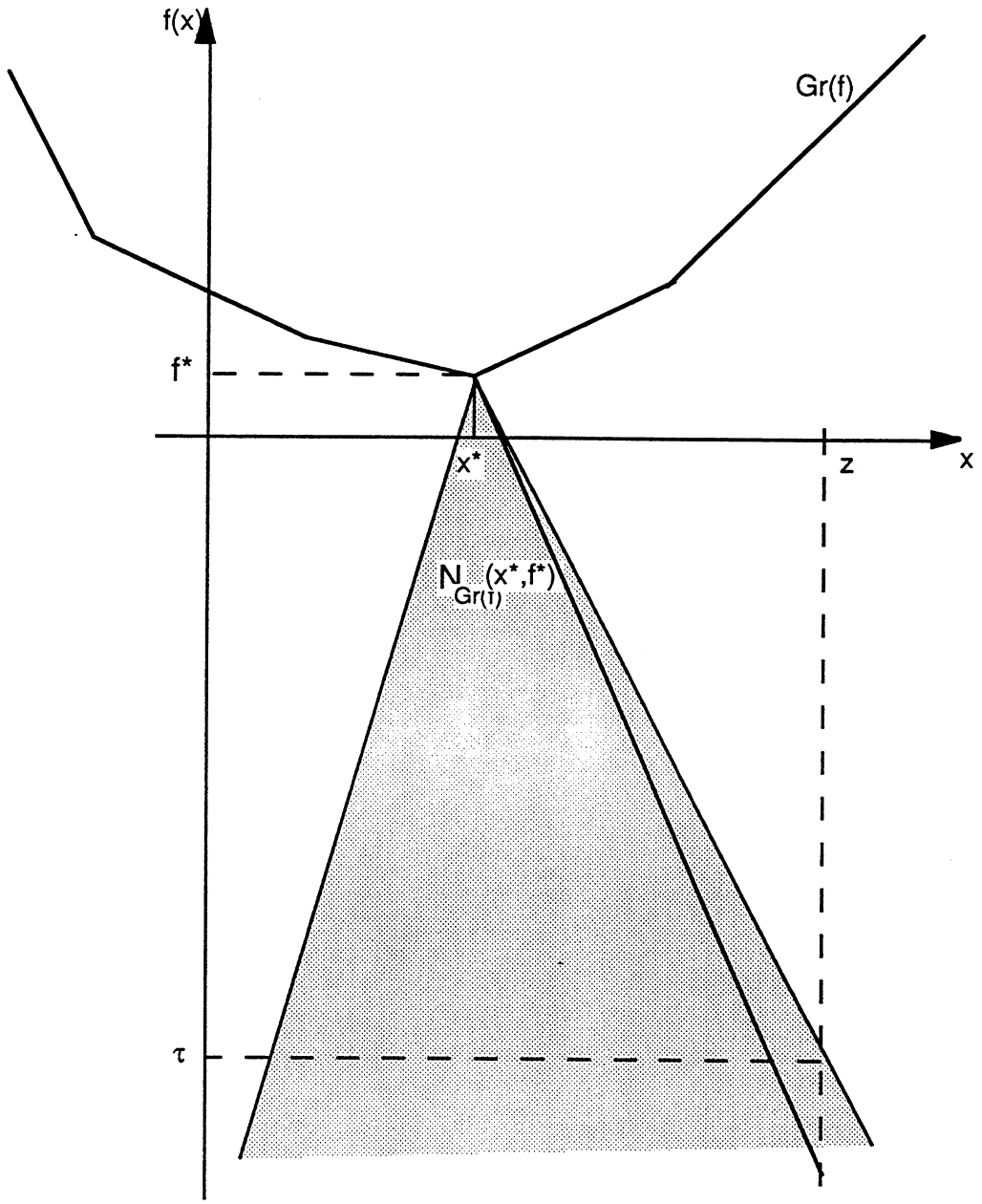



Figure mettant en évidence la convergence finie de l'algorithme (AP2)

Nous remarquons que si $\alpha \leq \tau$, alors le point $(z,\alpha)$ est dans le cône normal de epi(f) au point $x^*$

$$(z,\alpha) \in N_{Gr(f)}(x^*,f^*)$$

Par conséquent on a

$$Prox_{\lambda(\alpha)f} z = x^*$$

## 2-6 L'algorithme de Tikhonov

### 2-6-1 Description de l'algorithme

Nous avons évoqué au début de ce texte que des versions particulières de l'algorithme (AP2) ont pu être traitées de façon indirecte. Parmi ces travaux, on trouve celui de Tikhonov et Arsenine(1976) ; il traite le cas particulier des fonctions de $\Gamma^0(X)$.

Cet algorithme que nous noterons (AT) consiste à trouver un zéro d'un opérateur maximal monotone (résoudre l'inclusion (2-1-1) ). La différence qu'il a avec l'algorithme AP2 est qu' il génère une suite par la relation :

$$x^{k+1} = J_{\lambda_k}^T x^0 \qquad \textbf{(2-6-1)}$$

au lieu de la générer, comme l'algorithme AP2, par la relation (2-1-3), où $\lambda_k$ est une suite de réels positifs qui tend vers l'infini. $x^0$ est en général pris égal à zéro ; c'est l'algorithme dit de Tikhonov

**L'Algorithme de Tikhonov (AT)**

*1) Initialisation*

$x_0 \in X$ , $k=0$ ;

*2) Itération Proximale*

$$x^{k+1} = J_{\lambda_k}^T x^0$$

*3) Test*

*Si* $0 \in Tx^k$ alors fin et on a une solution ;
**Sinon** k:=k+1; retour à 2)

### 2-6-2 Convergence de l'algorithme (AT)

Nous rappelons ci-dessous certains résultats de convergence de l'algorithme AT (cf P.Tossings(1990)). Ces résultats ont été inspirés de Tikhonov et Arsenine(1976).

**Théorème 2-6-1 :**

Supposons que (2-1-1) admette une solution, que $0 < \lambda_k$ , $k \geq 0$ et que $\lim_k \lambda_k = +\infty$, alors la suite engendrée par l'algorithme (AT) converge vers la solution du problème (2-1-1) qui est de norme minimale.

Nous rappelons que sous les mêmes hypothèses sur $\{\lambda_k\}$ que le théorème précédent, le problème (2-1-1) admet une solution si et seulement si la suite $\{x_n\}$ générée par (AT) est bornée.

**Remarque 2-6-2 :**

Nous avons vu (§2-3-4), que quand on considère l'algorithme (AP2) la convergence est ergodique dans le cas où il n'y a pas de solution. On peut se demander si on peut avoir une telle convergence avec l'algorithme (AT).

## 2-7 Algorithme de Tikhonov perturbé (ATP)

### 2-7-1 Description de l'algorithme

On considère ici la version approchée de l'algorithme (AT), c.a.d. sa version implémentable.

Comme pour l'algorithme du point proximal, on doit prendre en compte d'éventuelles erreurs en introduisant une perturbation. Cette perturbation peut être ponctuelle comme dans le travail de Rockafellar(1976a). Elle peut être plus générale aussi. On tient compte alors des erreurs de calcul et de la perturbation de l'opérateur. Cette perturbation a l'intérêt de faciliter l'implémentation du prox dans certains cas.

**L'Algorithme de Tikhonov Perturbé (ATP)**

*1) **Initialisation***

$x_0 \in X$ , $k=0$ ;

*2) **Itération Proximale***

$$x^{k+1} = J_{\lambda_k}^{T^k} x^0 + e_k$$

*3) **Test***

***Si*** $0 \in Tx^k$ *alors fin et on a une solution ;*

***Sinon*** $k:=k+1$ *et retour à 2)*

où $\{\lambda_k\}$ est une suite de réels positifs destinée à tendre vers l'infini ; $\{e_k\}$ est une suite destinée à tendre vers zéro ; elle représente la perturbation ponctuelle tenant compte des erreurs de calcul éventuelles. $\{T^k\}$ est une suite d'opérateurs maximaux monotone qui approche T au sens des graphes.

### 2-7-2 Convergence

**Théorème 2-7-1 :**

Supposons que :

i- $0 < \lambda_n$ ; $n \geq 0$ et $\lim_n \lambda_n = +\infty$

ii- $T^n \xrightarrow{G} T$

iii- La suite engendrée par (ATP) est bornée ;

iv- $\lim_n \|e_n\| = 0$

alors , toute valeur d'adhérence de la suite $\{x^n\}$ engendrée par (ATP) est une solution de (3-1).

Pour la démonstration de ce théorème, nous faisons référence à la thèse de Tossings(1990).

Comme application de l'algorithme (AT), nous proposons au lecteur de se référer au paragraphe précédent portant sur l'application de l'algorithme (APP) à l'optimisation convexe, avec des petits changements ( on remplace dans les itérations $x^k$ par $x^0$ ).

## 2-8 Algorithme de Gol'shtein

### 2-8-1 Introduction

Dans ce paragraphe nous allons retrouver la version de l'algorithme proximale due à Gol'shtein et Tret'yakov(1979), lequel consiste aussi à trouver un zéro d'un opérateur maximal monotone. Pour cela, on procède ainsi :

$$0 \in Tx \Leftrightarrow x = (I+\lambda T)^{-1}x$$

$$\Leftrightarrow x = (1-\rho)x + \rho(I+\lambda T)^{-1}x$$

$$\Leftrightarrow x = x - \rho[I - (I+\lambda T)^{-1}]x$$

$$\Leftrightarrow x = (I - \rho\lambda\ A_\lambda^T)x$$

Il est naturel de penser à l'itération de point fixe suivante :

$$x^{k+1} = (I - \rho_k\lambda_k A_{\lambda_k}^T)x^k + \varepsilon_k = (1-\rho_k)x^k + \rho_k J_{\lambda_k}^T x^k + \varepsilon_k \qquad \textbf{(2-8-1)}$$

On obtient donc l'algorithme dit de Gol'shtein.

### 2-8-2 Convergence

Cet algorithme converge sous certaines conditions qui sont celles du théorème suivant (voir Bertsekas et Tsitsiklis(1989)):

**Théorème 2-8-1 :**

Soit T un opérateur maximal monotone sur X et $\{z^k\}$ la suite définie par

$$z^{k+1} = (1-\rho_k)z^k + \rho_k w^k \qquad k\geq 0$$

où

$$\|w^k - J_{\lambda_k}^T z^k\| \leq \varepsilon_k$$

On suppose que les suites $\{\varepsilon^k\}$, $\{\rho^k\}$, $\{\lambda^k\}$ sont telles que :

$$\sum_0^\infty \varepsilon_k < \infty$$

$$\text{Inf } \rho_k > 0 \ ; \ k\geq 0.$$

$$\sup \rho_k < 2 \ ; \ k\geq 0$$

$\text{Inf}\ \lambda_k = \bar{c} > 0\ ;\ k \geq 0..$

(Une telle suite est conforme à l'algorithme du point proximal). Alors,
Si $Zer(T) \neq \varnothing$, $\{z^k\}$ converge vers un zéro de T.
Et si $Zer(T) = \varnothing$, $\{z^k\}$ est une suite non bornée.

Pour étudier la qualité d'une convergence éventuelle de cette suite, nous sommes amenés à étudier l'opérateur

$$H^T_{\lambda,\rho} = (I - \rho\lambda A^T_\lambda)$$

Soient x et y deux éléments de X alors

$$||H^T_{\lambda,\rho}x - H^T_{\lambda,\rho}y||^2 = ||x-y||^2 + (\lambda\rho)^2||A^T_\lambda x - A^T_\lambda y||^2 - 2\lambda\rho\langle x - y, A^T_\lambda x - A^T_\lambda y\rangle$$

or $(I+\lambda T)^{-1}$ est fermement non expansif (Thm 1-2-10) et, si on suppose que $\rho$ est positif, nous pouvons écrire :

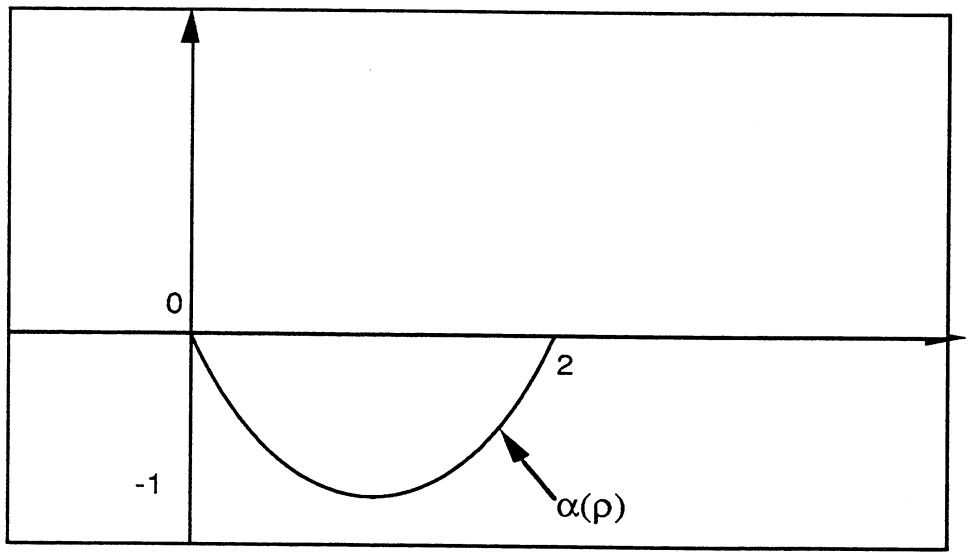
$$\begin{aligned} ||H^T_{\lambda,\rho}x - H^T_{\lambda,\rho}y||^2 &\leq ||x-y||^2 + (\lambda\rho)^2||A^T_\lambda x - A^T_\lambda y||^2 - 2\lambda^2\rho||A^T_\lambda x - A^T_\lambda y||^2 \\ &\leq ||x-y||^2 + \lambda^2(\rho^2-2\rho)||A^T_\lambda x - A^T_\lambda y||^2 \end{aligned} \quad \textbf{(2-8-2)}$$

Posons

$$\alpha(\rho) = \rho^2 - 2\rho$$

cette fonction est positive si $\rho > 2$. Par contre elle est négative si $0 < \rho < 2$, et dans ce cas, d'après (3-1-2) l'opérateur $H^T_{\lambda,\rho}$ a une propriété meilleure que la faible contraction.

Ci-dessous l'allure de la fonction $\alpha$.



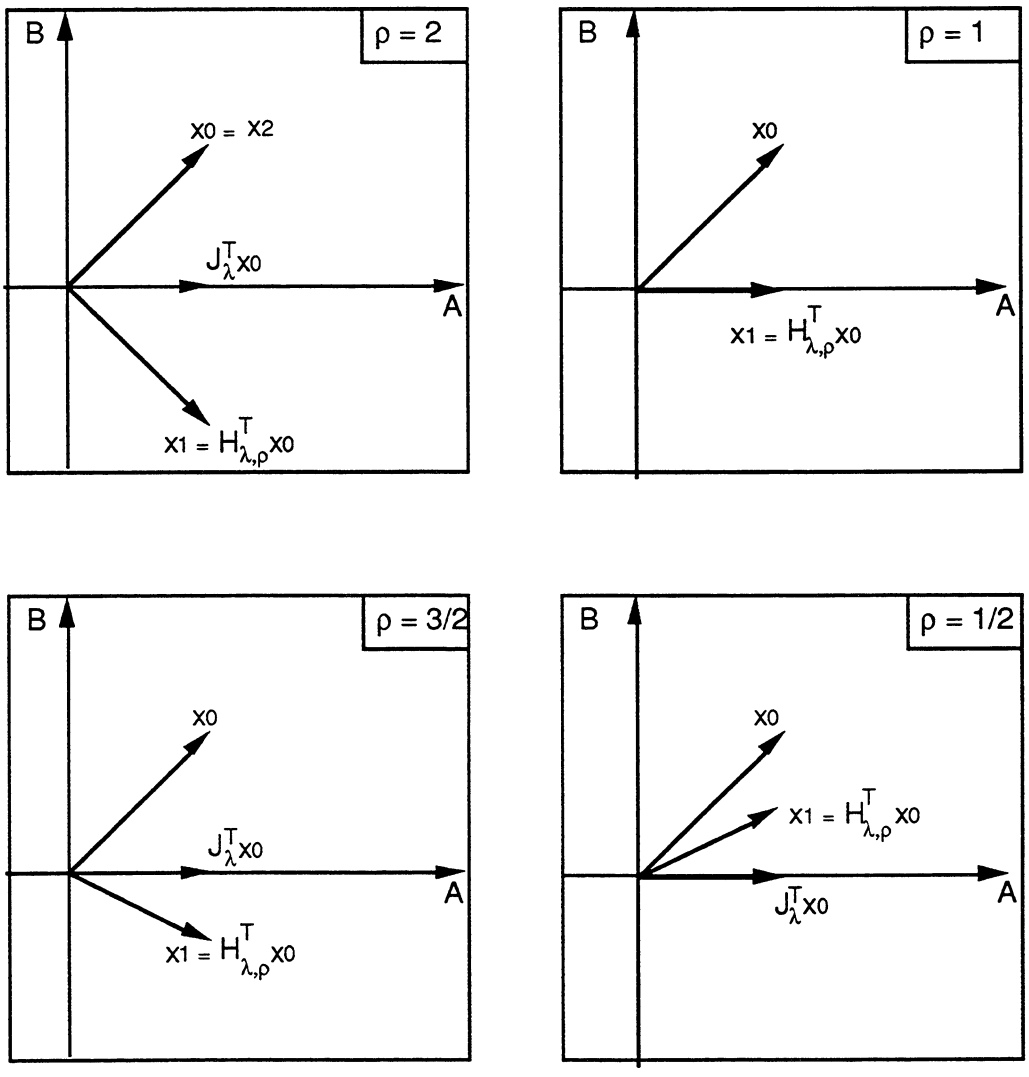

Cette fonction est strictement négative sur l'intervalle (0, 2).Elle atteint son minimum en $\rho=1$ avec une valeur égale à -1.

**Remarques 2-3-2 :**

**1-** Dans le cas où $\rho=1$, l'inégalité (2-8-2) veut dire que $H^T_{\lambda,\rho}$ est un opérateur fermement non expansif. D'autre part, on s'aperçoit aisément que $H^T_{\lambda,1}$ est la résolvante $J^T_\lambda$ de T. Ceci veut dire que $H^T_{\lambda,\rho}$ a au moins dans le cas, $\lambda=1$, les propriétés de contraction de $J^T_\lambda$.

**2-** Grâce à **1-**, une suite générée par la relation (2-8-1) convergera dans le meilleur des cas à la même vitesse qu'une suite générée par l'algorithme (AP2), toutes les deux partant d'un même point.

Ci-dessous un exemple par lequel nous illustrons ces dernières remarques.
Nous prenons $T=\partial\chi_A$ et $B = A^\perp$.



Si on considère une suite de points générée par la relation (2-8-1), elle converge vers un zéro de T s'il en a un. Nous pouvons le montrer de deux façons différentes; la première consiste à exploiter la relation (2-8-2) ci-dessus. La seconde qui est en faite la plus simple consiste à utiliser un théorème de Browder et Petryshyn(1966) en prenant comme opérateur, l'opérateur non expansif associé à T ; i.e. selon les notations du Chapitre 1, l'opérateur en question est $\beta(T) \in \mathcal{N}exp(X)$. Nous donnerons les grandes lignes de cette démonstration.

Nous rappelons que

$$U = \beta(T) = 2\, J_\lambda^T - I$$

L'opérateur U est non expansif, donc si on pose

$$U(a) = (1-a)I + a\beta(T) = (1-2a)I + 2aJ_\lambda^T \,,\ a \in (0,1)$$

$$= (1-\rho)I + \rho J_\lambda^T = B(\rho) \,,\ \rho \in (0,2)$$

D'après Browder et Petryshyn(1967), nous savons que si U admet un point fixe, ce qui est équivalent au fait que T admet un zéro, alors la suite générée par la relation suivante, converge vers un point fixe de U (zéro de T) :

$$x^{k+1} = U(a)x^k \qquad a \in (0,1)$$

ou bien

$$x^{k+1} = B(\rho)x^k \qquad \rho \in (0,2)$$

Ceci prouve la convergence de l'algorithme (AG). ∎

# Chapitre III

# l'algorithme proximal pour les problèmes avec contraintes :

## L'algorithme de décomposition proximale

## 3-1 Introduction

L'espace de travail étant toujours $X = \mathbb{R}^n$ et le problème que l'on se propose de résoudre est :

$$\text{Trouver } x \in X \text{ tel que } 0 \in L(x). \qquad \textbf{(3-1-1)}$$

où

$$L = S + T$$

S et T sont deux opérateurs maximaux monotones. Dans le cas où

$$L = \partial \chi_A + T$$

avec A un sous-espace vectoriel de X et $B = A^{\perp}$.

Le problème (3-1-1) est équivalent à

$$\text{Trouver } x \in A \text{ et } y \in B \text{ t.q } y \in Tx. \qquad \textbf{(3-1-2)}$$

Une des situations où on retrouve ce type de problème est le problème de minimisation d'une fonction convexe f ($f \in \Gamma^{\circ}(X)$) sur un sous-espace vectoriel, situation à laquelle on se ramène fréquemment, après certaines manipulations préliminaires, lorsqu'on se sert de certaines méthodes de décomposition pour les problèmes d'optimisation convexes séparables. Ces méthodes ont récemment pu prendre une place d'interêt dans beaucoup d'applications pratiques, grâce à leur implémentation sur des machines parallèles. Notamment, les problèmes d'optimisation dans les réseaux ou la programmation stochastique (D.P.Bertsekas & Tsitsiklis(1989). En général, on a différentes façons de transformer un programme convexe en un problème de la forme (3-1-2), cf Mahey et al(1992a). Mais l'idée générale est de représenter le couplage entre les sous-systèmes par un sous-espace vectoriel de l'espace produit des copies des variables primales et des variables duales. Les problèmes de localisation avec des coûts non linéaires et les problèmes de flot non linéaire sur les réseaux en sont des applications que nous allons traiter dans ce travail (voir Chapitre 5).

Nous avons vu au Chapitre 1 quelques conditions sous lesquelles l'opérateur L est maximal monotone. Lorsqu'il est maximal monotone, une idée immédiate est d'appliquer l'algorithme (AP2) pour trouver un zéro de cet opérateur ; cependant, le calcul de la résolvante de L en un point est généralement très difficile à exprimer en fonction de celles de S et T.

Dans le paragraphe qui va suivre, nous allons présenter certains problèmes classiques dont la résolution revient à celle d'un problème de type (3-1-2).

## 3-2 Exemple de motivation

**Intersection de convexes**

Soient $C_1$, ..., $C_n$, des sous-ensembles convexes fermés de X. Le problème est de trouver un point d'intersection de ces ensembles, ce qu'on peut formuler comme suit :

**($P_2$)** Trouver $x \in C_1 \cap ... \cap C_n$,

Ce problème est équivalent au suivant :

**($P'_2$)** $\min (\chi_{C_1}+...+\chi_{C_n})(x)$ ; $x \in X$

Pour avoir la propriété de séparabilité, que nous définissons au Chapitre 4, on fait des copies de la variable x et on aura à résoudre le problème séparable ci-dessous sur l'espace produit $X^n$ :

**($P''_2$)** $\min \{\chi_{C_1}(x_1)+...+\chi_{C_n}(x_n)\}$ ; $(x_1,...,x_n) \in A$

où A est le sous espace de $X^n$ défini par :

$$A = \{(x_1,...,x_n) \mid x_1=...=x_n\}$$

Ainsi on a un problème de minimisation d'une fonction convexe sur un sous-espace vectoriel, problème qu'on peut facilement traduire en un problème du type (3-1-2).

Une des méthodes les plus connues pour la résolution de ce type de problèmes est celle des projections successives. Cette méthode a été proposée en premier par Kaczmarz(1937), pour le cas particulier où les $C_i$ sont des variétés linéaires, et a été redécouverte par Von Neumann(1950) et d'autres auteurs. L'extension au cas général des convexes fermés a été proposée par Bregman(1965). Pour avoir une idée sur la généralisation au cas où les projections sont remplacées par des résolvantes d'opérateurs maximaux monotones, nous faisons référence à Tseng(1992).

La convergence de la méthode des projections successives peut être déduite du théorème 1-2-21.

## 3-3 Algorithme proximal projeté et variantes

### 3-3-1 Présentation

Pour ne pas alourdir les notations, il nous arrivera de noter l'approximation de Yosida, de paramètre $\lambda_k$, d'un opérateur T par $T_k$ et, occasionellement, la résolvante d'un opérateur T par $J^T$, lorsque nous jugerons qu'il n'y a pas d'ambiguité.

Une des façons de régulariser le problème (3-1-1) est de le remplacer par un autre dans lequel l'approximation de Moreau-Yosida joue le rôle de l'un des deux opérateurs. On obtient le problème suivant :

$$\text{Trouver } x_\lambda \in X \text{ tel que } 0 \in L_\lambda(x_\lambda). \qquad \textbf{(3-3-1)}$$

où

$$L_\lambda = A^S_\lambda + T \; ; \; \lambda > 0$$

Par un simple calcul, on peut voir que :

$$0 \in L_\lambda(x_\lambda) \Leftrightarrow (I+\lambda S)^{-1} x_\lambda \in x_\lambda + \lambda T x_\lambda$$

$$\Leftrightarrow x_\lambda = J^T_\lambda \circ J^S_\lambda x_\lambda \qquad \textbf{(3-3-2)}$$

Il est donc naturel de penser à itérer une suite de points $\{x^k\}$ générée par la règle

$$x^{k+1} = J^T_{\lambda_k} \circ J^S_{\lambda_k} x^k \qquad \textbf{(3-3-3)}$$

d'où l'algorithme que nous appelons l'algorithme proximal projeté. En effet, lorsque $T = \partial \chi_C$ où C est un convexe fermé, l'itération (3-3-3) devient :

$$x^{k+1} = \text{Proj}_C \circ J^S_{\lambda_k}(x^k)$$

Remarquons que le produit de deux résolvantes d'opérateurs maximaux monotones n'est pas forcément un opérateur proximal. Par contre, lorsque l'une de ces deux résolvantes est un projecteur, le produit est proximal (fermement non expansif) ; c'est ce que nous montrons par le lemme suivant :

**Lemme 3-3-0**

Soit U un opérateur fermement non expansif, A un sous-espace de X, $B = A^\perp$ et $P_A$ et $P_B$ deux projecteurs sur A et B respectivement. Alors $J = P_A \circ U$ est un opérateur fermement non expansif.

**Preuve :**

Soient x et y deux éléments de l'espace X ;
U est un opérateur proximal donc

$$||Ux-Uy||^2 \leq ||x-y||^2-||(I-U)x-(I-U)y||^2$$

$P_A$ est un projecteur sur A alors

$$||P_A x- P_A y||^2 = ||x-y||^2-||P_B x- P_B y||^2$$

$$||Jx-Jy||^2=||Ux-Uy||^2-||P_{B}oUx- P_{B}oUy||^2$$

$$\leq||x-y||^2-||(I-U)x-(I-U)y||^2-||P_{B}oUx- P_{B}oUy||^2$$

$$\leq||x-y||^2-||(I-U+P_{B}oU)x-(I-U+P_{B}oU)y||^2$$

Or

$$P_{B}oU = U- P_{A}oU$$

donc

$$||Jx-Jy||^2\leq||x-y||^2-||(I-J)x-(I-J)y||^2$$

### 3-3-2 Algorithme proximal projeté (PR)

#### 3-3-2-1 Cas général

**L'algorithme (PR)**

***0- Initialisation***

Soit $x^0 \in X$ ;

***1- Itération Proximale***

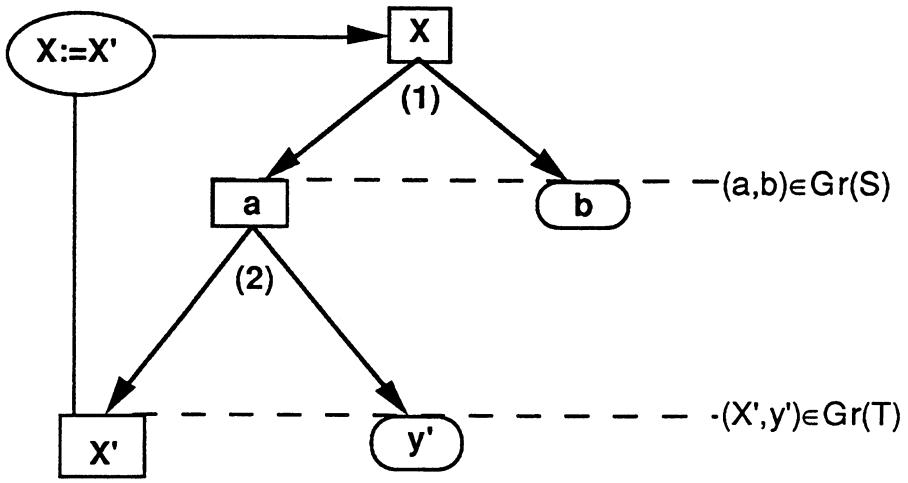
$$x^{k+1} = J^{T}_{\lambda_k} o\ J^{S}_{\lambda_k} x^k$$

***2- Test***

**Si** $x^{k+1} = x^k$ alors ***Fin***

**Sinon** k:=k+1 et aller à **1-**

On peut voir une itération de cet algorithme à travers la figure suivante :




**(1) :** Décomposition sur le graphe de S

**(2) :** Décomposition sur le graphe de T

*Figure représentant le schéma général d'une itération de l'algorithme (PR)*

D'après le théorème 1-2-20, la suite générée par l'algorithme (PR), converge vers un point qui est solution du problème régularisé (3-3-1), lorsque le paramètre $\lambda$ est constant et que ce problème admet une solution. Cette solution n'est pas en général celle du problème original (3-1-1). Cette convergence peut être démontrée en utilisant le lemme 3-3-0 et le théorème1 de Opial(1967).

Nous remarquons que, plus le paramètre $\lambda$ est grand, plus cette convergence est rapide et plus la "solution" $x_\lambda$ est éloignée de la solution du problème de départ. Par contre, lorsque le paramètre $\lambda$ tend vers zéro d'une certaine manière, on voit facilement que le point $x_\lambda$ s'approche de la solution $x^*$ du problème de départ, ce qui est en plein accord avec les tests numériques que nous avons effectués.

Grâce aux expérimentations numériques, nous avons pu constater que pour certains choix de la suite $\{\lambda_k\}$, on a la convergence. Pour plus de détails sur cette discussion, nous faisons référence à Mahey et al(1992b). D'autres auteurs ont étudié la convergence de cet algorithme, notamment Passty(1979), qui a montré qu'on a au moins une convergence ergodique.

En effet, soit la suite $\{z^k\}$ définie par :

$$z_k = \frac{\sum_{j=0}^{k} \lambda_j x_j}{\sum_{j=0}^{k} \lambda_j} \qquad \textbf{(3-3-4)}$$

Le théorème suivant est dû à Passty(1979).

**Théorème 3-3-2 :**

Soient $\{x^k\}$ et $\{z^k\}$ deux suites générées respectivement par les règles (3-3-3) et (3-3-4) ; si

$$\sum_{j=0}^{\infty} \lambda_j = +\infty \quad \text{et} \quad \sum_{j=0}^{\infty} \lambda_j^2 < +\infty$$

Alors

$$[\, Zer(L) = \varnothing \,] \Rightarrow [\, \|z^k\| \to \infty \,]$$

et

$$[\, Zer(L) \neq \varnothing \,] \Rightarrow [\, z^k \to z^* \in Zer(L)$$

où

$$Zer(L) = \{\, x \in X \mid 0 \in Tx \}.$$

Un nouvel algorithme, dû à Mahey et al (1992b), que nous allons voir dans la suite de ce paragraphe et qui est à deux étapes, permet, sous certaines conditions, de converger vers la solution du problème de départ. Cet algorithme est une version améliorée de l'algorithme PR.

**Remarque 3-3-1 :**

Soient $C_1$ et $C_1$ deux sous-ensembles convexes fermés tels que

$$\overline{\overset{o}{C_1 \cap C_2}} \neq \varnothing$$

si

$$S = \partial(\chi_{C_1}) \text{ et } T = \partial(\chi_{C_2})$$

Alors la suite générée par l'algorithme (PR) converge vers un point qui est solution du problème original (3-1-1). En effet, il est facile de vérifier que, dans ce cas spécial, les problèmes régularisés coïncident. La raison en est que pour tout $\lambda>0$, si C est un convexe fermé, la résolvante de $\partial\chi_C$ est la projection sur C.

De plus, une solution d'un problème régularisé (pour un $\lambda>0$) est une solution du problème originel.

Lorsqu'on génère une suite $\{x^k\}$ par l'algorithme (PR) avec un paramètre constant, on sait qu'elle converge vers un point $x_\lambda$ vérifiant (3-3-2) ce qui est équivalent à dire que

$$0 \in y + S(x_\lambda + \lambda y) \quad \text{et} \quad y \in T\, x_\lambda$$

alors que x est solution de (3-1-1) si et seulement si

$$0 \in y + Sx \quad \text{et} \quad y \in T\, x$$

Intuitivement, on itère une suite à deux étapes. Dans la première étape, on itère l'algorithme (PR) avec un paramètre constant. Dans la deuxième étape, on fait tendre le paramètre vers zéro. Cette idée est confirmée par le fait que $x_\lambda$ tend vers $x^* \in Zer(L)$ quand $\lambda$ tend vers zéro. On obtient l'algorithme suivant :

**L'algorithme (PRC)**

***0- Initialisation***

Soit $x \in X$ ; $\lambda>0$; k=0 ; l=0.

***1- Itération Proximale***

$$x^{k+1} = J_\lambda^T \circ J_\lambda^S x^k$$

***2- Test 1***

| | | |
|---|---|---|
| **Si** | $x^{k+1} = x^k$ alors $x^* = x^k$ ; | ***aller à 3*** |
| **Sinon** | k:=k+1 et | ***aller à 1-*** |

***3- test 2***

| | | |
|---|---|---|
| **Si** | $0 \in Lx^*$ alors | ***fin*** |
| **Sinon** | k:=0 ; $x^0 := x^*$ ; l:=l+1 ; réduire $\lambda$ | ***aller à 0-*** |

On obtient ainsi une suite de points $\{x_\lambda\}$, chaque $x_\lambda$ étant solution du problème(3-3-1), avec $\lambda>0$ et donc vérifie la relation suivante :

$$x_\lambda = J_\lambda^T \circ J_\lambda^S x_\lambda \qquad \textbf{(3-3-5)}$$

La suite $\{\lambda\}$ est destinée à tendre vers zéro. On notera par $\{y_k\}$ la suite associée à la suite $\{x_k\}$ par

$$y_k = S_k x_k$$

Cette suite joue un rôle important dans les résultats de convergence suivants :

**Théorème 3-3-3 :**

**1 -** Soit x une solution de (3-1-1) et $y \in Tx \cap (-Sx)$ alors

$$\|y_k\| \leq \|y\| \qquad \forall \lambda_k>0.$$

**2 -** Soit x un point d'accumulation de la suite $\{x_k\}$ et supposons la suite $\{y_k\}$ bornée. Alors x est une solution du problème (3-1-1).

**3 -** Supposons que la suite $\{x_k\}$ admet un point d'accumulation. Alors le problème (3-1-1) admet une solution si et seulement si la suite $\{y_k\}$ est bornée.

**4 -** Si le problème (3-1-1) admet une solution unique $x^*$ et si la suite $\{x_k\}$ est bornée, alors $x_k \to x^*$ et $y_k \to y^*$, qui est l'élément de norme minimale dans $Tx^* \cap (-Sx^*)$.

**Preuve :** Mahey et al(1992b).

Un cas particulier où on a une solution unique est lorsque l'un des deux opérateurs est fortement monotone. Si T est fortement monotone, le problème (3-1-1) admet au plus une solution ; par conséquent le problème régularisé (3-3-1), où S est remplacé par son approximation de Yosida, admet une solution unique pour tout $\lambda>0$. Dans ce cas, l'assertion 4 du théorème précédent a été raffinée sous la forme du théorème suivant :

**Théorème 3-3-4 :**

Supposons T fortement monotone avec une constante $\alpha$ et que S est régularisé. Alors, le problème (3-1-1) admet une solution unique $x^*$ si et seulement si la suite $\{y_k\}$ est bornée. Dans ce cas $x_k \to x^*$ et $y_k \to y^*$, qui est l'élément de norme minimale dans $Tx^* \cap (-Sx^*)$.

Lorsque $T = \partial\chi_A$ où A est un sous-espace vectoriel de X, l'itération de l'algorithme (PR) peut être écrite comme suit :

$$x^{k+1} = \mathrm{Proj}_A(J_\lambda^S x^k)$$

et l'itération de l'algorithme (PRC) devient :

***1- Itération Proximale***

$$x^{k+1} = \mathrm{Proj}_A(J_\lambda^S x^k)$$

***2- Test 1***

**Si** $x^{k+1} = x^k$ alors $x^* = x^k$ ; ***aller à 3***

**Sinon** k:=k+1 et ***aller à 1-***

***3- test 2***

**Si** $0 \in Lx^*$ alors ***fin***

**Sinon** k:=0 ; $x^0 := x^*$ ; réduire $\lambda^0$ ***aller à 1-***

Une application de ces deux algorithme au cas fonctionnel, i.e. $S = \partial f$, $f \in \Gamma^\circ(X)$ et $T = \partial\chi_A$, A un sous-espace vectoriel de X, sera donnée au Chapitre 4, dans le cadre des méthodes de décomposition.

### 3-3-3 Remarque : Algorithme du sous-gradient projeté

Soit x une solution du problème (3-1-1), i.e.

$$0 \in (S + T)x \Leftrightarrow 0 \in \lambda(S + T)x$$

$$\Leftrightarrow x \in (I + \lambda S + \lambda T)x$$

$$\Leftrightarrow [(x - \lambda S)x] \cap [(I + \lambda T)x] \neq \emptyset$$

$$\Leftrightarrow x \in (I + \lambda T)^{-1}(x - \lambda Sx)$$

L'idée est d'approcher la résolvante de l'opérateur S par l'opérateur (I - $\lambda$S) et de construire une suite de points $\{x^k\}$, générée par la relation suivante :

$$x^{k+1} \in (I + \lambda T)^{-1}(x^k - \lambda_k S x^k) \qquad \textbf{(3-3-6)}$$

Lorsque $T = \partial\chi_C$ où C est un sous-ensemble convexe fermé de X et $S = \partial f$ où $f \in \Gamma^\circ(X)$, l'itération (3-3-6) n'est rien d'autre qu'une itération de l'algorithme de sous-gradient projeté.

Le schéma (3-3-6) présente certains inconvénients dont le fait que l'opérateur S n'est pas forcément univoque ; de plus la direction opposée au sous-gradient n'est pas toujours une direction de descente. Par contre, si $x^{k+1} = x^k$, alors il est facile de voir que $x^k$ est un zéro de (S+T).

Comme cas particulier de cet algorithme, nous trouvons l'algorithme du gradient projeté, cf Gold'stein (1964), Polyak et Levitin(1966), dont une version due à Bertsekas utilise une recherche linéaire, cf Bertsekas(1976).

## 3-4 Quelques algorithmes de type Primal-Dual

Dans le paragraphe suivant, nous allons essayer de voir comment on retrouve certains algorithmes de type Primal-Dual, qui dérivent de l'algorithme proximal. Nous traiterons certains cas particuliers, ferons le lien avec d'autres algorithmes et rappellerons certains résultats de convergence.

### 3-4-1 Algorithme de Peaceman-Rachford

Dans la littérature, nous trouvons plusieurs algorithmes qui font intervenir les résolvantes d'opérateurs monotones, bien avant qu'on n'ait parlé explicitement de l'algorithme proximal. L'algorithme de Peacemann-Rachford(1955) (cas linéaire) consiste en une double itération (3-3-6), sauf que le rôle des opérateurs est inversé d'une étape à l'autre :

$$x^{k+1} = (I + \lambda_k T)^{-1}(I - \lambda_k S)(I + \lambda_k S)^{-1}(I + \lambda_k T)(x^k) \qquad \textbf{(3-4-1)}$$

Cet algorithme est étudié dans Lions et Mercier(1979) dans le cas non linéaire. Dans le but de retrouver cet algorithme, nous revenons encore sur le problème (3-1-1) dont on considère une solution u.

$$0 \in (S+T)u \iff 0 \in (\lambda S + \lambda T)u \ ; \lambda > 0.$$

$$\iff u \in (I + \lambda S)u + \lambda T u \ ; \lambda > 0.$$

donc il existe un élément v de X tel que

$$v \in (I + \lambda S)u \text{ et } u \in v + \lambda T u \qquad \textbf{(3-4-2)}$$

$$\iff v - u \in \lambda S u \text{ et } u - v \in \lambda T u$$

On pose alors,

$$b = (v-u)/\lambda, \qquad \textbf{(3-4-3)}$$

Le problème (3-1-1) est équivalent au problème suivant :

Trouver $u \in D(S+T) = D(S) \cap D(T)$, tel que $\exists b \in X$ vérifiant :

$$(u,b) \in Gr(S) \cap Gr(-T)$$

et b,u et v vérifient les relations (3-4-2) et (3-4-3)

ce qui revient à chercher un élément v dont la décomposition proximale selon le graphe de $\lambda S$ est :

$$v = u + \lambda b \text{ avec } (u, b) \in Gr(S) \text{ et } (u,-b) \in Gr(T)$$

On veut génèrer deux suites $u^k$ et $v^k$ de façon unique. Pour cela, nous définissons un nouveau problème (P2) relatif à la variable v tel que $v^*$ est une solution du problème (P2) si et seulement si

$$u^* = (I + \lambda S)^{-1} v^*$$

où $u^*$ est une solution du problème de départ (3-1-1).

A partir d'ici, il est naturel d'utiliser une itération à étapes proximales.

$$u\text{-}v \in \lambda T\, u \quad \Leftrightarrow \quad 2u\text{-}v \in (I+\lambda T)u$$

$$\Leftrightarrow \quad u = (I+\lambda T)^{-1}(2u\text{-}v)$$

$$\Leftrightarrow \quad (I+\lambda S)^{-1}v = (I+\lambda T)^{-1}(2u\text{-}v)$$

$$= (I+\lambda T)^{-1}[2\ (I+\lambda S)^{-1}v - v]$$

$$\Leftrightarrow \quad 2(I+\lambda S)^{-1}v = 2(I+\lambda T)^{-1}[2(I+\lambda S)^{-1}v - v]$$

$$\Leftrightarrow \quad v = [v - 2(I+\lambda S)^{-1}v] + 2(I+\lambda T)^{-1}[2(I+\lambda S)^{-1}v - v]$$

$$\Leftrightarrow \quad v = [I - 2(I+\lambda S)^{-1}]v + 2(I+\lambda T)^{-1}[2(I+\lambda S)^{-1} - I]v$$

$$\Leftrightarrow \quad v = [2(I+\lambda T)^{-1} - I] \circ [2(I+\lambda S)^{-1} - I]v$$

C'est ainsi que nous retrouvons l'algorithme de point fixe, équivalent à celui de Peaceman-Rachford (dans le cas univoque), cf Lions et Mercier(1979), que nous présentons ci-dessous.

Pour un opérateur maximal monotone T sur X et un réel positif $\lambda$, posons :

$$N_\lambda^T = [2(I+\lambda T)^{-1} - I] \qquad \textbf{(3-4-4)}$$

Remarquons tout d'abord que si T est maximal monotone, l'opérateur défini par la relation (3-4-4) ci-dessus, est l'opérateur non expansif associé à l'opérateur proximal de T, $(I+\lambda T)^{-1}$ (proposition 1-2-8 (1)). Par contre rien ne prouve qu'il est fermement non expansif, ce qui présente un point faible pour les algorithmes n'utilisant que de tels opérateurs. Un point positif de cet algorithme est que si la suite générée par (a), ci-dessous, converge vers $v^*$ alors la suite générée par (b), ci dessous, converge aussi vers une solution du problème de départ, contrairement à d'autres méthodes où la suite converge sans que l'on ait aucune information sur la solution du problème.

## L'algorithme (RP)

***0- Initialisation***

*donner* $\lambda>0$, $u\in D(S)\cap D(T)$ ; $u = (I+\lambda S)^{-1}v$

***1- Itération proximale***

$$\mathbf{v}^{k+1} := N_\lambda^T N_\lambda^S \mathbf{v}^k \qquad (a)$$

$$\mathbf{u}^{k+1} := (I+\lambda S)^{-1}\mathbf{v}^{k+1} \qquad (b)$$

***2- Test***

**Si** $\mathbf{v}^{k+1} - \mathbf{u}^{k+1} \in \lambda T\mathbf{u}^{k+1}$ ; $\mathbf{u}^{k+1}$ est zéro de (T+S) ; **fin.**

**Sinon** k:=k+1 et relancer 1)

Remarquons que si on ne se sert pas de $\{u^k\}$ pour le test d'arrêt, on peut ne faire appel à l'itération (b) qu'après que la suite $\{v^k\}$ ait convergé. Quand on considère le cas particulier ; $T=\partial\chi_A$, où A est un sous-espace vectoriel de X et on pose $B=A^\perp$ on a :

$$N_1^T = Proj_A - Proj_B \qquad (c)$$

Plus généralement

$$N_1^T z = x - y$$

où (x, y) est la décomposition proximale (unique) de z dans le graphe de T.

**Remarque 3-4-0**

Grâce à ces propriétés on peut facilement montrer que dans le cas particulier ci-dessus, l'itération (a) de l'algorithme (RP) peut être écrite sous la forme :

$$v := N_1^{(\lambda S)_A}(v) \qquad (d)$$

où $S_A$ est l'opérateur inverse partiel de S. En effet, dans ce cas particulier, la relation (a) précédente est équivalente à :

$$v := N_\lambda^{\partial\chi_A} z$$

où

$$z = x-y,\ y\in Sx \text{ et } v=x+y$$

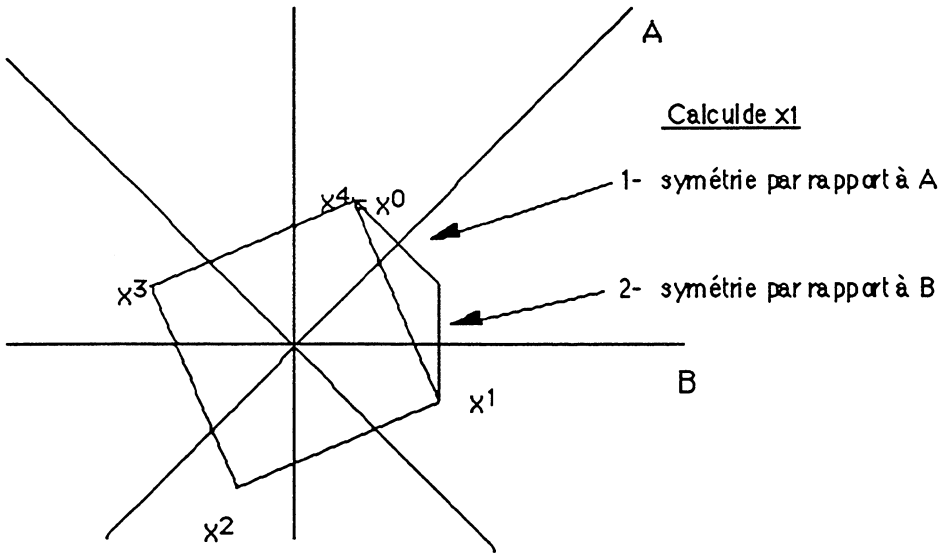
d'après la relation(c), on déduit que

$$v:=x_A - x_B - y_A+y_B = (x_A+y_B)-(y_A+x_B)$$

et la définition de l'inverse partiel nous conduit à la relation (c).

Le problème que l'on rencontre avec cet algorithme, c'est qu'il ne converge pas forcément même si les opéateurs T et S sont maximaux monotones (voire fortement monotones). Ci-dessous un exemple où l'algorithme (RP) génère une suite qui ne converge pas ; $T=\partial\chi_A$ et $S=\partial\chi_B$ où A et B sont deux sous-espaces vectoriels de X.

$x^1$, $x^2$, $x^3$ et $x^4$ sont les quatres premiers itérés de l'algorithme (RP)




Exemple où l'algorithme (RP) ne converge pas

Si on suppose que l'un des deux opérateurs est fortement monotone, on peut se demander si cette propriété supplémentaire va nous assurer la convergence. La réponse est non et nous nous référons à l'exemple donné dans Eckstein(1989), p81. Par contre, une des conditions suffisantes pour que cet algorithme converge est de supposer que l'un des deux opérateurs, S ou T est univoque et fortement monotone. C'est une conséquence immédiate du théorème de convergence de Lions-Mercier(1979), dans lequel la condition suffisante est que l'un des deux opérateurs vérifie la propriété dite de dimension finie de Lions et Mercier que nous définissons ci-dessous. Cette définition a été transcrite de Eckstein(1989).

**Définition 3-4-1 :**

Un opérateur T sur X vérifie la propriété dite de dimension finie de Lions et Mercier s'il est univoque et si pour tout $x\in dom(T)$ et pour toute suite convergente $\{x_k\}\subseteq dom(T)$ telle que $\{Tx_k\}$ soit convergente aussi, on a :

$$\lim_k \langle Tx_k - Tx, x_k - x\rangle = 0 \Rightarrow \lim_k x_k = x.$$

Remarquons que cet algorithme est en quelque sorte un algorithme de point fixe. Sa convergence nécessite des conditions assez fortes ; c'est ainsi que nous avons pensé à

renforcer ces propriétés en le transformant de sorte à obtenir un algorithme un peu différent, mais sûrement convergent, si le problème admet une solution.

En effet, il a suffi d'utiliser un résultat de Browder et Petryshyn(1967) et Opial(1967), pour obtenir cet algorithme modifié, la même démarche a d'ailleurs été utilisée par Goldstein.

**L'algorithme (RPM)**

***0- Initialisation***

donner $\lambda>0$ et $\alpha\in(0,1)$ et $u\in D(S)\cap D(T)$ ; $u = (I+\lambda S)^{-1}v$

***1- l'itération proximale***

$$\mathbf{v}^{k+1} := \alpha\mathbf{v}^k + (1-\alpha)N_\lambda^T N_\lambda^S \mathbf{v}^k \qquad (a')$$

$$\mathbf{u}^{k+1} := (I+\lambda S)^{-1}\mathbf{v}^{k+1} \qquad (b)$$

***2- Test***

**Si** $\mathbf{v}^{k+1} - \mathbf{u}^{k+1} \in \lambda T\mathbf{u}^{k+1}$ ; $\mathbf{u}^{k+1}$ est zéro de (T+S) ; **fin.**

**Sinon** k:=k+1 et relancer 1)

D'après Opial(1967), si le problème (3-1-2) admet une solution alors la suite générée par cet algorithme converge vers la solution de ce problème.

**Remarque**

Si $\alpha=1/2$ alors l'algorithme (PRM) coïncide avec l'algorithme de Douglas-Rachford, dont l'opérateur associé est proximal. (Voir aussi la conclusion du présent chapitre).

## 3-5 L'algorithme de l'inverse partiel

Soit A un sous-espace vectoriel de X et $B = A^{\perp}$ le complément orthogonal de A dans X. Nous considérons le problème (3-1-2) :

Trouver $x \in A$ et $y \in B$ tels que $y \in Tx$

T étant un opérateur maximal monotone sur X.

Nous savons que tout élément z de X peut être écrit de façon unique sous la forme :

$z = x + y$ ; $x \in A$ et $y \in B$

et que

$x = Proj_A z$ et $y = Proj_B z$.

Nous noterons dans tout ce qui suit $z_A$ la projection de z sur A et $z_B$ la projection de z sur B.

Dans le chapitre 1, nous avons défini l'opérateur $T_A$, inverse partiel d'un opérateur T (définition 1-2-4). Grâce à cet opérateur, nous pouvons faire le lien entre un problème de type (3-1-2) et un problème de type (2-1-1).

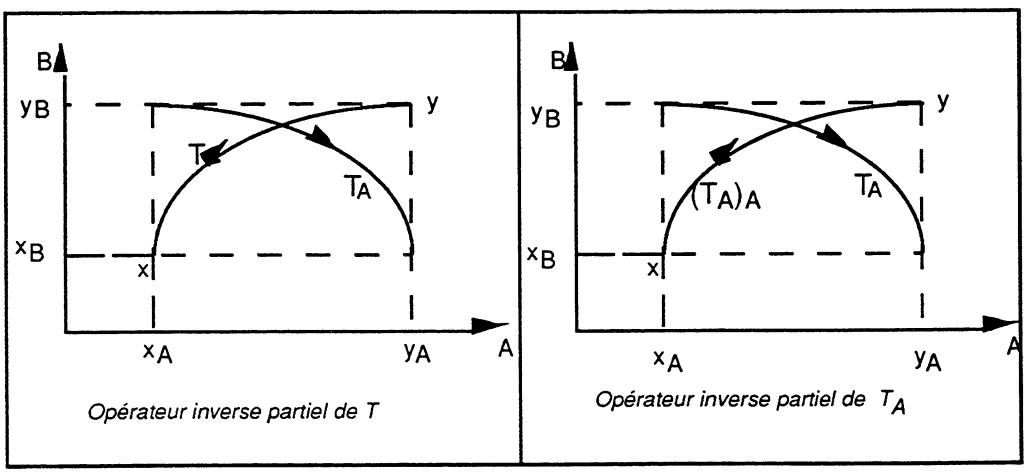
**Proposition 3-5-1 :**

Soient x et y deux éléments de X, alors (x,y) est une solution du problème (3-1-2) si et seulement si z=x+y est une solution du problème (2-1-1).

Nous savons aussi que l'opérateur T est maximal monotone ssi l'opérateur $T_A$ l'est aussi ; la condition nécessaire est due à Spingarn(1983). La condition suffisante est immédiate, il suffit de remarquer que l'opérateur inverse partiel par rapport à A est idempotent i.e. pour un opérateur T et un sous-espace vectoriel A de X on a

$$(T_A)_A = T$$

La figure suivante met en évidence cette propriété :



Partant du fait que $T_A$ est maximal monotone et de la proposition 3-5-1 et s'inspirant des travaux de Rockafellar(1976a), J.Spingarn(1983) a donné un algorithme de résolution des problèmes de type (3-1-2) : l'algorithme de l'inverse partiel. Cet algorithme qui est du type primal-dual, bénéficie des mêmes propriétés de convergence que l'algorithme du point proximal (AP2).

***L'algorithme (AIP)***

***0- Initialisation***

donner $\lambda_0>0$, Choisir $x^0 \in A$ et $y^0 \in B$ ; n:=0.

***1- A) l'itération proximale***

Trouver $x'^n$ et $y'^n \in X$ tels que

$$x'^n+y'^n = x^n+y^n \text{ et}$$

$$\left(\frac{1}{\lambda_n}(y'^n)_A + (y'^n)_B\right) \in T\left((x'^n)_A + \frac{1}{\lambda_n}(x'^n)_B\right) \; ; \; \lambda_n > 0$$

***B) Projections sur les sous-espaces***

$$x^n := (x'^n)_A \text{ et } y^n := (y'^n)_B$$

***2- Test***

**Si** $(x^n,y^n) \in Gr(T)$ alors $(x^n,y^n)$ est solution **Fin**

**Sinon** aller à **1-**

Nous rappelons ci-dessous quelques résultats de convergence de cet algorithme dus à Spingarn(1983). En fait tous ces résultats sont analogues à ceux de R.T.Rockafellar(1976), vu que l'algorithme (AIP) est équivalent à l'algorithme (AP2) appliqué à la recherche d'un zéro de $T_A$. Ces résultats tiennent compte des erreurs éventuelles de calcul. Le terme

$$||z^n - J^{T_A}_{\lambda_n}||$$

peut être considéré comme étant l'erreur totale résultant de l'itération n de l'algorithme (AIP). Cette erreur est l'erreur commise sur le calcul de $x'_n$, cf Spingarn(1983), i.e., si

$$x''_n - x'_n = e_n$$

alors

$$||z^n - J^{T_A}_{\lambda_n}|| = ||e_n||$$

où $x''_n$ est la valeur approchée de $x'_n$ obtenue.

**Théorème 3-5-2 :**

Supposons que le problème (3-1-2) admet une solution et que

i- $0 < c < \lambda_n$ , $\forall n \in \mathbb{Z}_+$ ,

ii- $\sum_{n>0} ||e_n|| < +\infty$

Alors les suites $\{x^n\}$ et $\{y^n\}$ générées par l'algorithme (AIP) sont bornées et $(x^n \rightarrow x^*)$ et $(y^n \rightarrow y^*)$ où $(x^*,y^*)$ est une solution de (3-1-2).
Si le problème (3-1-2) n'admet pas de solution alors

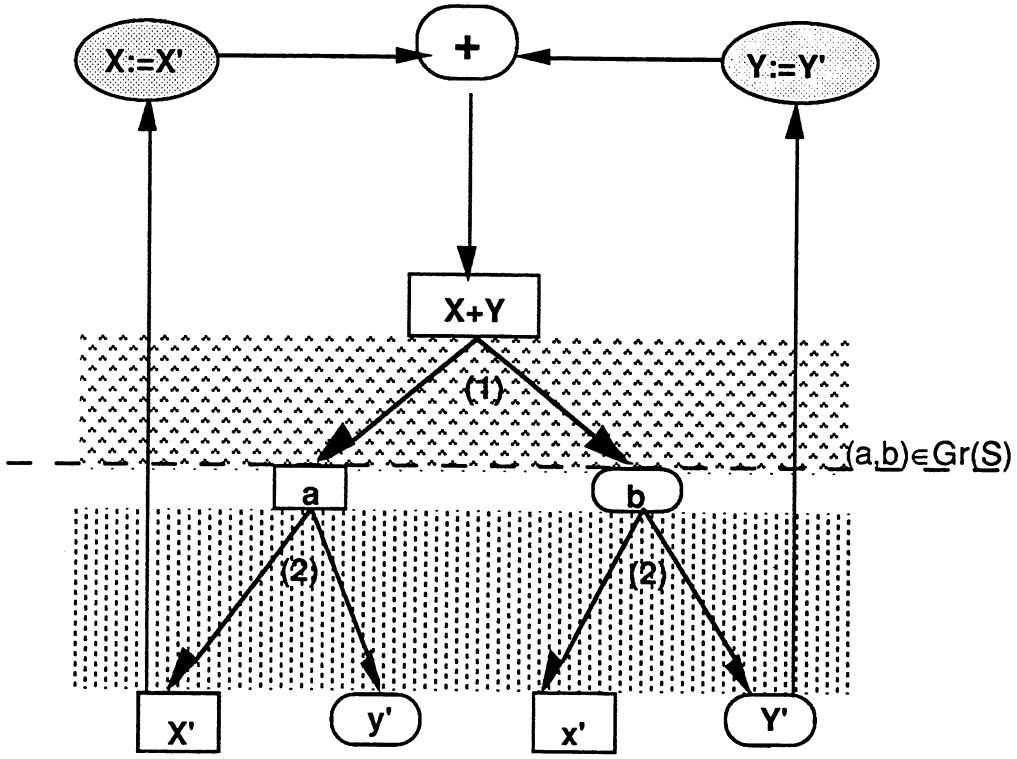
$$||x^n + y^n||^2 \rightarrow +\infty$$

Comme pour l'algorithme (APP), nous remarquons que sous les hypothèses, i- et ii- du théorème précédent, le problème (3-1-2) admet une solution si et seulement si les suites $\{x^n\}$ et $\{y^n\}$ générées par l'algorithme (AIP) sont bornées.

**Remarque 3-5-3 :**

L'algorithme (AIP) n'est pas implémentable pour une valeur du paramètre $\lambda_k \neq 1$, c'est pour cela que nous nous sommes abstenus de rappeler les résultats de convergence, mettant en évidence l'influence du paramètre sur la performance de l'algorithme. Le lecteur intéressé pourra les trouver dans J.Spingarn(1983).

Dans le cas où la suite des paramètres $\{\lambda_k\}$ est constante et vaut 1, l'itération de l'inverse partiel est implémentable (calculable) pour certains opérateurs particuliers et l'algorithme peut être schématisé par la figure suivante :




**(1)** : Décomposition sur le graphe de S

**(2)** : Décomposition sur le graphe de $\partial\chi_A$

A partir de cette représentation, on peut facilement voir qu'une itération de l'algorithme AIP($\lambda$=1) revient à faire deux décompositions successives, par rapport à des graphes d'opérateurs maximaux monotones ; la première par rapport au graphe de S et la seconde, par rapport au graphe de T ($=\partial\chi_A$). Disons que la seconde décomposition s'effectue sur les deux points obtenus par la première. Cette remarque nous a amené sur plusieurs voies de généralisation de ce schéma. Le but était de trouver un algorithme qui nous permet de résoudre les problèmes du type :

$$\text{Trouver } (x,y)\in Gr(S)\cap Gr(T) \qquad \textbf{(3-5-4)}$$

en introduisant un paramètre (réel positif) nous permettant d'améliorer la vitesse de convergence vers la solution. Le schéma de cet algorithme est analogue au schéma ci-dessus. L'algorithme résultant est l'algorithme de décomposition proximale que nous traitons dans le paragraphe suivant.

## *3-6 L'algorithme de Décomposition Proximale*

Le but principal est toujours de résoudre le problème (3-1-2) que l'on peut écrire sous la forme suivante :

$$\text{Trouver } (x,y)\in AxB\cap Gr(T). \qquad \textbf{(3-6-1)}$$

T étant toujours un opérateur maximal monotone, A un sous-espace vectoriel de X et B son orthogonal.

Nous rappelons que $Gr(\partial\chi_A) = AxB$. Par conséquent, le problème (3-6-1) devient un cas particulier du problème (3-5-4). Dans l'algorithme de l'inverse partiel, que nous avons traité au paragraphe précédent, l'itération proximale est une décomposition proximale sur le graphe de l'opérateur T.

Nous signalons que l'algorithme que nous traitons dans la section suivante est équivalent à l'algorithme AIP appliqué a $\lambda T$ au lieu de T. Notre contribution est d'avoir montré la convergence de façon directe sans passer par l'inverse partiel.

### 3-6-1 Décomposition Proximale

Trouver l'unique point $(x'_k,y'_k)$ tel que

$$x'_k+y'_k = x_k+y_k \text{ et } (x'_k,y'_k)\in Gr(T)$$

Si $(x'_k,y'_k)\in AxB$ alors fin.

Sinon $(x_{k+1},y_{k+1}) = ((x'_k)_A,(y'_k)_B)$.

L'unique solution de l'itération de décomposition proximale est donnée par :

$$x'_k = (I+T)^{-1}(x_k+y_k)$$

$$y'_k = (I+T^{-1})^{-1}(x_k+y_k). \qquad \textbf{(3-6-2)}$$

En fait, on a besoin d'un seul calcul proximal, vu qu'on a :

$$(I+T^{-1})^{-1} = I - (I+T)^{-1}$$

Nous proposons alors un algorithme de décomposition proximale modifié en introduisant deux facteurs d'échelle $\lambda$ et $\mu$. L'inclusion du problème (3-1-2) peut être écrite de deux formes différentes :

$$y\in Tx \quad \Leftrightarrow \quad x+\lambda y \in (I+\lambda T)x$$

$$x\in T^{-1}y \quad \Leftrightarrow \quad y+\mu x \in (I+\mu T^{-1})y$$

On a donc l'itération de point fixe (3-6-3) qui suivera.

### 3-6-2 Décomposition Proximale modifiée

L'algorithme en question génère une suite par l'itération suivante :

$$x'_k = (I+\lambda T)^{-1}(x_k+\lambda y_k)$$

$$y'_k = (I+\mu T^{-1})^{-1}(x_k+\mu y_k) \qquad \textbf{(3-6-3)}$$

**Si** $(x'_k,y'_k)\in AxB$ alors **fin.**

**Sinon** $(x_{k+1},y_{k+1}) = ((x'_k)_A,(y'_k)_B)$.

On peut voir que cette itération proximale modifiée est déterminée de façon unique et correspond à la décomposition proximale sur le graphe de $\lambda T$ si $\lambda\mu = 1$. On retrouve alors la version de l'algorithme (AIP) avec facteur d'échelle proposée par Spingarn (1985). Dans Idrissi et al(1989), il a été signalé que la performance de l'algorithme (AIP) est très sensible à la variation du facteur d'échelle ; nous en donnerons une explication partiel. Dans le cas fortement monotone, nous donnerons une estimation de la valeur optimale de ce paramètre. Avant de passer à la description de l'algorithme de décomposition proximale, nous allons rappeler quelques résultats sur les opérateurs maximaux fortement monotones (Définition 1-2-1).

Nous rappelon qu'un opérateur T est dit Lipschitzien de rapport L si

$$\forall\, x,x'\in X \quad \forall\, y\in Tx\,, \quad \forall\, y'\in Tx'$$

$$||y'-y|| \le L||x'-x|| \qquad \textbf{(3-6-4)}$$

Pour les opérateurs qui sont fortement monotones de rapport a et qui vérifient (3-6-4), on a cf Mahey et al(1992a) :

$$a||x'-x|| \le ||y'-y|| \le L||x'-x||$$

Lorsque T est un opérateur linéaire représenté par une matrice définie positive M, les meilleures estimations pour les valeurs de a et L sont, respectivement, la plus grande et la plus petite valeur propre de M. Nous avons énoncé au Chapitre 1 que si T est un opérateur maximal monotone, alors $\lambda T$ l'est aussi, que la résolvante de T est définie sur X tout entier et qu'elle est fermement non expansive(voir (1-2-5) qui est équivalente à (1-2-6) ) ; cette propriété est plus forte que la faible contraction.

**Proposition 3-6-1 :**

Lorsque l'opérateur T est fortement monotone (1-2-2), de rapport a, alors pour tout $x,x'\in X$, on a :

$$||\mathbf{J}x - \mathbf{J}x'||^2 \le [1/(1+a)]\langle \mathbf{J}x - \mathbf{J}x', x - x'\rangle \qquad \textbf{(3-6-6)}$$

où **J** est la résolvante de T, ( $\mathbf{J} = (I+T)^{-1}$)

**Preuve :**

Soient $y = Jx$, $y' = Jx'$, $z = (I-J)x$ et $z' = (I-J)y$

alors

$$x \in y+Ty \text{ et } x' \in y'+Ty'$$

$$\Leftrightarrow \quad z \in Ty \text{ et } z' \in Ty'$$

Par la monotonie de T on a :

$$\langle z-z', y-y' \rangle \geq 0$$

ce qui entraine

$$\langle x-x', y-y' \rangle \geq ||y-y'||^2.$$

Par un simple calcul, on obtient

$$||y'-y||^2 \leq ||x'-x||^2 - ||z'-z||^2.$$

Dans le cas fortement monotone, on a

$$\langle z-z', y-y' \rangle \geq a||y-y'||^2.$$

donc

$$\langle x-x', y-y' \rangle \geq (1+a)||y-y'||^2.$$

**Théorème 3-6-2 :**

Soit A et B deux sous-espaces complémentaires orthogonaux de X. L'application $\mathcal{F}$ qui fait correspondre, à chaque élément $(x,y) \in AxB$, la décomposition proximale $(u,v)$ de $(x+y)$ sur le graphe de T, est non expansive. De plus, si T est fortement monotone de rapport a et Lipschitzien de rapport L, alors $\mathcal{F}$ est une contraction.

**Preuve :**

Soitent $z = x+y$, $z' = x'+y'$, $(u,v) = \mathcal{F}(x,y)$ et $(u',v') = \mathcal{F}(x',y')$

On a alors les relations suivantes :

$$u = Jz \text{ et } v = z-u$$

$$u' = Jz' \text{ et } v' = z'-u'$$

Si on note par $||.||_X$ la norme Euclidienne dans XxX, on a

$$||(u-u', v-v')||^2 = ||u-u'||^2 + ||v-v'||^2$$

L'opérateur J étant fermement non expansif, on

$$||u-u'||^2 + ||v-v'||^2 \leq ||z'-z||^2$$

or $(x'-x) \in A$ est orthogonal à $(y'-y) \in B$, donc

$$||z'-z||^2 = ||x+y-x'-y'||^2 = [||(x-x', y-y')||_X]^2.$$

Ainsi, on a montré que $\mathcal{F}$ est non expansive.

Si T est fortement monotone de module a, on sait que,

$$||u-u'||^2 \leq [1/(1+a)]<z-z',u-u'>$$

et par conséquent, on a :

$$||(u-u',v-v')||_X^2 \leq ||z'-z||^2 - 2a||u'-u||^2 \qquad (*)$$

Si de plus, T est Lipschitzien de rapport L, on peut écrire

$$||z-z'|| \leq (1+L)||u-u'||$$

ou

$$||u-u'|| \geq [1/(1+L)]||z-z'||$$

en remplaçant dans la relation (*) ci-dessus, on obtient :

$$[||(u-u',v-v')||_X]^2 \leq \{1-2a/[(1+L)^2]\}||z'-z||^2 \qquad \textbf{(3-6-6)}$$

ce qui implique que $\mathcal{F}$ est une contraction de rapport $\{1-2a/[(1+L)^2]\}$.

Nous allons présenter par la suite un algorithme qui alterne, à chaque itération, une décomposition proximale et une projection sur le sous-espace vectoriel AxB de XxX : *L'Algorithme de Décomposition Proximale sur un Graphe.*

***L'algorithme ADPG***

***0- Initialisation***

Choisir $x^0 \in A$ et $y^0 \in B$ ; n:=0.

***1- A) Itération proximale***

**Si** $(x^n,y^n) \in Gr(T)$ alors **Fin**

**Sinon** Trouver $(x'_n,y'_n)$ décomposition proximale de $x^n+y^n$ sur le graphe de T, i.e.

$$x'_n = (I+T)^{-1}(x^n+y^n)$$

$$y'_n = x^n+y^n - x'_n$$

***B) Projections sur les sous-espaces***

Décomposition de $x'_n$ et de $y'_n$ sur le graphe de $\partial\chi_A$.

$$x^n := (x'_n)_A \text{ et } y^n := (y'_n)_B$$

$$n := n+1.$$

Signalons que l'algorithme ci-dessus coïncide avec l'algorithme AIP. Il nous servira d'algorithme introductif afin d'aborder plus facilement la version paramétrée.

Une itération de cet algorithme peut être représentée comme suit :

$$(x,y)\in AxB \mapsto (u,v) = \mathcal{F}(x,y) \mapsto (u_A,v_B) = \mathcal{H}(x,y)$$

$\mathcal{F}$ étant une contraction, l'application $\mathcal{H}$ définie ci-dessus est non expansive. Nous montrons maintenant que tout point fixe de l'algorithme (ADPG) est une solution du problème (3-6-1).

**Théorème 3-6-3 :**
Tout point fixe de l'algorithme (ADPG) est solution de (3-6-1) et inversement.

**Preuve :**
Ce résultat peut être montré en utilisant les propriétés de l'inverse partiel d'un opérateur maximal monotone (Spingarn(1983)). Nous proposons ci-dessous une démonstration plus directe.

Soit $Proj_{AxB}$, la projection orthogonale sur AxB et soit (x,y) un point fixe de l'algorithme (ADPG), alors

$$(x,y) = \mathcal{H}(x,y) = Proj_{AxB}\circ\mathcal{F}(x,y)$$

Soit $(u,v) = \mathcal{F}(x,y)$

Si $(u,v)\in AxB$, alors (x,y) est une solution du problème (3-6-1). Sinon, on a :

$$(u-x,v-y)\in L = \{(a,b)\in XxX \mid a+b = 0\}$$

Mais , puisque

$$(x,y) = Proj_{AxB}(u,v)$$

on peut voir que

$$(u-x,v-y)\in BxA.$$

A et B étant deux sous-espaces orthogonaux de X, l'unique point d'intersection de L et AxB est (0,0). Par conséquent (x,y) = (u,v) et (x,y) est solution du problème (3-6-1).

Inversement, Si (x,y) est solution de (3-6-1) alors

$$\mathcal{F}(x,y) = (x,y) \text{ et } (x,y)\in AxB$$

ce qui veut dire que (x,y) est un point fixe de l'algorithme (ADPG).

Avant de citer certains résultats de convergence de l'algorithme (ADPG), nous allons énoncer un lemme sur la nature d'un point d'accumulation d'une suite générée par une itération non expansive.

**Lemme 3-6-4 :**

Soit $\mathcal{N}$ une application non expansive dans X et soit une suite $\{x^n\}$ définie par :

$$x^{n+1} = \mathcal{N}(x^n) \qquad n=0, 1, 2, ...$$

Supposons que cette suite $\{x^n\}$ est bornée et que

$$||x^{n+1} - x^n|| \to 0 \text{ quand } n \to +\infty$$

**Alors**

**i-**Tout point d'accumulation de $\{x^n\}$ est un point fixe de $\mathcal{N}$.

**ii-**La suite $\{x^n\}$ converge.

**Preuve :**

**i-** Les hypothèses impliquent qu'il existe une sous-suite, convergente, $\{x^{\Phi(n)}\}$ de $\{x^n\}$. Soit $x^*$ son point d'accumulation. Or

$$\mathcal{N}(x^{\Phi(n)}) = x^{\Phi(n)+1}$$

et

$$||x^{\Phi(n)+1} - x^{\Phi(n)}|| \to 0 \text{ quand } n \to +\infty$$

donc en utilisant la continuité de $\mathcal{N}$, on déduit que $\mathcal{N}(x^*) = x^*$

**ii-** Comme $\{x^{\Phi(n)}\}$ converge vers $x^*$, pour tout réel $\varepsilon>0$ ; il existe un entier $K>0$ tel que, pour tout entier $n>K$

$$||x^{\Phi(n)} - x^*|| < \varepsilon$$

Par conséquent, si on pose $N° = \Phi(K)$ et vu que $\mathcal{N}$ est non expansive, alors pour tout $n>N°$, on a

$$||x^n - x^*|| < \varepsilon$$

Ce qui nous permet de conclure que $x^n \to x^*$ ∎

Ce lemme a été démontré autrement dans la littérature, (i) est dans Browder et Petryshyn(1967), (ii) est dans Opial(1967).

**Théorème 3-6-5 :**

Supposons que l'ensemble des solutions du problème (3-6-1) n'est pas vide, alors la suite $\{(x^k,y^k)\}$ générée par l'algoritme (ADPG) converge vers une solution de (3-6-1).

**Preuve :**

A l'itération k de l'algorithme (ADPG), nous pouvons écrire :

$$(u_k,v_k) = \mathcal{F}(x^k,y^k)$$

Soit (x,y) une solution de (3-6-1) ; on a donc

$$(x,y) = \mathcal{F}(x,y)$$

D'après, le théorème 3-6-2, $\mathcal{F}$ est un opérateur non expansif, (ce qui n'est pas suffisant pour la convergence, cf Browder et Petryshin(1967)). Donc

$$||(u_k-x,v_k-y)||^2_X \leq ||(x^k-x,y^k-y)||^2_X$$

Maintenant, si $(x^{k+1},y^{k+1})$ est la projection de $(u_k,v_k)$ sur AxB et $(x,y)\in$ AxB, alors en tenant compte de l'inéquation ci-dessous, on obtient :

$$||(x^{k+1}-x,y^{k+1}-y)||^2_X \leq ||(x^k-x,y^k-y)||^2_X - ||(x^{k+1}-u_k,y^{k+1}-v_k)||^2_X$$

Par conséquent, la limite de $\{(x^{k+1}-x,y^{k+1}-y)\}$ existe et le dernier terme dans l'inégalité précédente, $||(x^{k+1}-u_k,y^{k+1}-v_k)||^2_X$, tend vers zéro.

Or $x^{k+1}$ est la projection de $u_k$ sur A, et $y^{k+1}$ celle de $v_k$ sur B, nous pouvons déduire que la projection de $u_k$ sur B et celle de $v_k$ sur A tendent vers le vecteur nul. Nous pouvons immédiatement vérifier que :

$$\text{Proj}_A v_k = \text{Proj}_A(x^k+y^k-u_k) = x^k - x^{k+1}$$

et

$$\text{Proj}_B u_k = y^k - y^{k+1}$$

Ainsi, nous avons montré que :

$$\lim_{k\to+\infty} ||(x^{k+1}-x^k,y^{k+1}-y^k)||^2_X = 0$$

Remarquons que la suite $\{(x^k,y^k)\}$ est contenue dans le sous-ensemble compact C, de X, défini comme suit :

$$C = \{(r,s)\} \mid ||(r-x,s-y)||_X \leq ||(x^o-x,y^o-y)||_X \}$$

Utilisant le lemme 3-6-4 et le théorème 3-6-3, on obtient le résultat cherché. ∎

Nous attirons l'attention du lecteur sur le fait que l'opérateur $F=\text{Proj}_{AxB^o}\,\mathcal{F}$ est l'opérateur proximal associé à l'inverse partiel de T par rapport à A, donc fermement non expansif, argument dont nous ne nous sommes pas servis pour démontrer la convergence.

**Remarques 3-6-6 :**

**1 -** L'algorithme (ADPG) est équivalent à l'algorithme de l'inverse partiel pour $\lambda=1$.

**2 -** Si A=X et $\lambda_k=1$, alors les trois algorithmes (AP2), (AIP1) et (ADPG) sont équivalents.

**3 -** Plus généralement, une itération de l'algorithme (AP2) équivaut à une itération de (ADPG) appliquée à $\lambda_k T$, avec A=X.

### 3-6-3 L'Algorithme de Décomposition Proximale Parametrée Sur un Graphe

Dans ce paragraphe, nous introduisons une version de l'algorithme précédent, dans laquelle nous faisons intervenir un facteur d'échelle.

**Définition 3-6-7 :**

Soient $(x,y)\in XxX$, T un opérateur maximal monotone et $\lambda$ un réel positif. Alors, la décomposition proximale paramètrée de (x,y) sur le graphe de T est l'unique point (u,v) de XxX tel que :

$$u+\lambda v = x+\lambda y$$

$$(u,v)\in Gr(T)$$

L'existence et l'unicité de cette décomposition sont dues au fait que T est maximal monotone. Si $(u,v)\in Gr(T)$, on peut écrire que :

$$u+\lambda v\in u+\lambda Tu$$

$$\Rightarrow \quad u = (I+\lambda T)^{-1}(u+\lambda v)$$

$$= (I+\lambda T)^{-1}(x+\lambda y)$$

$$v = \lambda^{-1}(x+\lambda y-u) \qquad (r^*)$$

Utilisant l'opérateur $T^{-1}$, inverse de T, on peut symétriquement écrire pour $\mu>0$ :

$$v+\mu u\in v+\mu T^{-1}v$$

$$\Rightarrow \quad v = (I+\mu T^{-1})^{-1}(v+\mu u) \quad (*)$$

Posons

$$S = T^{-1},\ a = v \text{ et } b = \mu u+v$$

$$(*) \quad \Leftrightarrow \quad b-a\in \mu Sa$$

$$\Leftrightarrow \quad [(b-a)/\mu]\in Sa$$

$$\Leftrightarrow \quad (a)\in T[(b-a)/\mu]$$

$$\Leftrightarrow \quad (b/\mu)\in (I+(1/\mu)T)[(b-a)/\mu]$$

$$\Leftrightarrow \quad [(b-a)/\mu] = (I+(1/\mu)T)^{-1}(b/\mu)$$

$$\Leftrightarrow \quad a = \mu[I-(I+(1/\mu)T)^{-1}](b/\mu)$$

$$\Leftrightarrow \quad v = \mu[I-(I+(1/\mu)T)^{-1}](u+v/\mu)$$

et si $\mu = 1/\lambda$, on retrouve la relation (r*) précédente.

Ainsi on peut résumer la décomposition proximale paramétrée sur le graphe de l'opérateur T par :

$$u = (I+\lambda T)^{-1}(x+\lambda y)$$

$$v = (I+\mu T^{-1})^{-1}(\mu x+y)$$

que l'on peut voir comme une paramètrisation de la décomposition proximale sur le graphe d'un opérateur. Afin de préserver les propriétés désirées, un seul paramètre doit être introduit en imposant $\lambda\mu = 1$.

Nous allons maintenant décrire l'itération de la version paramétrée de l'algorithme (ADPG) :

**L'algorithme ADP2G**

***0- Initialisation***

donner $\lambda>0$, Choisir $x^0 \in A$ et $y^0 \in B$ ; n:=0.

***1- A) l'itération proximale***

**Si** $(x^n,y^n) \in Gr(T)$ alors **Fin**

**Sinon** Trouver $(u_n,v_n)$ décomposition proximale parametrée de $x^n+y^n$ sur le graphe de T, i.e.

$$u_n = (I+\lambda T)^{-1}(x^n+\lambda y^n)$$

$$v_n = (x^n+\lambda y^n - u_n)/\lambda$$

***B) Projections sur les sous-espaces***

$$x^n := (u_n)_A \text{ et } y^n := (v_n)_B$$

$$n := n+1.$$

Nous remarquons que dans l'algorithme de décomposition proximale paramètrée sur le graphe d'un opérateur , on peut introduire un changement de variable.

Soient $w = \lambda v$ et $z = \lambda y$. Alors si (u,v) est la décomposition proximale parametrée de (x,y) sur le graphe de T, (u,w) est la décomposition proximale de (x,z) sur le graphe de $\lambda T$. Par conséquent, grâce au théorème 3-6-5, nous pouvons déduire que la suite $\{(x^n,z^n)\}$ converge vers un point dans $A \times B \cap Gr(\lambda T)$. Ce qui implique que la suite $\{(x^n,y^n)\}$ converge vers une solution du problème (3-6-1).

D'autre part, on peut montrer que l'algorithme (ADP2G) est équivalent à la version de l'algorithme de l'inverse partiel (avec $\lambda_n = 1 \ \forall n$), faisant intervenir un facteur d'échelle

décrite par Spingarn(1985) dans sa présentation de la méthode proximale des multiplicateurs.

Pour analyser l'influence du paramètre $\lambda$ sur le rapport de convergence de l'algorithme (ADP2G), nous considérons le cas où l'opérateur T est fortement monotone et Lipschitzien.

**Théorème 3-6-8 :**

Si T est fortement monotone de module $\rho$ et Lipschitzien de rapport L, la convergence de la suite $\{(x^n,z^n)\}$, générée par l'algorithme (ADP2G), avec $z^n=\lambda y^n$, est linéaire de rapport :

$$r(\lambda) = \sqrt{1-\frac{2\lambda\rho}{(1+\lambda L)^2}}$$

**Preuve:**

Soit $(x^*,z^*)$ un point d'accumulation de la suite $\{(x^n,z^n)\}$. C'est donc un point fixe de l'application $\mathcal{F}_\lambda$ associée à la décomposition proximale sur le graphe de l'opérateur $\lambda T$. Puisque $(u_n,v_n)$ est la décomposition proximale de $(x^n,z^n)$ sur le graphe de $\lambda T$, on sait, à partir de la relation (3-6-6) précédente, que

$$||(u_n-x^*,v_n-z^*)||_X^2 \leq \{1-2\lambda\rho/[(1+\lambda L)^2]\}[||(x^n-x^*,z^n-z^*)||_X]^2.$$

Par conséquent, après projection sur AxB, on obtient le résultat cherché. ∎

Notons que du fait que $L\geq\rho$, le rapport de convergence r ci-dessus est plus petit que 1. On déduit facilement la valeur optimale théorique $\lambda^*$, du paramètre $\lambda$.

$$\lambda^* = \frac{1}{L} \text{ et } r(\lambda^*) = \sqrt{1-\frac{\rho}{2L}}$$

Dans le chapitre 5, nous allons voir que les résultats que nous avons obtenus dans cette section sont confirmés par les tests numériques que nous avons effectués.

Il est intéressant d'analyser le comportement de la suite $\{(x^k,y^k)\}$ et de chercher les valeurs du paramètre d'échelle pour lesquelles la suite est engendrée par un opérateur contractant. Pour être plus précis, nous considèrons les opérateurs $\mathcal{G}_\lambda$ et $\mathcal{H}_\lambda$, qui engendrent les suites $\{(x^k,z^k)\}$ et $\{(x^k,y^k)\}$, respectivement. Alors, si $\mathcal{D}_\lambda$ est l'opérateur défini par

$$\mathcal{D}_\lambda(x,y) = (x,\lambda y)$$

on a

$$\mathcal{H}_\lambda = \mathcal{D}_\lambda^{-1} \circ \mathcal{G}_\lambda \circ \mathcal{D}_\lambda$$

puisque

$$(x^k,y^k) = \mathcal{D}_\lambda^{-1}(x^k,z^k),$$

on sait déjà que la suite $\{(x^k,y^k)\}$ converge lorsque $\{(x^k,z^k)\}$ converge. Une démonstration directe (n'utilisant pas les propriétés de l'inverse partiel) de ce fait semble assez difficile. La raison en est que l'opérateur $\mathcal{H}_\lambda$ n'est pas forcément contractant, pour tout $\lambda$. Ci-dessous, nous étudions les conditions sur $\lambda$ pour que ce dernier opérateur soit une contraction, dans le cas fortement monotone.

D'après le théorème 3-6-2, on sait que $\mathcal{H}_\lambda$ est une contraction pour $\lambda = 1$. Le théorème qui suit montre que $\mathcal{H}_\lambda$ reste contractant si $\lambda$ reste dans un certain intervalle contenant 1 :

**Théorème 3-6-9 :**

Soit T un opérateur fortement monotone de module $\rho$. Alors si $\lambda \in [1,\rho+\sqrt{1+\rho^2})$, l'opérateur $\mathcal{H}_\lambda$ est contractant.

**Preuve :**

Soient (x,y) et (x',y') deux éléments de AxB et (u,v) et (u',v') leurs images respectives par l'opérateur $\mathcal{H}_\lambda$, alors

$$u = (I+\lambda T)^{-1}(x+\lambda y) \text{ et } u' = (I+\lambda T)^{-1}(x'+\lambda y')$$

$$v = (x+\lambda y - u)/\lambda \text{ et } v' = (x'+\lambda y' - u')/\lambda$$

donc on a

$$\|\mathcal{H}_\lambda(x,y) - \mathcal{H}_\lambda(x',y')\|^2_{AxB} = \|(u-u',\lambda^{-1}(x-x'+\lambda(y-y')-u+u'))\|^2_{AxB}$$

$$= \|u-u'\|^2+\lambda^{-2}\|x-x'\|^2+\|y-y'\|^2+\lambda^{-2}\|u-u'\|^2-2\,\lambda^{-1}\langle(x+\lambda y)-(x'+\lambda y'),u-u'\rangle$$

En utilisant la relation (3-6-6), nous obtenons l'inéquation suivante :

$$\|\mathcal{H}_\lambda(x,y) - \mathcal{H}_\lambda(x',y')\|^2_{AxB} \le \lambda^{-2}\|x-x'\|^2+\|y-y'\|^2 + \frac{\lambda^2-2\lambda\rho-1}{\lambda^2}\|u-u'\|^2$$

En utilisant la propriété de Lipschitz, on obtient :

$$\|\mathcal{H}_\lambda(x,y) - \mathcal{H}_\lambda(x',y')\|^2_{AxB} \le \lambda^{-2}\left(1+\frac{\lambda^2-2\lambda\rho-1}{(1+\lambda L)^2}\right)(\|x-x'\|^2+\lambda^2\|y-y'\|^2)$$

Par conséquent, une condition suffisante pour que l'opérateur $\mathcal{H}_\lambda$ soit contractant est :

$$\lambda > 1 \text{ et } \Delta(\lambda) = \lambda^2 - 2\lambda\rho - 1 < 0$$

Cette condition ne dépend pas explicitement de la constante de Lipschitz. Mais en réalité, on peut dire qu'elle en dépend relativement du moment que $0 < \rho < L$. On a $\Delta(1) = -2\rho < 0$ ; et donc l'intervalle décrit est :

$$\lambda \in [1,\rho+\sqrt{1+\rho^2})$$

## 3-7 Quelques équivalences entre algorithmes

La relation entre l'algorithme de l'inverse partiel et celui de Douglas-Rachford peut être expliquée de la manière suivante, directement inspirée du travail de Lawrence et Spingarn(1986). Plus tard, elle a été expliquée par Eckstein(1989).

Reprenons les applications $\alpha$, $\beta$ et $\gamma$, définies au Chapitre1 (cf §1-2-2), et qui réalisent le lien entre les trois classes d'opérateurs $\mathcal{M}on(X)$, $\mathcal{N}onexp(X)$ et $\mathcal{P}rox(X)$. Nous rappelons que

$$\alpha : \mathcal{P}rox(X) \to \mathcal{N}exp(X)$$
$$P \mapsto (2P-I)$$

$$\gamma : \mathcal{M}on(X) \to \mathcal{P}rox(X)$$
$$T \mapsto (I+T)^{-1}$$

et $\beta = \alpha o \gamma$

Point par point, on traduit ces correspondances comme suit :

$$\alpha : (x,y) \mapsto (x, 2y - x) \text{ et } \beta : (x, y) \mapsto (x + y, y - x).$$

Soient les opérations de composition suivantes :

$$P_1 * P_2 = \alpha^{-1}(\alpha(P_1) o\, \alpha(P_2))$$

$*$ est donc l'opération obtenue en composant deux opérateurs proximaux $P_1$ et $P_2$ à travers leurs images non expansives respectives. De même

$$T_1 \Theta T_2 = \beta^{-1}(\beta(T_1) o\, \beta(T_2))$$

est l'opérateur monotone obtenu en composant deux opérateurs monotones $T_1$ et $T_2$ à travers leurs images non expansives respectives. Un calcul élémentaire montre que, si $P_1$ et $P_2$ sont les résolvantes respectives des opérateurs $T_1$et $T_2$, alors $P = P_1 * P_2$ est la résolvante de $T = T_1 \Theta T_2$.

On peut voir aussi, à partir de ces définitions que l'opération " * " peut être interpretée comme suit :

$$P_1 * P_2 = P_1 o (2P_2 - I) + (I - P_2)$$

On retrouve ainsi l'opérateur associé à l'itération de l'algorithme de **Douglas-Rachford**, cf Lions et Mercier(1979). Nous remarquons aussi que l'opérateur non expansif $\alpha(P_1)o\alpha(P_2)$ est l'opérateur associé à l'itération de l'algorithme de Peaceman-Rachford.

Grâce à la remarque 3-4-0 (cf §3-4-1) on montre que lorsque $T_1$ est le sous-différentiel de la fonction indicatrice d'un sous-espace vectoriel A, alors $T_1 \Theta T_2 = (T_1)_A$. Nous allons résumer toutes ces remarques dans la proposition suivante :

**Proposition 3-7-1**

Soient $T_1$ et $T_2$ deux opérateurs maximaux monotones sur X. L'opérateur associé à l'itération de l'algorithme de Douglas-Rachford

$$P = P_1 \circ (2P_2 - I) + (I - P_2),$$

où

$$P_1 = (I + \lambda T_1)^{-1} \text{ et } P_2 = (I + \lambda T_2)^{-1},$$

est un opérateur proximal. En plus

$$P = (I + T)^{-1} \text{ où } T = \lambda T_1 \Theta \lambda T_2 .$$

De plus, si

$$Gr(T_1) = A \times A^{\perp} \text{ et } T_2 = T$$

Alors

$$P = (I + (\lambda T)_A)^{-1}$$

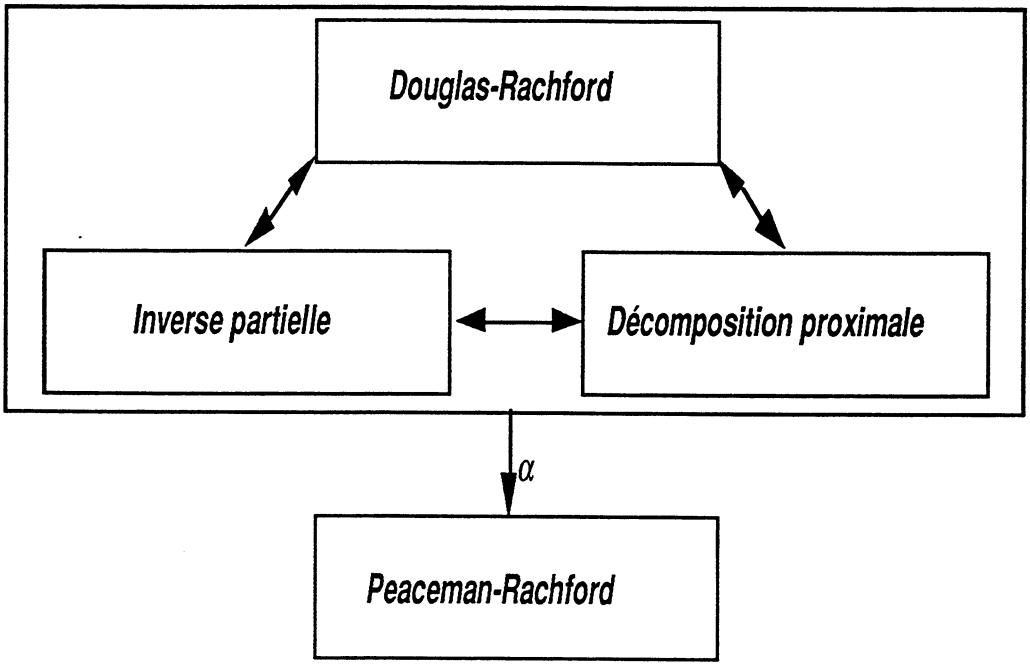
Dans ce cas l'itération de **Douglas-Rachford** est exactement celle de la **décomposition proximale (inverse partiel)**.

Remarquons que

$$(I + (\lambda T)_A)^{-1} \neq (I + \lambda (T_A))^{-1}$$

En effet, l'itération générée par l'opérateur $(I + (\lambda T)_A)^{-1}$ est implémentable pour tout $\lambda > 0$. Par contre l'itération générée par $(I + \lambda (T_A))^{-1}$ ne marche que pour $\lambda = 1$.

Nous concluons par le schéma explicatif suivant :


Douglas-Rachford
Inverse partielle
Décomposition proximale
α
Peaceman-Rachford

# Chapitre IV

# Méthodes de décomposition

## 4-1 Introduction

Au cours de ces trente dernières années, on a pu constater un grand effort de recherche aussi bien dans le domaine théorique que dans le domaine d'analyse numérique pour les méthodes de décomposition. En effet, la conjonction des préoccupations dans un certain nombre de domaines des mathématiques appliquées comme en Automatique, Recherche opérationnelle, Econométrie etc, et l'évolution rapide du matériel informatique vers des calculateurs très puissants, notamment les calculateurs parallèles, ont été des facteurs décisifs dans l'évolution de la "théorie" de la décomposition des *problèmes (systèmes) complexes* (problèmes de grandes taille).

De manière très générale, ce que nous entendons par la décomposition des problèmes d'optimisation provient d'un souci algorithmique. On a, par exemple, un gros problème à résoudre, numériquement, et on cherche tout naturellement à le remplacer par une série de problèmes de taille plus restreinte, particulièrement si on dispose de possibilités de parallélisation.

Un problème complexe peut être un problème avec un grand nombre de variables et/ou de contraintes, comme il peut être un problème avec peu de variables, mais dont la résolution nécessite un grand temps de calcul. C'est le cas, par exemple, d'un système décrit par une E.D.P, discrètisée dans le temps et l'espace en milliers de points. Citons aussi une autre source de difficultés : l'hétérogénéité. Un système hétérogène étant formé d'un ensemble de sous-systèmes interconnectés qui ne sont pas de même nature. On est donc tenté d'isoler ces sous-systèmes au niveau de la résolution afin de traiter chacun d'entre eux par une technique qui lui est bien adaptée. A priori, ces techniques peuvent être différentes.

Schématiquement, l'investissement dans les méthodes de décomposition a été motivé par plusieurs points que nous citons brièvement ci-dessous :

*Motivation 1* - Réduire la dimension d'un problème de "grande dimension" lorsque le nombre de variables et/ou de contraintes est trop grand, afin de se ramener à la résolution de sous-problèmes de dimensions inférieures. Ce fut la motivation principale des méthodes classiques de décomposition. (par ex. Dantzig et Wolfe(1961), Lasdon(1970), ...)

*Motivation 2*- Partitionner un problème hétérogène en le décomposant en sous-problèmes homogènes. (Benders(1962), Lasdon(1970)).

*Motivation 3*- Décentraliser un problème en le décomposant en sous-problèmes indépendants, avec leurs données propres, et qui sont capables de trouver leurs propres solutions optimales, cf Bensoussan et al dans Lions et Marchouk(1974). En termes économiques, considérons n centres de décision, et supposons que l'on ait défini un problème global. La décentralisation consiste en ceci : Est-il possible de résoudre ce problème global en laissant les centres de décision résoudre des problèmes locaux (c'est à dire en ne faisant intervenir que les décisions individuelles et en particulier sans contraintes globales)

*Motivation 4-* Paralléliser un problème : le décomposer en sous-problèmes "indépendants" que nous pouvons résoudre à chaque itération en parallèle avec des communications (liaisons) éventuelles qui servent de moyen d'échange d'informations (couplage) entre sous-systèmes. Actuellement, la plupart des méthodes d'optimisation sont orientées dans cet axe grâce à l'évolution rapide du matériel informatique, dans le domaine du parallélisme.

C'est dans le sens de la parallélisation que nous avons orienté une bonne partie de nos investigations sur les méthodes de décomposition. Plusieurs auteurs se sont intéressés à cet aspect, notamment Bertsekas et Tsitsiklis(1989).

*Motivation 5-* Hiérarchiser une structure décisionnelle avec un ou plusieurs niveaux d'échange. Chaque niveau comporte une unité de contrôle qui assure la coordination entre les sous-systèmes du niveau inférieur, cf Titli(1975).

En fait, les premières idées de la décomposition en programmation mathématique semblent être apparues vers 1960. En programmation linéaire, elles ont débuté avec le principe de décomposition de Dantzig et Wolfe(1961), Benders(1962), ... Et en économétrie avec les travaux de Arrow et Hurwicz(1959). Ultérieurement sont apparus des travaux en programmation convexe, cf Lasdon(1970). Puis, à partir de 1970, un certain nombre d'ouvrages ont été publiés, (Lasdon(1970), Wismer(1970), Lions et Marchouk [LM ],...).

En général, les problèmes (linéaires) de grande taille que l'on rencontre dans la pratique bénéficient des propriétés suivantes :
- Beaucoup de variables,
- Beaucoup de contraintes,
- Peu de variables dans chaque contrainte et chaque variable apparaît dans peu de contraintes.

On dit alors que le problème est faiblement couplé.

**Exemple 4-1-1 :**
*Problèmes structurés par blocs de variables et de contraintes.*
Soit le programme linéaire

$$\begin{aligned} &\text{Min : } z = cx \\ &Ax = b \qquad\qquad\qquad\qquad (\mathbf{P}) \\ &x \geq 0 \end{aligned}$$

où l'on suppose que le nombre n de variables est très largement supérieur au nombre, m, de contraintes couplantes (contraintes où apparaissent plus d'une variable) et que la taille de la matrice A est tellement élevée qu'elle ne peut être stockée entièrement dans la mémoire du calculateur dont on dispose. Ce problème (P) est donc très difficile à exploiter dans une approche globale. On observe tout de même que dans les problèmes, que l'on rencontre en pratique, il est excessivement rare de rencontrer des programmes linéaires de très grande dimension sans structure particulière.

Les matrices des contraintes de ces programmes sont en général "très creuses". Par ailleurs, les termes non nuls de ces matrices ne sont pas disposés de façon quelconque.

On a par conséquent des problèmes faiblement couplés.

Ci-dessous trois figures montrant des structures de matrices de contraintes qui sont rencontrées dans les applications.

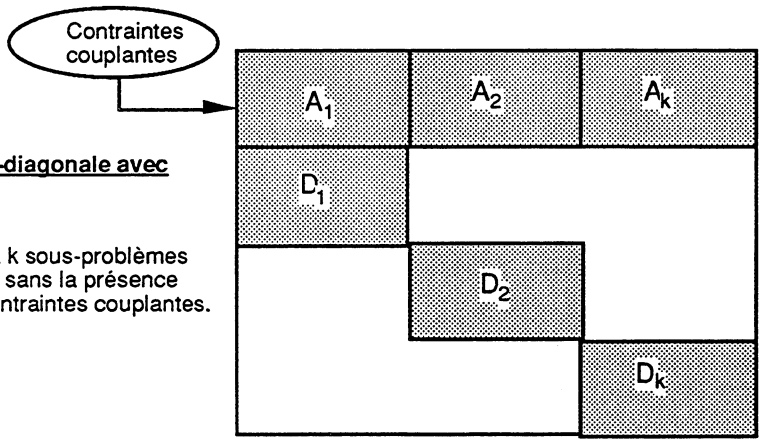



*Figure 1:* **Structure Bloc-diagonale avec contraintes couplantes.**

Cas de k blocs associés à k sous-problèmes qui seraient indépendants sans la présence d'un certain nombre de contraintes couplantes.

Bloc non nul

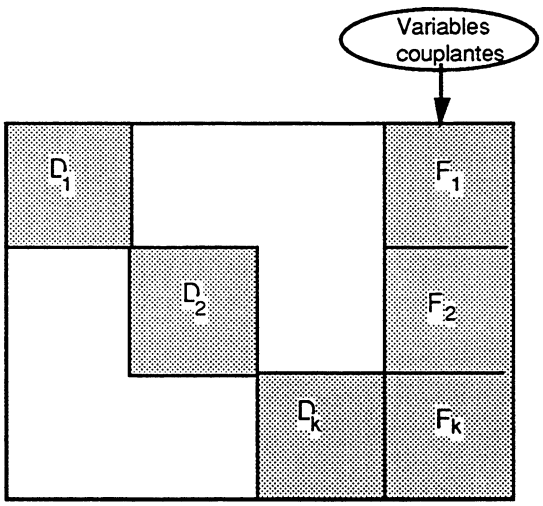
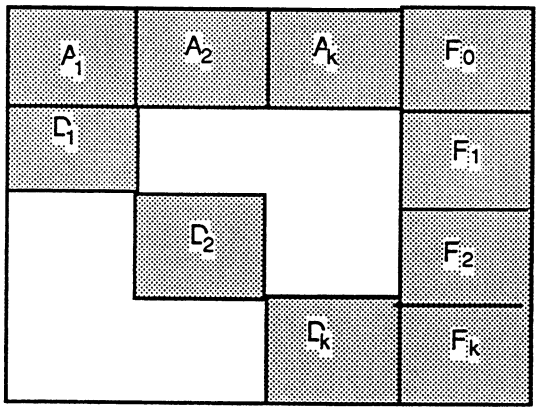



*Figure 2:* **Structure Bloc-diagonale avec variables couplantes.**

Cas de k blocs qui seraient indépendants sans la présence d'un certain nombre de variables couplantes

Bloc non nul

*Figure 3:* Structure Bloc-diagonale avec contraintes et variables couplantes.




Cas un peu plus complexe où il y a à la fois des variables et des contraintes couplantes

Bloc non nul

Nous remarquons qu'une approche par décomposition comprend en général trois phases dont une est optionnelle.

***Phase 1*** : *Analyse structurale* du problème ; elle consiste à identifier la structure du problème. Plusieurs structures sont possibles et l'on choisit celle qui a les meilleures propriétés et s'adapte le mieux au modèle particulier qu'on traite et au matériel dont on dispose. Par exemple on peut chercher une structure qui nous conduit à un découplage maximal des sous-systèmes.

***Phase 2*** : *Introduction des variables de couplage (interactions):* Cette phase, bien qu'optionnelle, est souvent nécessaire. Elle consiste soit à

-(i) Faire des copies de certaines variables, ou

-(ii) Ajouter certaines autres variables.

Le but étant de se ramener à une structure séparable (sous-problèmes indépendants), ou faiblement séparable. Plus généralement, on se ramène après cette phase intermédiaire à la résolution d'un problème (P) sur un espace produit.

**Exemple 4-1-2**

Soit le problème (P') suivant :

$$(P^0) \qquad \begin{cases} \text{Min} \sum_{i=1}^{n} f_i(x) \\ \sum_{i=1}^{n} g_i(x) \leq 0 \end{cases}$$

La manipulation proposée consiste à faire n copies de la variable x afin de faire apparaître la séparabilité de la fonction objectif. Le problème équivalent obtenu est :

$$(P^1) \qquad \begin{cases} \text{Min} \sum_{i=1}^{n} f_i(x_i) \\ \sum_{i=1}^{n} g_i(x_i) \leq 0 \\ x_1 = \ldots = x_n \end{cases}$$

Et pour découpler les contraintes, afin d'obtenir une faible séparabilité, on ajoute des perturbations. Le problème équivalent obtenu est le suivant :

$$(P^2) \qquad \begin{cases} \text{Min} \sum_{i=1}^{n} f_i(x_i) \\ g_i(x_i) \le y_i \\ \sum_{i=1}^{n} y_i = 0 \\ x_1 = \ldots = x_n \end{cases}$$

Le nouvel espace de travail est $X^n x \mathbb{R}^n$ et les sous problèmes obtenus sont de la forme :

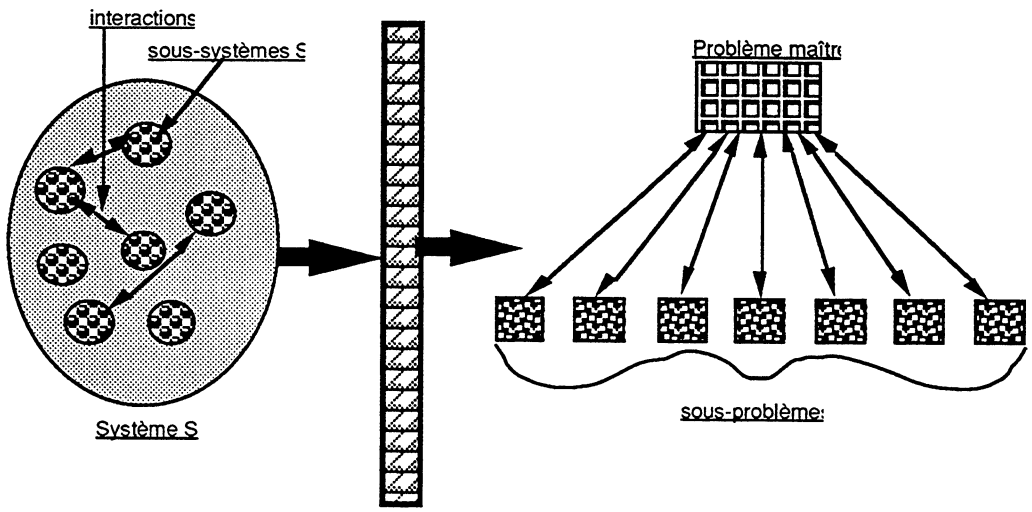
$$(P_i) \qquad \begin{cases} \text{Min } f_i(x_i) \\ g_i(x_i) \le y_i \end{cases}$$

Le couplage entre les sous-problème est donné par :

$$(C) \qquad \begin{cases} \sum_{i=1}^{n} y_i = 0 \\ x_1 = \ldots = x_n \end{cases}$$

***Phase 3*** : *Résolution* : Elle consiste à mettre en évidence un algorithme qui est adapté au problème. Elle est généralement itérative. Cette phase nous permet d'organiser les calculs autour de la structure du système. Les sous-problèmes peuvent alors être traités en séquentiel ou parallèle. Les sous-systèmes peuvent être supervisés par un niveau de coordination qui tient compte du couplage éventuel entre les sous-systèmes, par l'intermédiaire d'un certain nombre de paramètres de coordination.

Dans la figure suivante, nous donnons un exemple de schéma de système avant et après les trois étapes précédentes.




Grâce à ces trois phases que nous venons d'énumérer, on arrive à remplacer la résolution du problème global par celle de plusieurs sous-problèmes moins difficiles et souvent indépendants, qu'on note ($P_i$).

Ces sous-problèmes peuvent être indépendants, dans le sens où chacun d'entre eux possède ses propres données qui lui permettent d'identifier sa propre solution.

Dans le cas contraire, les sous-systèmes $(P_i)$ peuvent dépendre de certains paramètres dits de coordination. Si les suites $\{(x_i^k, y_i^k)\}$, itérées pour les sous-problèmes $(P_i)$, convergent vers des points $\{(x_i^*, y_i^*)\}$, on est amené à se poser plusieurs questions : Que peut-on dire du point

$$(x^*, y^*) = (x_1^*, x_2^*, \dots, x_n^*, y_1^*, y_2^*, \dots, y_n^*) ?$$

est-il une solution du problème global (P) ?

Comment reconnaître l'optimalité des solutions ?

Et dans le cas de la décentralisation, chaque sous-problème $(P_i)$, est-il capable de reconnaître l'optimalité globale de la solution locale $(x^*, y^*)$ c.a.d, telle que $(x^*, y^*) = (x_1^*, x_2^*, \dots, x_n^*, x_1^*, y_2^*, \dots, y_n^*)$ soit solution de (P) ?

Dans ce travail, nous n'avons pas traité la phase 1. Nous nous sommes principalement interessé à la phase 2 dont nous présenterons les idées principales. Nous présenterons aussi quelques algorithmes étudiés pour la phase 3 de résolution et qui utilisent les concepts des méthodes proximales présentées au Chapitre précédent.

## *4-2- Phase 2 : Manipulation des problèmes.*

### 4-2-1 Séparabilité

**Définition 4-2-1 :**

Si $X=X_1 x\ X_2 x\ ...\ x\ X_m$ est un espace de Hilbert.

On dit que la fonction F définie sur X est *séparable* par rapport à la partition $X_1, ..., X_m$ de X si et seulement si F est une combinaison linéaire de fonctions définies sur les espaces $X_i$ $(1 \leq i \leq m)$. Par exemple

$$F(x) = \sum_{i=1}^{m} F_i(x_i)$$

On dit qu'un problème de minimisation d'une fonction F est *séparable* par rapport à la partition $X_1, ..., X_n$, si la fonction F est séparable par rapport à la même combinaison (et qu' il n'y a pas de contraintes de couplage.

**Intérêt de la séparabilité**

Soit le problème séparable (P1) suivant :

$$\begin{aligned} &\textbf{min} \sum_{i=1}^{m} f_i(x_i) \\ &x=(x_1,..., x_n),\ x_i \in S_i,\ (1 \leq i \leq m) \\ &S_i \subset X_i,\ (1 \leq i \leq m) \end{aligned} \qquad \textbf{(P1)}$$

Il est évident que pour résoudre le problème (P*) il suffit de résoudre séparément les n sous-problèmes $(P_i)$, $(1 \leq i \leq m)$, suivants :

$$\begin{aligned} &\min f_i(x_i) \\ &x_i \in S_i \end{aligned} \qquad \textbf{(P}_i\textbf{)}$$

Le fait de se ramener à un problème séparable est d'un intérêt primordial dans la décomposition. Une situation proche où le couplage est concentré dans un certain sous-espace est appelée faible séparabilité :

**Définition 4-2-1-2 :**

Un problème est dit *faiblement séparable* s'il est de la forme (P1) avec une contrainte de couplage : $x \in A$, où A est un sous-epace de X.

Cette situation est celle à laquelle on se ramène le plus fréquemment. Et comme nous l'avons déjà signalé, c'est grâce aux manipulations de la phase 2 que nous pouvons obtenir la faible séparabilité.

Dans le paragraphe suivant, nous illustrons certaines manipulations de la phase 2 au travers d'exemples :

### 4-2-2 Types de manipulations

#### 4-2-2-1 Copies des variables

On considère le problème ci-dessous :

$$\min \sum_{i=1}^{n} f_i(x)$$
$$x \in \cap S_i \qquad \textbf{(P2)}$$

où les $S_i$ sont des sous-ensembles convexes fermés de X. Nous supposerons que leur intersection est d'intérieur non vide.

(P2) est équivalent au problème suivant :

$$\min \sum_{i=1}^{n} f_i(x_i)$$
$$z=(x_1,...,x_n)\ ;\ x_i \in S_i\ ;\ z \in A \qquad \textbf{(tP2)}$$

avec $A = \{ z \in X^n : z = (x_1, x_2, \dots, x_n)$ et $x_1 = x_2 = \dots = x_n \}$

A est un sous-espace de l'espace produit $X^n$. Il représente l'interaction entre les sous-problèmes.

Grâce à cette manipulation, appelée transformation par *copies des variables*, nous avons pu obtenir un problème faiblement séparable et plus facile à traiter. On retrouve cette manipulation dans Pierra(1984).

#### 4-2-2-2 : Restriction des variables (Approche primale)

Ce genre de manipulation se fait lorsqu'on a des contraintes couplantes (en petit nombre). Ainsi, on arrive à transformer le problème de départ en un problème équivalent, faiblement séparable (ref. littérature Lasdon(1970) Allocation des ressources).

Nous considérons le problème

$$\min \sum_{i=1}^{n} f_i(x_i)$$
$$\sum_{i=1}^{n} g_i(x_i) \leq 0$$
$$x_i \in S_i,\ i=1 \dots n \qquad \textbf{(P3)}$$

où $f_i$, $g_i$, i=1...n, sont des fonctions convexes, propres s.c.i. sur les espaces respectifs $X_i$. Il est équivalent au problème

$$\begin{cases} \min \Psi(s) \\ s=(s_1,\ldots,s_n)\in H \end{cases} \qquad \textbf{(tP3)}$$

où

$$\Psi(s)= \sum_{i=1}^{n} \Psi_i(s_i)$$

avec

$$\Psi_i(s_i) = \begin{cases} \min f_i(x_i) \\ g_i(x_i)\le s_i \ ; \ x_i\in S_i \end{cases} \quad 1\le i \le n$$

et

$$H = \left\{ (s_1,\ldots,s_n) \ ; \ \sum_{i=1}^{n} s_i = 0 \right\} \qquad (**)$$

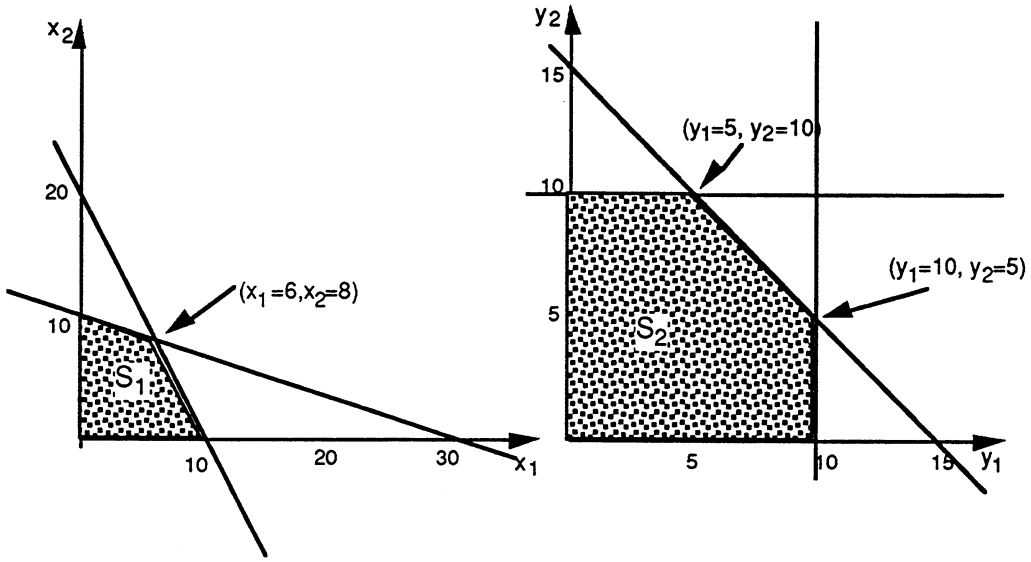
La fonction $\Psi$ est convexe semi-continue inférieurement, cf Rockafellar(1970c) §21, et séparable. Elle n'est en général pas différentiable.

**Exemple 4-3 :**

Pour illustrer le principe de décomposition que nous venons decrire, considèrons le problème suivant : "exemple tiré de Lasdon(1970)"

$$(E) \qquad \begin{cases} \min \ z = - x_1 - x_2 - 2y_1 - y_2 \\ \qquad [x_1+2x_2+2y_1+y_2\le 40] \\ x\in S_1 = \begin{cases} x_1+3x_2 & \le 30 \\ 2x_1+x_2 & \le 20 \end{cases} \\ y\in S_2 = \begin{cases} y_1 & \le 10 \\ y_2 & \le 10 \\ y_1+y_2 & \le 15 \end{cases} \end{cases}$$

et toutes les variables sont positives ou nulles. Ce problème a une seule contrainte de couplage et deux blocs. Les régions admissibles pour ces deux blocs sont représentées sur la figure ci-dessous :




Les deux sous-problèmes que nous obtenons sont :

**Sous-problème 1**

$$(E_1) \quad \begin{cases} \min \; - x_1 - x_2 \\ [x_1+2x_2 \leq u_1+20] \\ x \in S_1 \end{cases}$$

*Solution*

Si $u_1 \leq -20$ ou $u_1 \geq 2$ alors le sous-problèmes $(E_1)$ n'admet pas de solution.

Si $-20 < u_1 \leq -10$ alors la solution est $x=(u_1+20, 0)$.

Si $-10 \leq u_1 < 2$ alors la solution est $x=([20- u_1]/3, [20+2u_1]/3)$.

$$\Phi_1(u_1) = -x_1(u_1) - x_2(u_1)$$

**Sous-problème 2**

$$(E_2) \quad \begin{cases} \min \; - 2y_1 - y_2 \\ [2y_1+y_2 \leq u_2+20] \\ y \in S_2 \end{cases}$$

*Solution*

Si $u_2 \leq -20$ alors le sous-problèmes $(E_2)$ n'admet pas de solution.

Si $-20<u_2\leq 5$ alors la solution $y\in S_2\cap\{y|2y_1+y_2=u_2+20\}$.

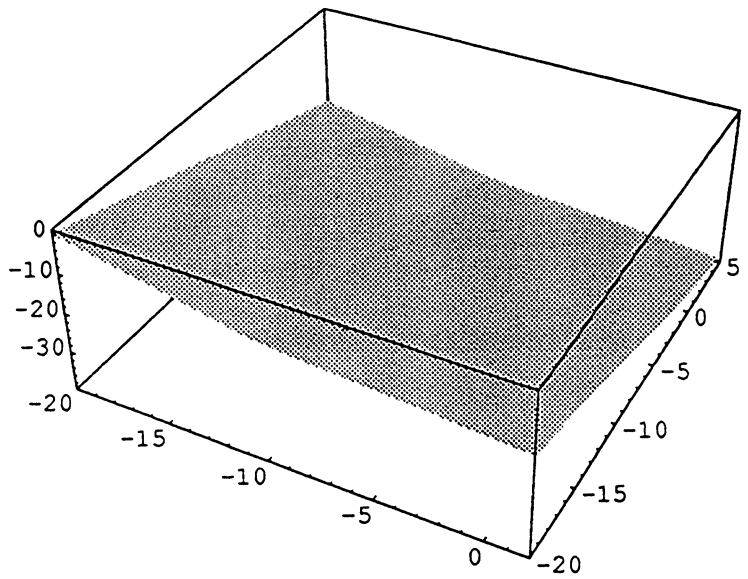
$$\Phi_2(u_2) = -(u_2 + 20)$$

$$\Phi(u_1,u_2) = \Phi_1(u_1)+\Phi_2(u_2)$$

$$\Phi(u_1,u_2) = u_2 - u_1 \text{ si } -20\leq u_1\leq -10 \text{ et } -20\leq u_2\leq 5$$

$$= (3u_2 - u_1 + 20)/3 \text{ si } -10\leq u_1\leq 2 \text{ et } -20\leq u_2\leq 5$$

Cette fonction n'est pas différentiable comme le montre la figure ci-dessous. Celle-ci a été tracée par le logiciel Matematica sur une dtation Next, nous avons changé le choix des axes dans le but de mettre en évidence la non différentiabilité de $\Phi$.




(tP3) est un problème avec des variables supplémentaires et une contrainte d'intéraction en plus, mais plus facile à traiter grâce à leur homogénéité. La finalité est de traiter n sous-problèmes, chacun prenant en compte une contrainte (ou un bloc de contraintes). L'intéraction entre ces sous-problèmes prend en compte la contrainte (**) qui est facile à traiter.

D'autres manipulations sont possibles lors de la transformation d'un problème donné. Par exemple, la restriction des variables, cf Benders(1962). Pour obtenir la séparabilité du problème, on utilise plusieurs techniques dont les techniques de linéarisation de la fonction coût, lorsqu'il n'y a pas de contrainte de couplage. Ce qui permet d'obtenir un

problème séparable, du moment que les fonctions linéaires le sont. Dans certains travaux, on trouve en plus que la linéarisation, l'ajout d'un noyau séparable : un terme régularisant séparable, cf Cohen et Zhu(1983). Un cas particulier de noyau séparable est le terme quadratique (de régularisation proximale).

### 4-2-2-3 Relaxation Lagrangienne (approche duale)

Revenons sur les problèmes de type (P3). Dans le cas convexe, une des manipulations classiques qu'on peut lui appliquer est la relaxation lagrangienne, elle le transforme en un problème Min-max de la forme :

$$\underset{u\geq 0}{\text{Max}} \left\{\Phi(u) = \sum_{i}^{n} \Phi_i(u)\right\}$$

$$\Phi_i(u) = \text{Inf}_{x\in S_i}\, f_i(x) + \langle u, g_i(x)\rangle,\ i=1 \ldots n$$

Afin de rappeler certaines propriétés de la fonction $\Phi$, nous allons supposer que n=1 donc que

$$\Phi(u) = \text{Inf}_{x\in S}\, f(x) + \langle u, g(x)\rangle$$

Cette fonction $\Phi$ (fonction duale) est concave sur tout sous-ensemble convexe de son domaine (cf. Lasdon(1970)), mais elle n'est pas forcément différentiable. Si on pose :

$$X(u) = \left\{x(u) \mid x(u) = \text{argmin}_{x\in S}\, f(x) + \langle u, g(x)\rangle\right\}$$

que l'on suppose que S est fermé et borné et que les fonctions f et g sont continues sur S, la fonction duale $\Phi$ est différentiable en un point $u^*$ si et seulement si la fonction g est constante sur $X(u^*)$. Par conséquent, si f est fortement convexe, la fonction duale $\Phi$ est différentiable, cf Lasdon(1970).

La qualité d'une méthode de décomposition dépend non seulement des propriétés de la fonction objective, notamment la convexité, la différentiabilité et la séparabilité ; mais aussi du degré de couplage entre les sous-systèmes. Par conséquent, lorsque le nombre de contrainte couplantes est trop élevé, nous devons également les partitionner.

Remarquons que, nous obtenons après la plupart des manipulations classiques des fonctions de coordination non différentiables.

## 4-3 Algorithmes de décomposition

### 4-3-1 Approches classiques

La non différentiabilité évoquée à la fin du paragraphe précédent justifie le fait que beaucoup d'algorithmes classiques de décomposition trouvent leur base théorique dans l'optimisation non différentiable. Nous allons citer brièvement à titre d'exemple certaines méthodes classiques de décomposition :

*Méthodes de sous-gradient*, :

Etudiées principalement par des chercheurs russes de l'école de Kiev dans les années 60, ces méthodes consistent à mettre à jour les paramètres de couplage dans la direction d'un sous-gradient de la fonction objectif. Ces méthodes on été étudiées par plusieurs auteurs dont Shor(1968), Minoux(1983), ...

Bien que ces méthodes soient de premier ordre (presque sous-linéaires) donc relativement lentes, elles sont couramment utilisées. En effet, elles demandent un faible coût d'implémentation et sont peu sensibles à la dimension du problème. elles génèrent des trajectoires en zigzag, de part et d'autre des surfaces de non différentiabilité de la fonction objectif.

*Les méthodes de plans de coupe* : Selon Kelley(1960), ou de linéarisation externe, cf Geoffrion(1970).

Elles sont appliquées à la maximisation d'une fonction concave, non nécessairement différentiable. Elles consistent à remplacer la fonction objectif par l'enveloppe inférieure des supports affines associés aux sous-gradients calculés antérieurement. Il est bien connu que cette méthode présente une grande instabilité numérique. Les applications les plus connues sont les algorithmes de Dantzig-Wolf(1961), et de Benders(1962)

**Remarque** : *Méthodes des faisceaux*

Introduites en 1975 par C.Lemarechal(1980) adaptées aux techniques de décomposition plus tard, cf Medhi(1987), on peut les interpréter comme une régularisation itérative de la méthode des plans de coupe. En effet, on y construit une approximation du sous-différentiel par des petits pas de type proximaux, cf Lemaréchal(1991).

Signalons qu'il n'existe pas de recette précise pour manipuler un problème donné. Selon la structure du problème (objectif et/ou contraintes), les manipulations peuvent être différentes et c'est là où réside une des difficultés de la décomposition.

### 4-3-2 Nouvelles méthodes de décomposition

#### 4-3-2-1 Introduction

La non différentiabilité de la fonction de coordination que nous avons évoquée ci-dessus, présente un des grands défauts des méthodes classiques. D'où la nécessité de proposer d'autres algorithmes qui permettent d'éviter cette non différentiabilité (par exemple en remplaçant le traitement d'un problème convexe par celui d'un problème

fortement convexe) et qui préservent la séparabilité du problème. Il a donc été question d'introduire des termes de régularisation (proximale, dans notre cas) dans les problèmes. On impose ainsi la différentiabilité de la fonction de coordination et l'unicité de la solution de chaque sous-problème.

La régularisation proximale fait partie de ces techniques, cf Rockafellar(1976a). Elle consiste effectivement à ajouter un noyau quadratique et séparable afin de transformer un problème convexe en une suite de problèmes fortement convexes (donc unicité de la solution, coordination différentiable) tout en conservant la séparabilité du problème. Plusieurs travaux ont été orientés dans ce sens, notamment ceux de Spingarn(1985), Pierra(1984), Mouallif et al(1991), Eckstein(1989), Mahey et al(1992c), etc.

Dernièrement, de nouvelles approches, ayant la même idée générale, sont apparues. Elles consistent à remplacer le terme quadratique, de la régularisation proximale, par une distance de Bregmann, cf Teboulle(1992).

On retrouve cette idée d'ajouter un noyau, mais plus générale, dans les travaux de Cohen et Zhu(1983). Selon la forme du noyau que l'on ajoute, on retrouve des méthodes déjà existantes, par exemple l'algorithme de sous-gradient.

### 4-3-2-2 Exemple de situations et sous-problèmes obtenus

Soient $C_i$, i=1, ..., n, des sous-ensembles de X convexes fermés tels que

$$ri(C_1 \cap \ ... \cap C_n) \neq \varnothing$$

le problème est de trouver un point x tel que

$$x \in C_1 \cap \ ... \cap C_n \qquad (P^{1,1})$$

ce problème est équivalent au problème suivant qui est un cas particulier de (P1) :

$$\min f(x) \qquad (P^{1,2})$$

où

$$f(x) = \sum_{i=1}^{n} \chi_{C_i}(x)$$

Ce dernier, équivalent à son tour à :

$$\min \sum_{i=1}^{n} \chi_{C_i}(x_i) \qquad (P^{1,3})$$
$$x \in A=\{x=(x_1,\cdots, x_n) \mid x_1 = x_2 = ... = x_n\}$$

L'application de l'algorithme (PR) conduit aux sous-problèmes :

$$\begin{cases} \min \ ||x - x_i^k||^2 \\ x \in C_i \end{cases} \qquad (P^{1,2})_i$$

dont la solution unique est :

$$x_i^{k+\frac{1}{2}} = \text{Proj}_{C_i}(x_i^k)$$

L'itéré suivant, $x^{k+1} = (x_1^{k+1},...,x_n^{k+1})$ est donné par :

$$x^{k+1} = \text{Proj}_A(x_1^{k+\frac{1}{2}}, \ldots, x_n^{k+\frac{1}{2}})$$

Dans ce cas particulier l'algorithme converge vers la solution du problème.

Dans la suite de ce Chapitre, nous allons reprendre certains algorithmes, étudiés au Chapitre 3 dans un cas général. Nous remarquerons que selon les manipulations que nous appliquons à un problème donné, on peut obtenir des versions différentes d'un algorithme général (décrit par exemple au Chapitre 3).`

### 4-3-2-3 Variantes associées à l'algorithme proximal projeté (PR)

Dans ce paragraphe nous nous intéressons aux problèmes du type :

$$\min f(x) \; ; \; x \in A. \qquad (P_4)$$

Où A est un sous-espace vectoriel de X et f une fonction propre, convexe, séparable et s.c.i sur X, telle que

$$\text{ri}(\text{Dom } f) \cap A \neq \varnothing.$$

L'algorithme (PR), déjà étudié au Chapitre 3 précédent, appliqué au problème $(P_4)$, nous permet d'obtenir la version fonctionnelle que nous énonçons ci-dessous :

**L'algorithme PR1**

*A l'itération k,*

soit $x^k \in A$

$$x^{k+\frac{1}{2}} = \arg\min f(x) + \frac{1}{2\lambda} \| x - x^k \|^2$$

où $\lambda$ est un réel positif.

$$x^{k+1} = \text{Proj}_A(x^{k+\frac{1}{2}}) .$$

Nous rappelons que la suite $\{x^k\}$ générée par cet algorithme converge vers un point $x_\lambda$ qui est solution du problème régularisé :

$$\min f_\lambda(x) \; ; \; x \in A. \qquad (P^\lambda_4)$$

Cette solution n'est généralement pas une solution du problème ($P_4$). Par contre la suite $\{x_\lambda\}$, converge vers un point $x^*$ qui est solution du problème ($P_4$) lorsque $\lambda$ tend vers zéro. D'où l'idée d'un algorithme à deux étapes, notamment l'algorithme (PRC) que nous avons décrit au Chapitre précédent. Pour le lecteur désirant avoir plus de détails sur l'analyse de la convergence de cet algorithme, nous conseillons de se référer au travail de Mahey et Pham dans Mahey et al(1992c). Pratiquement, cet algorithme génère à chaque itération une suite, par l'algorithme (PR), avec un paramètre $\lambda$ constant. Ce paramètre est réduit lorsqu'on passe à l'itération suivante. Cette double suite converge vers une solution du problème ($P_4$) s'il en admet une.

Nous avons testé cet algorithme et l'avons appliqué à certains modèles pris comme tests de référence, notamment les problèmes de localisation.

#### 4-3-2-3-1 Remarques

**1-** La convergence des algorithmes (PR) et (PRC) est très sensible au comportement de la suite des paramètres $\lambda_k$.

**2-** La nécessité de réduire le paramètre $\lambda$ vers zéro a un effet négatif sur la performance de la méthode et par conséquent sur le comportement de la double suite. Cette dernière, à partir d'une certaine valeur (petite) de $\lambda$, n'est plus sensible à ce dernier paramètre et devient simple, dans le sens que la recherche de l'itéré suivant par l'algorithme (APPC) se réduit à une recherche par l'algorithme (PPC). Par contre, cet algorithme bénéficie d'une grande stabilité numérique.

**3-** On aurait pu remplacer $\lambda$ par $\lambda_k$ dans l'algorithme, avec $\lambda_k$ qui tend vers zéro afin d'éviter une des deux étapes. Le problème est qu'on n'obtient qu'une convergence érgodique en gardant les mêmes défauts que ceux cités dans la remarque 2.

#### 4-3-2-3-2 Algorithme dual (au sens de Fenchel) associé à PR

L'algorithme dual associé à l'algorithme PR peut être facilement obtenu. Pour cela, nous allons considérer le problème dual au sens de Fenchel du problème ($P_4$) En effet, ce dernier peut s'écrire sous la forme du problème sans contrainte suivant :

$$\min \{f(x) + \chi_A(x)\}$$

f et $\chi_A$ sont dans $\Gamma^\circ(X)$, Le dual de ce problème est :

$$\min \{f^*(-y) + \chi^*_A(-y)\}$$

où $f^*$ est la fonction conjuguée de Fenchel de f. Ce problème peut être réécrit de façon équivalente, comme suit :

$$\inf_{y \in \mathbb{R}^n} \left\{ f^*(-y) + \sup_{x \in A} \langle x,y \rangle \right\}$$

Or A est un sous-espace vectoriel. Dès lors, on peut voir facilement la relation suivante :

$$\sup_{x \in A} \langle x,y \rangle = \chi_B(y)$$

où B est le sous-espace orthogonal de A. Par conséquent, nous obtenons la formulation duale suivante :

$$\min f^*(-y) \; ; \; y \in B. \qquad (D_4)$$

Par le théorème de dualité de Fenchel (cf R.T. Rockafellar [R ] théorème 31.1., p. 32), sous l'hypothèse considérée,

$$ri(\mathrm{Dom}\, f) \cap A \neq \emptyset,$$

la valeur optimale de la fonction objectif primale est égale à l'opposé de la valeur optimale de la fonction objectif duale.

Remarquons que le problème $(D_4)$ est du même type que le problème $(P_4)$. On peut donc lui appliquer l'algorithme PR1 et nous obtenons l'itération suivante :

*A l'itération k,*

$$\text{soit} \quad y^k \in B$$

$$y^{k+\frac{1}{2}} = \arg\min f^*(-y) + \frac{1}{2\lambda} \| -y+y^k \|^2 \qquad (d1)$$

où $\lambda$ est un réel positif.

$$y^{k+1} = \mathrm{Proj}_B(y^{k+\frac{1}{2}})$$

Nous voudrions pouvoir modifier l'itération (d1) de sorte que la fonction manipulée soit la fonction de départ f et non sa conjuguée f*.

La condition d'optimalité issue de (d1) est :

$$0 \in \partial(f^*(-.) + \frac{1}{2}\|-. + y^k\|^2)(y^{k+1/2})$$

c'est à dire

$$\frac{1}{\lambda}(y^{k+1/2} - y^k) \in \partial f^*(-y^{k+1/2})$$

De manière équivalente cf Rockafellar(1976b) th. 23.5, p. 218, nous pouvons écrire :

$$-y^{k+1/2} \in \partial f(\frac{1}{\lambda}(y^{k+1/2} - y^k))$$

ou bien

$$0 \in \partial f(\frac{1}{\lambda}(y^{k+1/2} - y^k)) + y^{k+1/2}$$

ce que nous pouvons exprimer de la manière suivante :

$$y^{k+1/2} = \underset{y \in X}{\operatorname{argmin}} \left\{ f(\frac{1}{\lambda}(y-y^k) + \frac{1}{2\lambda}\|y\|^2 \right\}$$

Dès lors, grâce à cette relation, nous pouvons modifier l'itération (d1) et obtenir l'algorithme suivant :

**L'algorithme PR2**

*A l'itération k,*

soit $y^k \in B$

**(d3)**

$$y^{k+\frac{1}{2}} = \arg\min f(\frac{y - y^k}{\lambda}) + \frac{1}{2\lambda}\|y\|^2$$

où $\lambda$ est un réel positif.

$$y^{k+1} = \mathrm{Proj}_B(y^{k+\frac{1}{2}})$$

### 4-3-2-3-3 Algorithme primal-dual

Considèrons maintenant une version primal-duale de ($P_4$) où les deux sous-espaces A et B apparaissent de façon redondante :

$$\max_{y \in B} \min_{x \in A} \{f(x) - \langle x,y \rangle\} \qquad (S_4)$$

en d'autre terme, nous recherchons sur AxB un point-selle de la fonction

$$(x,y) \mapsto f(x) - \langle x,y \rangle$$

Nous pouvons alors proposer un nouvel algorithme en appliquant la méthode proximale projetée à ($S_4$).

Dans un premier temps, nous pouvons écrire le problème ($S_4$) sous une forme équivalente, qui consiste à chercher le zéro de la somme de deux opérateurs maximaux monotones. Ensuite, nous le régulariserons, comme dans le Chapitre 3 et le réécrirons sous la forme d'un problème d'optimisation noté ($S_4^{\lambda}$). Dans un deuxième temps, nous étudierons l'itération associée au problème régularisé et nous écrirons l'algorithme primal-dual.

Notons

$$L(x,y) = f(x) - \langle x,y \rangle \text{ ; sur } XxX$$

Cette fonction est convexe par rapport à la variable x et concave par rapport à y (convexe-concave). De plus, elle est s.c.i en x pour tout y et est s.c.s en y pour tout x. Le problème $(S_4)$ peut donc être écrit sous la forme suivante :

$$\max_{y\in B} \min_{x\in A} \{L(x,y) + \chi_A(x) - \chi_B(y)\} \qquad (S'_4)$$

La régularisation de ce problème consiste à ajouter un terme quadratique sur la variable x et on en retranche un sur la variable y. Le support théorique de cette manipulation se trouve dans R.T.Rockafellar [R], où est définie la notion de sous-différentiel d'un fonction convexe-concave S, noté $\partial S$, à partir duquel est construit un opérateur maximal monotone, noté $T^S$, tel qu'un point-selle de $S = L+\chi$ soit un zéro de l'un de ces deux opérateurs et inversement, cf K. Mouallif(1989). $\chi$ est définie par :

$$\chi(x,y) = \chi_A(x) - \chi_B(y)$$

Nous savons que

$$\text{ri}(\text{dom } f) \cap A \neq \varnothing.$$

donc

$$\partial(L+\chi) = \partial L + \partial\chi.$$

Par conséquent, trouver une solution (x,y) du problème $(S_4)$ donc de $(S'_4)$ revient à trouver un point (x,y) de XxX tel que

$$(0,0)\in \partial L(x,y) + \partial\chi(x,y)$$

ce qui est équivalent à trouver (x,y) tel que

$$(0,0)\in T^L(x,y) + T^\chi(x,y)$$

où $T^L$ et $T^\chi$ sont les opérateur maximaux monotones dont nous avons parlé plus haut. Ils sont définis comme suit :

$$T^\chi = \{(u,-v) \mid (u,v) \in \partial\chi(x,y)\}$$

et

$$T^L = \{(u,-v) \mid (u,v) \in \partial L(x,y)\}$$

Nous considèrons le problème régularisé, en remplaçant l'un des deux opérateurs, notamment $T^L$ par son approximation de Moreau-Yosida que nous notons ici par $(T^L)_\lambda$. Le problème régularisé est donc de trouver (x,y) dans XxX tel que

$$(0,0)\in (T^L)_\lambda(x,y) + T^\chi(x,y) \qquad (P_\lambda)$$

cette approximation $(T^L)_\lambda$ peut être écrite comme $T^{L_\lambda}$ où $L_\lambda$ est définie par :

$$L_\lambda(x,y) = \min_u \max_v \left\{ L(u,v) + \frac{1}{2\lambda}\|u-x\|^2 - \frac{1}{2\lambda}\|v-y\|^2 \right\}$$

$$= \max_u \min_v \left\{ L(u,v) + \frac{1}{2\lambda}\|u-x\|^2 - \frac{1}{2\lambda}\|v-y\|^2 \right\}$$

(pour plus de détails sur les étapes intermédiaires, cf K. Mouallif [M]).

Le problème régularisé $(P_\lambda)$ revient donc à trouver un point-selle de $L_\lambda+\chi$. L'itération qui correspond à l'algorithme proximal projeté, adapté au problème primal-dual $(P_\lambda)$ est donc donnée par :

*A l'itération k,*

soient $x^k \in A$ et $y^k \in B$

$$(x^{k+\frac{1}{2}}, y^{k+\frac{1}{2}}) = \arg\max_y \min_x \left\{ f(x) - \langle x,y\rangle + \frac{1}{2\lambda}\|x-x^k\|^2 - \frac{1}{2\lambda}\|y-y^k\|^2 \right\}$$

où $\lambda$ est un réel strictement positif.

$$(x^{k+1}, y^{k+1}) = \mathrm{Proj}_{A\times B}(x^{k+\frac{1}{2}}, y^{k+\frac{1}{2}})$$

Pour simplifier l'algorithme, nous allons le modifier de sorte que l'espace sur lequel nous projetons reste l'espace primal-dual, AxB, mais que la fonction que nous optimisons soit en x seulement : On élimine la variable duale y en commençant à optimiser par rapport à y

Pour cela, nous reprenons le problème suivant :

$$\max_y \min_x \left\{ f(x) - \langle x,y\rangle + \frac{1}{2\lambda}\|x-x^k\|^2 - \frac{1}{2\lambda}\|y-y^k\|^2 \right\} \qquad \textbf{(R)}$$

En écrivant la condition d'optimalité par rapport à y, nous obtenons

$$-x^{k+\frac{1}{2}} - \frac{1}{\lambda}(y^{k+\frac{1}{2}} - y^k) = 0$$

ou

$$y^{k+\frac{1}{2}} = -\lambda x^{k+\frac{1}{2}} + y^k$$

Ainsi, il suffit d'écrire les inégalités que vérifie le point-selle pour le problème (R) pour voir que nous pouvons le remplacer par un problème (R') équivalent. La seule modification consiste à supprimer la variable y dans (R) en tenant compte de la relation suivante :

$$y = -\lambda x + y^k$$

Par conséquent, l'itération proximale peut être remplacée par :

$$\begin{cases} x^{k+\frac{1}{2}} = \arg\min_x \left\{ f(x) - \langle x, y^k - \lambda x \rangle + \frac{1}{2\lambda}\|x - x^k\|^2 - \frac{1}{2\lambda}\|\lambda x\|^2 \right\} \\ y^{k+\frac{1}{2}} = -\lambda x^{k+\frac{1}{2}} + y^k \end{cases}$$

et par une simple transformation, nous obtenons l'itération suivante :

$$\begin{cases} x^{k+\frac{1}{2}} = \arg\min_x \left\{ f(x) + \frac{\gamma}{2\lambda}\|x - \frac{1}{\gamma}(x^k + \lambda y^k)\|^2 \right\} \\ y^{k+\frac{1}{2}} = -\lambda x^{k+\frac{1}{2}} + y^k \end{cases}$$

où $\gamma = 1+\lambda^2$ avec $\lambda$ un réel strictement positif. D'où l'algorithme suivant :

**L'algorithme PR3**

A l'itération k soit

$$(x^k, y^k) \in A \times B$$

$$x^{k+\frac{1}{2}} = \arg\min \left\{ f(x) + \frac{\gamma}{2\lambda}\|x - \frac{1}{\gamma}(x^k - \lambda y^k)\|^2 \right\}$$

où $\gamma = 1+\lambda^2$

$$y^{k+1} = y^k - \lambda x^{k+\frac{1}{2}}$$

$$(x^{k+1}, y^{k+1}) = \mathrm{Proj}_{A \times B}(x^{k+\frac{1}{2}}, y^{k+\frac{1}{2}})$$

Une application de (PR3) au cas où f est linéaire avec un modèle de décomposition par blocs est proposée dans Mahey et al(1992a). On retrouve un algorithme similaire à celui proposé par Goldstein(1986).

Nous signalons que les algorithmes que nous avons présentés (PR1, 2, 3, 4 et 5) bénéficient tous d'une grande stabilité numérique. Par contre, nous avons remarqué, à travers les expérimentation numériques, que le plus performant est le dernier (PR5). Nous allons présenter dans le paragraphe suivant une version de l'algorithme de décomposition proximale (voir Chapitre 3) adapté à un certain type de problèmes qui nous sera d'une grande utilité pour l'application de cet algorithme au problème de transport.

### 4-3-2-4 Variante associée à l'algorithme de décomposition proximale

#### 4-3-2-4-1 Préliminaires

Nous considérons ici un problème d'optimisation, général et séparable, sur un espace produit, $X = X_1 x \dots x X_p$, où $X = \mathbb{R}^n$ et $X_i = \mathbb{R}^{ni}$ ; i=1, ..., p :

$$(\mathbf{P5}) \quad \begin{cases} \text{Minimiser} \sum_{i=1}^{p} f_i(x_i) \\ \sum_{i=1}^{p} g_i(x_i) = 0 \end{cases}$$

où

$$x = (x_1, ..., x_p) \in X,$$

$$f_i : X_i \to \mathbb{R} \text{ et } g_i : X_i \to \mathbb{R}^m (=Y).$$

La fonction objectif est supposée convexe, s. c. i. et propre et les contraintes sont linéaires (pour simplifier la présentation) Le cas des contraintes non linéaires peut être traité d'une façon analogue. Pour obtenir un problème dual (faiblement) séparable, nous perturbons chaque morceau $g_i$ par un vecteur de perturbation $u_i$ tel que

$$u = (u_1, ..., u_p) \in Y^p$$

et

$$\sum_{i=1}^{p} u_i = 0$$

On définit une bifonction Fu par :

$$(Fu)(x) = \begin{cases} \sum_{i=1}^{p} f_i(x_i) & \text{si } g_i(x_i) = u_i \ i = 1, ..., p \\ +\infty & \text{sinon} \end{cases}$$

Cette bifonction est convexe (voir annexe) et elle est séparable par rapport aux variables x et u, i. e.

$$(Fu)(x) = \sum_{i=1}^{p} (F_i u_i)(x_i)$$

où

$$(F_i u_i)(x_i) = \begin{cases} f_i(x_i) \text{ si } g_i(x_i) = u_i \\ +\infty \quad \text{sinon} \end{cases}$$

Notons par F la fonction minimum perturbée définie sur $Y^p$ telle que :

$$F(u) = \text{Inf}_x (Fu)(x)$$

Il est clair que si $z^*$ est la valeur optimale du problème (P5) on a

$$\text{Inf}_{u \in A} F(u) = z^*$$

où

$$A = \left\{ u=(u^1, ..., u^p) \in Y^p \mid \sum_i u^i = 0 \right\}$$

On définit la bifonction adjointe (F*x*) (cf R. T. Rockafellar [R ]) sur

$$\{u^*=(u^{*1}, ..., u^{*p}) \in Y^p,$$

par

$$(F^*x^*)(u^*) = \mathrm{Inf}_{u,x}(Fu)(x) - \langle x,x^*\rangle + \langle u,u^*\rangle$$

où $x^*=(x^{*1}, ..., x^{*p}) \in X$ est le vecteur perturbation pour le problème dual concave :

$$\mathrm{Sup}_{u^* \in A^\perp} (F^*0)(u^*) \qquad \textbf{(D5)}$$

Le théorème A1 de l'annexe établit le lien entre un multiplicateur optimal de (P5) et une solution optimale de (D5).

Une formulation symétrique entre les problèmes (P5) et (D5) est possible lorsque les variables primales x sont elles-même contraintes à appartenir à un sous-espace vectoriel L : considérons une partition de l'espace Y des contraintes en $Y_1 x ... x Y_q$. Pour chaque sous-ensemble de la partition, on crée une copie différente des variables primales, notée par $x^j$, j=1, ..., q, et à chaque contrainte j, on associe un vecteur perturbation ($u_j^i$), la variable duale associée étant notée ($u^*{}_j^i$). L est donc le sous-espace de $X^q$ tel que :

$$L = \left\{(x^1, ... ,x^q) \in X^q \mid x^1 = ... = x^q\right\}$$

Le problème primal peut donc être écrit de façon équivalente comme suit :

$$\mathrm{inf}_{x \in L, u \in A}(Fu)(x) \qquad \textbf{(Q)}$$

et son dual peut être écrit sous la forme :

$$\mathrm{sup}_{u^* \in A^\perp, x^* \in L^\perp} \mathrm{Inf}_{x,u} (Fu)(x)$$

L'intérêt de cette formulation est que le couplage entre blocs de contraintes consiste seulement en quatre sous-espaces vectoriels, deux à deux orthogonaux. Les pq sous-problèmes que nous obtenons sont de la forme :

$$\begin{cases} \mathrm{Sup} \ \mathrm{Inf} \ \left\{f_i(x_i^j) - \langle x_i^j , x_i^{*j}\rangle + \langle u_j^j , u_j^{*i}\rangle\right\} \\ \qquad (u_j^{*i}, x_i^{*j}) \quad (x_i^j, u_j^j) \quad g_i^j(x_i^j) = u_j^j \end{cases}$$

### 4-3-2-4-2 L'algorithme ADPG1

Dans ce paragraphe, nous allons appliquer l'algorithme ADP2G dans une situation particulières que nous avons ci dessus : le problème (Q). A une itération donnée, soient

$$(\overline{x},\overline{u}) \in L \times A \text{ et } (\overline{x^*}, \overline{u^*}) \in L^\perp \times A^\perp$$

les solutions (courantes) primale et duale. Les nouvelles solutions sont obtenues en deux étapes :

## L'algorithme ADPG1

**Pas proximal**

Résoudre les sous-problèmes suivants pour tout i,j :

$$\begin{cases} \text{Sup Inf} \left\{ f_i(x_i^j) - \langle x_i^j , x_i^{*j} \rangle + \langle u_i^j , u_i^{*j} \rangle + \frac{1}{2\lambda} \| x_i^j - (\overline{x_i^j} + \lambda \overline{x_i^{*j}}) \|^2 \right) + \frac{1}{2\lambda} \| u_i^j - (\overline{u_i^j} - \lambda \overline{u_i^{*j}}) \|^2 \Big\} \\ (u_i^{*j}, x_i^{*j}) \quad (x_i^j, u_i^j) \end{cases}$$

Les solutions duales sont données par :

$$x + \lambda \overline{x^*} = \overline{x} + \lambda \overline{x^*}$$

$$u + \lambda \overline{u^*} = \overline{u} + \lambda \overline{u^*}$$

**Pas de projection**

Projeter les solutions primales et duales sur leurs sous-espaces respectifs :

$$(\overline{x}, \overline{x^*}) := \text{Proj}_{L \times L^\perp}(x, x^*)$$

$$(\overline{u}, \overline{u^*}) := \text{Proj}_{A \times A^\perp}(u, u^*)$$

Nous allons revoir plus en détail cet algorithme lorsque nous parlerons au Chapitre suivant de l'application aux problèmes de transport. En ce qui concerne les résultats de convergence, nous faisons référence au Chapitre 3 de ce rapport.

# Chapitre V

# Applications Numériques

## 5-1 Introduction

Dans ce Chapitre, nous distinguerons deux types d'applications numériques. D'une part celles implémentées sur une machine séquentielle et celles implémentées sur une machine parallèle, d'autre part.

Le premier type a été programmé en langage C sur une station SUN 4 (Sparc 2) "flagada", faisant partie du matériel du laboratoire ARTEMIS. Le second a été programmé sur une Connection Machine (CM2), par connection à travers la passerelle d' ETCA "scapin", en CParIS, langage dérivant du C et adapté à une machine parallèle.

## 5-2 Application au problème de localisation

Nous examinons dans ce paragraphe la modélisation et la décomposition du problème de localisation avec normes mixtes (les résultats que nous présenterons dans la suite portent sur des normes euclidiennes). Ce problème a déjà été étudié par Idrissi et al(1989) qui l'ont ramené à la forme (3-1-2). Nous signalons au lecteur que la plupart des résultats numériques que nous présenterons portent sur ce problème. Ainsi, nous avons pu tester et comparer certains algorithmes.

### 5-2-1 Formulation du problème

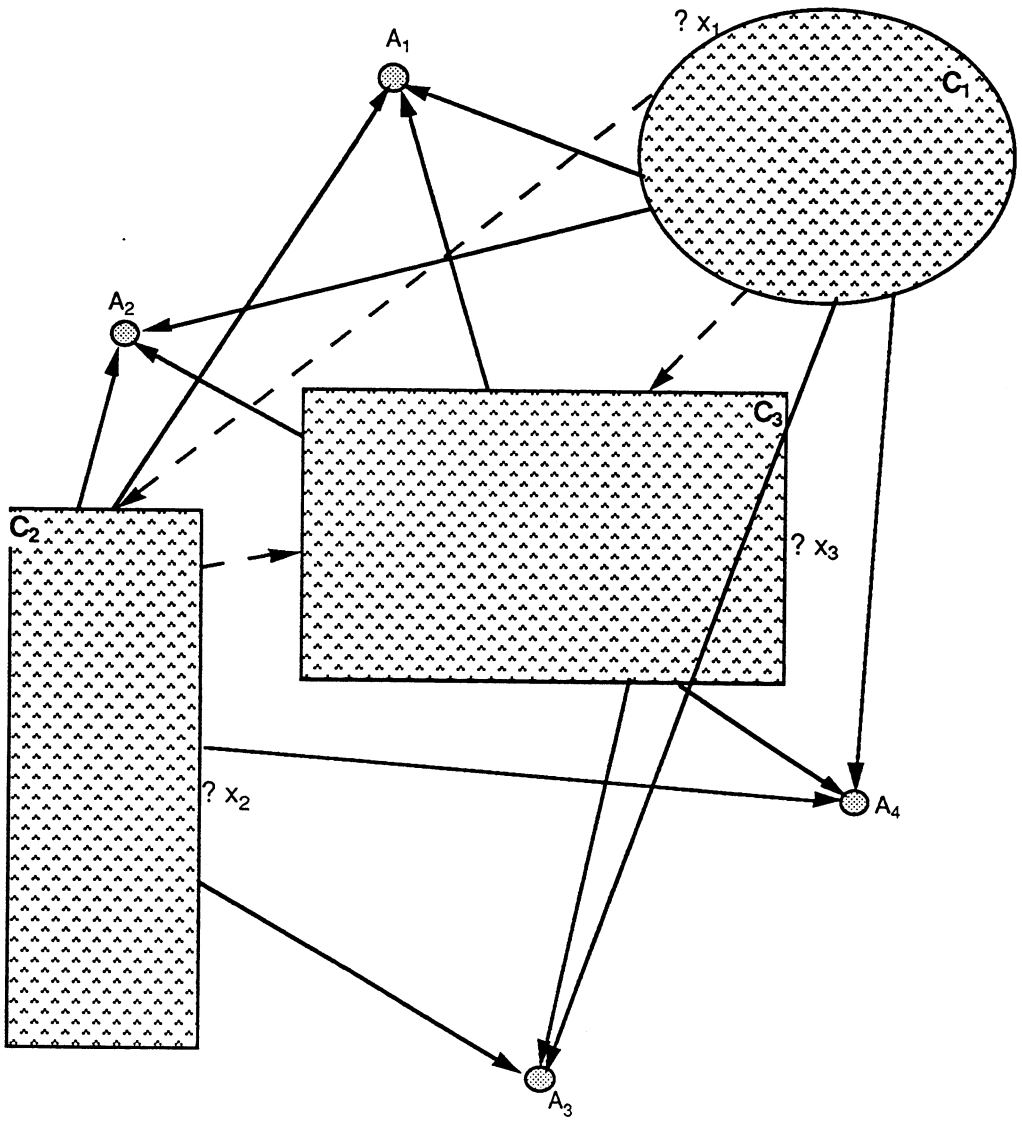
On suppose que l'on a m installations existantes aux points $a_j \in \mathbb{R}^N$ $j \in J=\{1,...,m\}$ et que l'on veut en placer n nouvelles, dont on notera les positions par $x_i \in \mathbb{R}^N$ $i \in I=\{1,...,n\}$ satisfaisant certaines conditions (contraintes) de sorte que le coût des activités entre les anciennes et les nouvelles installations d'une part et entre les nouvelles d'autre part soit minimal.

Les connections entre les installations sont données par un graphe G dont l'ensemble des sommets est $\{a_j, j \in J\} \cup \{x_i, i \in I\}$ et l'ensemble des arcs est donné par $E_a \cup E_x$ ; $E_a$ étant l'ensemble des arcs liant une nouvelle et une ancienne facilité qu'on orientera par convenance des $x_i$ vers les $a_j$ et qu'on notera par (i,j), et $E_x$ l'ensemble des arcs liant deux nouvelles installations arbitrairement orientés et notés par (i,l) si $x_i$ est le sommet de départ et $x_l$ est le sommet d'arrivée de l'arc.

On peut imposer deux types de contraintes à cette localisation :

- $x_i \in C_i$ (par exemple pour limiter les activités d'une facilité) et
- $(x_i - x_l) \in C_{il}$ (par exemple pour éviter que deux nouvelles installations soient éloignées);

$C_i$ et $C_{il}$ peuvent être des convexes fermés, polyhèdres ou demi-espaces. Sur la figure ci-dessous est représenté un exemple de graphe correspondant à un problème à quatre centres existants et trois à localiser.




$A_i$, i=1...4 : les centres existants
$x_i$, i=1...3 : les nouveaux centres à chercher dans les convexes $C_i$, i=1...3, respectivement

Exemple de schéma pour le problème de localisation.

Le problème de localisation de multicentres avec contraintes et normes mixtes peut être formulé de la manière suivante:

$$\text{Min} \quad \sum_{(i,j)\in E_a} \gamma_{ij}(x_i - a_j) \quad + \quad \sum_{(i,l)\in E_x} \tilde{\gamma}_{il}(x_i - x_l)$$

$$(\mathcal{P}) \qquad x=(x_1,\ldots,x_n)$$
$$x_i \in C_i$$
$$(x_i - x_l) \in C_{il}$$

$\gamma_{ij}$ et $\tilde{\gamma}_{il}$ sont des normes qui peuvent être polyhèdrales.
$C_i$ et $C_{il}$ sont des convexes fermés qui peuvent être des polyhèdres.

On suppose de plus que les hypothèses suivantes sont vérifiées:

**(H1)** Le problème ($\mathcal{P}$) admet au moins une solution.

**(H2)** La restriction $G_X$ du graphe G aux connections entre nouvelles installations est connexe.

**(H3)** $(\prod_{i\in I} ri(C_i)) \cap (\bigcap_{(i,l)\in E_X} D_{il}) \neq \varnothing$

où $D_{il} = \{x / (x_i - x_l) \in ri(C_{il})\}$.

L'hypothèse (H1) assure l'existence de solutions. L'hypothèse (H2) suppose que (P) n'est pas décomposable en sous-problèmes complètement indépendants les uns des autres. Dans le cas où les contraintes sont des polyhèdres, (H3) implique seulement que l'ensemble des contraintes est non vide, c'est une condition de qualification des contraintes dans le cas géneral.

### 5-2-2 Dualité et conditions d'optimalité

Reprenons le développement de Idrissi et al(1989).

En remplaçant les contraintes par leurs fonctions indicatrices, le problème ($\mathcal{P}$) est équivalent au problème suivant :

($\mathcal{P}'$) Minimiser $\varphi(x)$

où

$$\varphi(x)= \sum_{(i,j)\in E_a} \gamma_{ij}(x_i-a_j)+ \sum_{(i,l)\in E_X} \tilde{\gamma}_{il}(x_i-x_l)+ \sum_{(i,l)\in E_X} \chi_{C_{il}}(x_i-x_l)+\sum_{i\in I}\chi_{C_i}(x_i)$$

($\chi_A$ désigne la fonction indicatrice de A).

Dans le but de simplifier le problème, on va dans la suite faire des copies des variables (grace au théorème 5-2-1). Cette opération apparait à première vue comme une complication du problème (augmentation de la taille du problème). Mais son interêt est de transformer le problème (P) en un problème faiblement séparable sur l'espace produit U :

$$X = \left(\mathbb{R}^N\right)^n \text{ et } U = \left(\mathbb{R}^N\right)^{|E_a|} x \left(\mathbb{R}^N\right)^{|E_X|} x \left(\mathbb{R}^N\right)^{|E_X|} x \left(\mathbb{R}^N\right)^n$$

Considèrons maintenant la fonction objective perturbée suivante:

$$\Phi : X x U \rightarrow \overline{\mathbb{R}}$$

où $\Phi$ est une fonction propre, convexe et fermée définie par :

$$\Phi(x\ ;\ \alpha,\tilde{\alpha},\hat{\alpha},\alpha_0) = \sum_{(i,j)\in E_a} \gamma_{ij}(x_i - a_j + \alpha_{ij}) + \sum_{(i,l)\in E_x} \tilde{\gamma}_{il}(x_i - x_l + \tilde{\alpha}_{il})$$

$$+ \sum_{(i,l)\in E_x} \chi_{C_{il}}(x_i - x_l + \hat{\alpha}_{il}) + \sum_{i\in I} \chi_{C_i}(x_i + \alpha_i)$$

Avec ces notations, il est clair que

$$(\mathcal{P}) \iff \{ \min_x \Phi(x;0) \}$$

Selon Laurent (1972), le problème dual de (P) est alors donné par

$$(\mathcal{D}) \qquad \{\max_p -\Phi^*(o;p) \}$$

$\Phi^*$ etant la fonction conjuguée de $\Phi$ i.e.

$$\Phi^*(0;\ p,\tilde{p},\hat{p},p_0) = \begin{cases} \sum_{(i,j)\in E_a} \langle p_{ij},a_j\rangle + \sum_{(i,l)\in E_x} \chi^*_{C_{il}}(\hat{p}_{il}) + \sum_{i\in I} \chi^*_{C_i}(p_{0i}) \\ \qquad \gamma^o_{ij}(p_{ij}) \le 1\ ,\ (i,j)\in E_a \\ \qquad \text{si} \quad \tilde{\gamma}^o_{il}(\tilde{p}_{il}) \le 1\ ,\ (i,l)\in E_x \\ \qquad Mp + \tilde{M}(\tilde{p}+\hat{p}) + p_0 = 0 \\ +\infty \qquad \text{sinon} \end{cases}$$

où $p = (p_{ij})$ , $(i,j) \in E_a$, $\tilde{p} = (\tilde{p}_{ij})$ et $\hat{p} = (\hat{p}_{ij})$, $(i,k) \in E_x$ et $p_0 = (p_{0i})$, $i \in I$.
M est la $n \times |E_a|$ matrice définie par :

$$M_{i,(k,j)} = 1 \text{ si } k = i$$

$$= 0 \text{ sinon}$$

$\tilde{M}$ est la matrice d'incidence arcs-sommets du graphe $\tilde{G}$.
On désigne par $\gamma^o_{ij}$ et (resp. $\tilde{\gamma}^o_{il}$ ) la norme duale de $\gamma_{ij}$ (resp. $\tilde{\gamma}_{il}$ )

**Theoreme 5-2-1**

Sous les hypothèses (H1) et (H3), x est solution de (P) et $P = (p,\tilde{p},\hat{p},p_0)$ est solution de (D) ssi :

$$x_i \in C_i \qquad , i \in I$$

$$x_i - x_k \in C_{ik} \qquad , (i,k) \in E_x$$

$$p_{ij} \in \partial\gamma_{ij}(x_i - a_j) \qquad , (i,j) \in E_a$$

$$\tilde{p}_{ik} \in \partial\tilde{\gamma}_{ik}(x_i - x_k) \qquad , (i,k) \in E_x$$

$$p_{ik} \in N_{C_{ik}}(x_i - x_k) \qquad , (i,k) \in E_x$$

$$p_{oi} \in N_{C_i}(x_i) \qquad , i \in I$$

$$M\,p + \tilde{M}(\tilde{p}+\hat{p}) + p_o = 0$$

**Preuve** cf Idrissi et al (1989)

On se ramènera à la résolution du problème sur l'espace $U$, défini ci-dessus. Soit A le sous-espace de $U$ défini par :

$x \in A$ ssi

* $x_{ij} = x_i \qquad (i,j) \in E_a$
* $\tilde{x}_{ik} = x_i - x_k \qquad (i,k) \in E_x$
* $\hat{x}_{ik} = x_i - x_k \qquad (i,k) \in E_x$
* $x_{oi} = x_i \qquad i \in I$

où $x_i \in \mathbb{R}^N$ pour $i \in I$.

et B le sous-espace de $U$ défini par :

$$B = \{ P \in U / Mp + \tilde{M}(\tilde{p}+\hat{p}) + p_o = 0 \} = A^{\perp}$$

On définit l'opérateur T comme suit :

$$T = \prod_{(i,j)\in E_a} T_{ij} \; x \prod_{(i,k)\in E_x} \tilde{T}_{ik} \; x \prod_{(i,k)\in E_x} \hat{T}_{ik} \; x \prod_{i\in I} T_i$$

où

$$T_{ij}(x) = \partial\gamma_{ij}(x_{ij} - a_j) \quad , (i,j) \in E_a$$

$$\tilde{T}_{ik}(x) = \partial\tilde{\gamma}_{ik}(\tilde{x}_{ik}) \quad , (i,k) \in E_x$$

$$\hat{T}_{ik}(\hat{x}) = {}^{N}C_{ik}(\hat{x}_k) \quad , (i,k) \in E_X$$

$$T_i(x) = {}^{N}C_i(x_{oi}) \quad , i \in I$$

Le problème (P) se ramène au problème suivant qui est de la forme (3-2-1) :

**Trouver $a \in A$ et $b \in B$, tels que $b \in Ta$ (L1)**

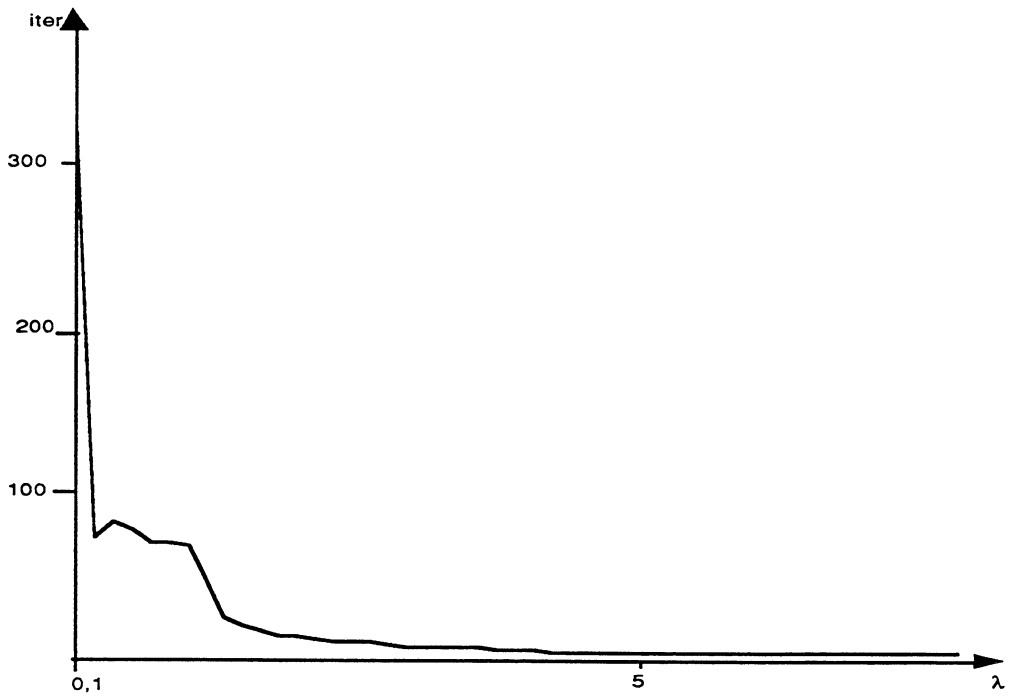
Cette forme nous permet de résoudre le problème (P) par plusieurs algorithmes proximaux, notamment ceux que nous avons décrits au Chapitre 3.

### 5-2-3 Algorithme proximal projeté au problème de localisation

Dans ce paragraphe, nous allons présenter certains résultat avec les deux versions, primale et primal-duale de l'algorithme proximal projeté.

L'algorithme (PR1) avec paramètre constant :

Reprenons l'algorithme PR1 (cf debut de §4-3-2-3). Les résultats que nous avons obtenus confirment le fait que cet algorithme converge d'autant plus vite que le paramètre est grand




*Convergence de l'algorithme Proximal projeté avec λ constant*

Le problème est que pour un paramètre $\lambda$ constant, la suite générée par cet algorithme converge vers la solution du problème régularisé et pas vers celle du

problème de départ. A part pour certaines valeurs du paramètre, la figure suivante confirme (en moyenne) le fait que plus $\lambda$ est grand, plus $x_\lambda$ est loin de la solution $x^*$ du problème de départ.

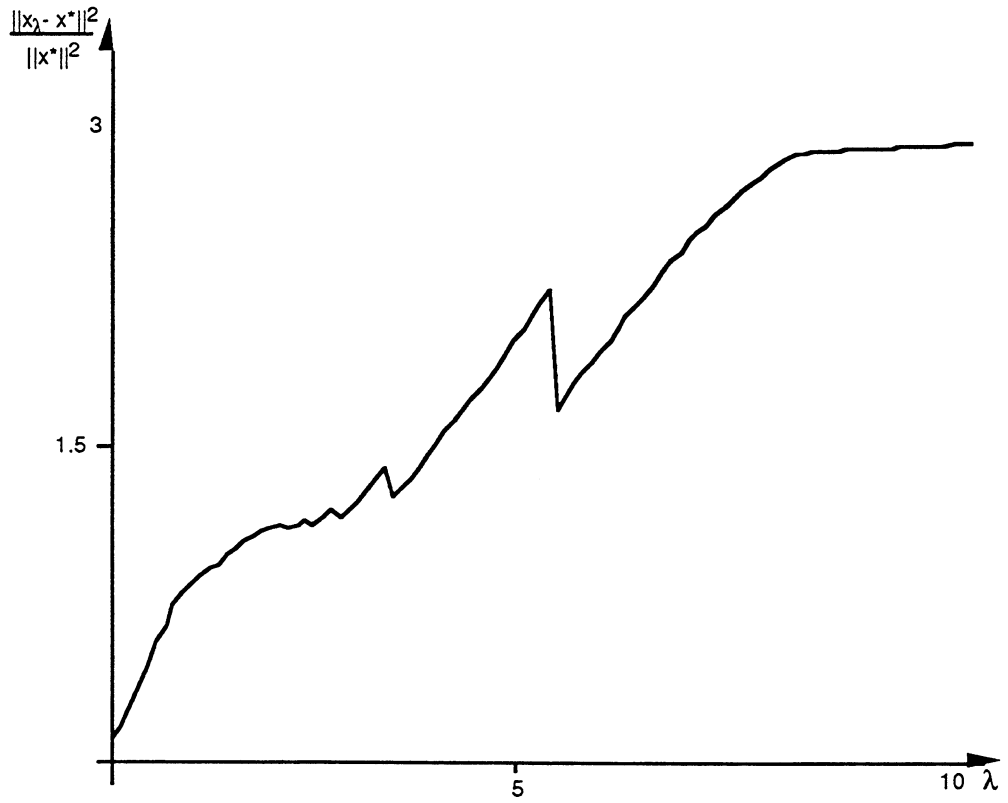



*Figure représentant la distance relative de la solution $x_\lambda$, du problème régularisé, à $x^*$, solution du problème originel.*

L'algorithme (PR1) avec paramètre tendant vers zéro

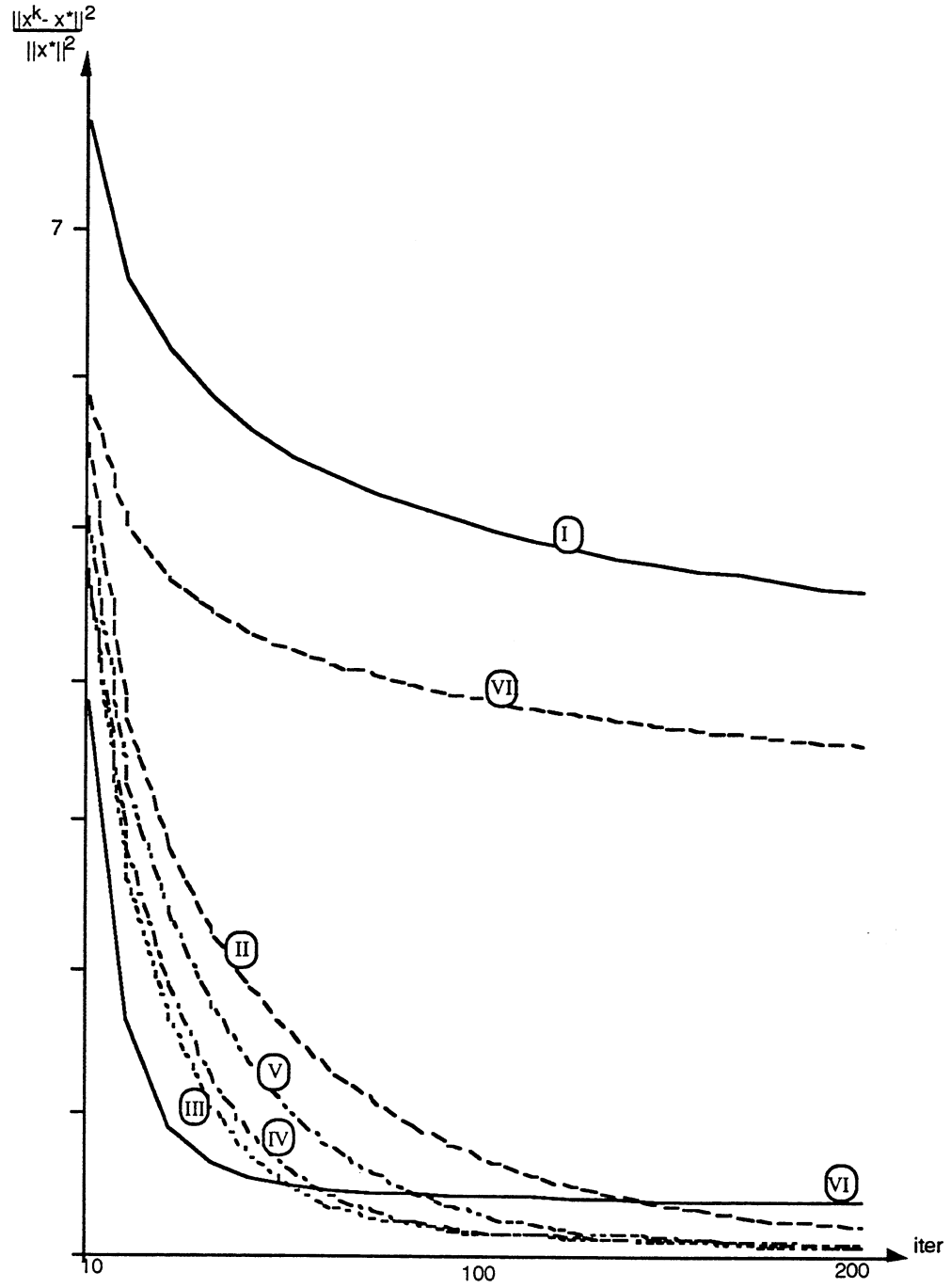
Comme nous l'avons indiqué dans le Chapitre 3, on n'a pas de preuve que cet algorithme converge vers la solution $x^*$, lorsque la suite $\{\lambda_k\}$ tend vers zéro. La seule propriété qu'on a est la convergence ergodique, cf Passty (1979). Après plusieurs discussions, et tests numériques, nous avons conjecturé qu'on a convergence si la suite $\{\lambda_k\}$ n'est pas sommable et est de carré sommable.

La suite générée par cet algorithme est très sensible au comportement de la suite $\{\lambda_k\}$. Ce que nous mettons en évidence par le tableau et la figure qui suivent :

| I | II | III | IV | V | VI | VII |
|---|---|---|---|---|---|---|
| 23,00 | 23,00 | 23,00 | 23,00 | 23,00 | 23,00 | 23,00 |
| 7,75 | 5,54 | 4,67 | 4,69 | 5,03 | 3,78 | 5,85 |
| 6,65 | 3,65 | 2,58 | 2,77 | 3,21 | 1,62 | 4,98 |
| 6,17 | 2,79 | 1,63 | 1,84 | 2,34 | 0,88 | 4,62 |
| 5,86 | 2,24 | 1,05 | 1,24 | 1,75 | 0,64 | 4,40 |
| 5,64 | 1,83 | 0,69 | 0,84 | 1,32 | 0,53 | 4,24 |
| 5,46 | 1,52 | 0,47 | 0,58 | 0,99 | 0,48 | 4,13 |
| 5,33 | 1,26 | 0,34 | 0,41 | 0,75 | 0,46 | 4,04 |
| 5,21 | 1,06 | 0,27 | 0,31 | 0,57 | 0,44 | 3,96 |
| 5,12 | 0,89 | 0,22 | 0,24 | 0,44 | 0,43 | 3,89 |
| 5,03 | 0,76 | 0,19 | 0,20 | 0,35 | 0,42 | 3,83 |
| 4,96 | 0,64 | 0,16 | 0,17 | 0,28 | 0,41 | 3,78 |
| 4,89 | 0,55 | 0,15 | 0,16 | 0,23 | 0,41 | 3,74 |
| 4,83 | 0,48 | 0,13 | 0,14 | 0,19 | 0,40 | 3,70 |
| 4,78 | 0,42 | 0,12 | 0,13 | 0,16 | 0,40 | 3,66 |
| 4,73 | 0,37 | 0,11 | 0,12 | 0,14 | 0,39 | 3,63 |
| 4,69 | 0,32 | 0,11 | 0,12 | 0,13 | 0,39 | 3,59 |
| 4,65 | 0,29 | 0,10 | 0,11 | 0,11 | 0,39 | 3,57 |
| 4,61 | 0,26 | 0,09 | 0,11 | 0,10 | 0,38 | 3,54 |
| 4,57 | 0,23 | 0,09 | 0,10 | 0,10 | 0,38 | 3,51 |
| 4,54 | 0,21 | 0,08 | 0,10 | 0,09 | 0,38 | 3,49 |

Sur chaque colonne, on a la distance relative de $x^k$ à $x^*$ pour k = 1, 10, 20, ..., 200.

Les suites de paramètres $\lambda_k$ correspondantes sont données par les relations qu'on trouvera apès la figure ci-dessous :

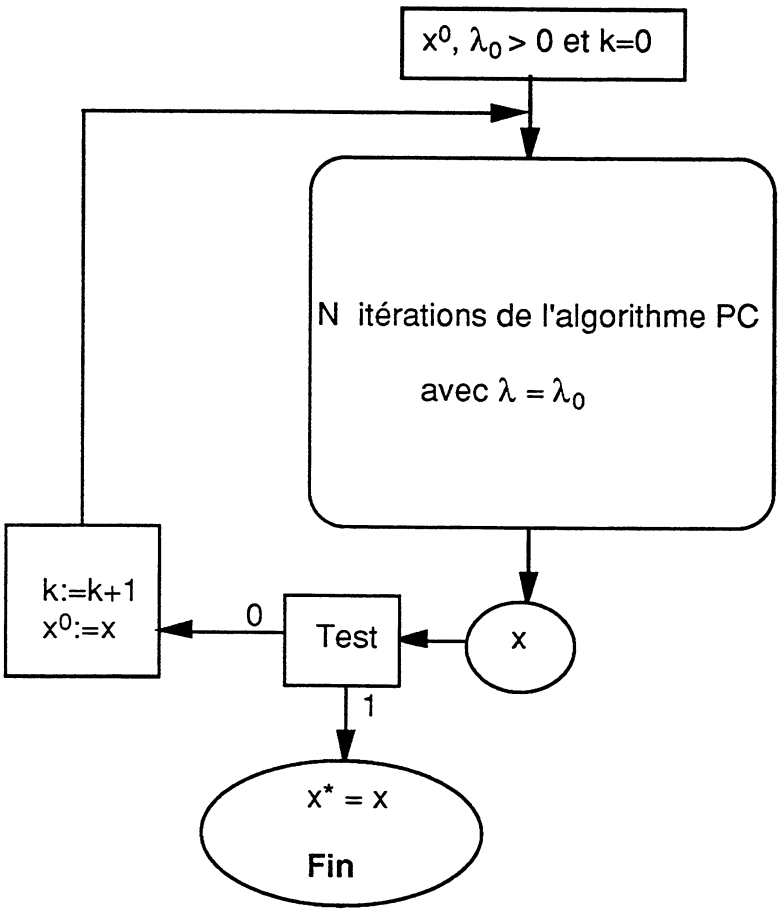



| | | | |
|---|---|---|---|
| I | $\lambda_k = \frac{1}{k}$ | V | $\lambda_k = \frac{1}{[\mathrm{Log}(k)]^{1.2}}$ |
| II | $\lambda_k = \frac{1}{\sqrt{k}}$ | VI | $\lambda_k = \frac{1}{\sqrt{\mathrm{Log}(k)}}$ |
| III | $\lambda_k = \frac{1}{\sqrt[3]{k}}$ | VII | $\lambda_k = \frac{1}{\mathrm{Log}(k) - k + 1}$ |
| IV | $\lambda_k = \frac{1}{\mathrm{Log}(k)}$ | | |

Nous avons effectué plusieurs tests numériques afin de conclure que la meilleure suite de paramètres est la suite générée par [V]. Initialement, nous avions pris cette suite avec une puissance $(1 + \varepsilon)$. Les tests nous ont permis de conclure que la meilleure valeur de $\varepsilon$ est égale à 0.2. Ces suites de paramètres ont été selectionnées entres plusieurs après une grande série de tests afin de mettre en évidence l'effet du choix de la suite sur la convergence de l'algorithme PR.

L'algorithme PRC

Nous allons présenter dans ce paragraphe certains résultats concernant l'algorithme PRC, dont nous rappelons le principe par la figure suivante :




On appelle N le nombre d'itérations locales, c.a.d., le nombre d'itérations pendant lesquelles $\lambda$ est maintenu constant.

Remarquons que nous pourrions faire autant d'itérations locales qu'il faut pour arriver à chaque fois à la solution du problème régularisé. Mais les essais numériques que nous avons effectués nous ont permis de conclure que cette stratégie est coûteuse et peu intéressante.

Les trois tableaux suivants donnent un exemple de comparaison entre trois choix du

nombre d'itérations locales, pour différents choix de la suite des paramètres. Nous faisons aussi la comparaison avec l'algorithme PR1 (N = 1) :

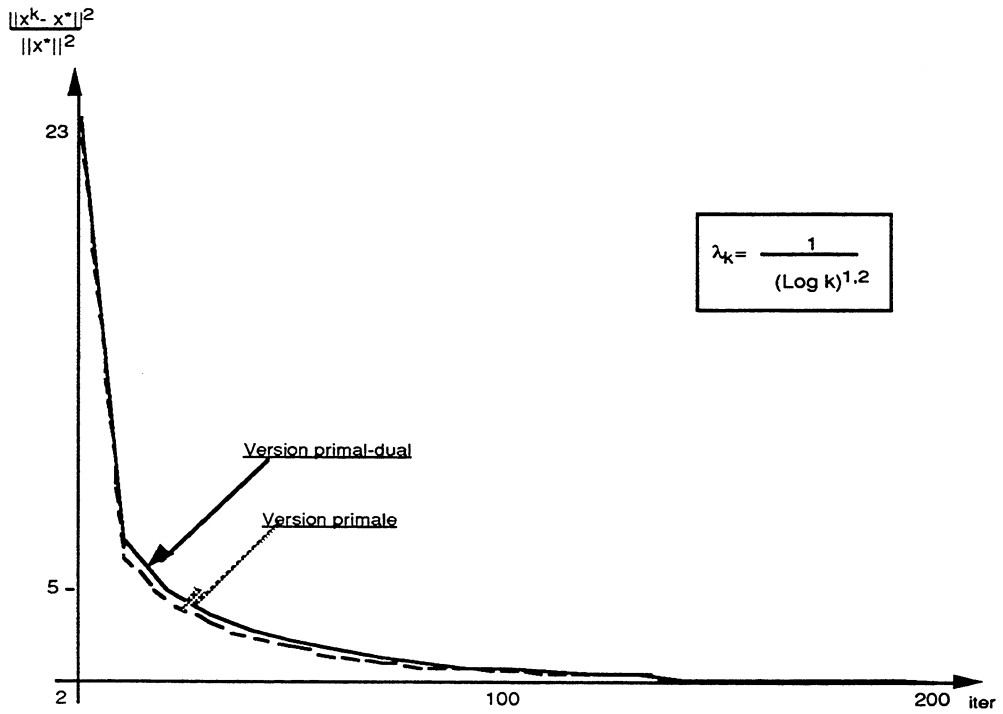
| Pour la suite [I] | | | | |
|---|---|---|---|---|
| Nombre d'itérations locales | 2 | 5 | 10 | 1 |
| Après 50 itérations, f(x)= | 233.751 | 226.376 | 226.228 | 248.354 |
| Après 100 itérations, f(x)= | 231.743 | 226.265 | 226.228 | 243.292 |

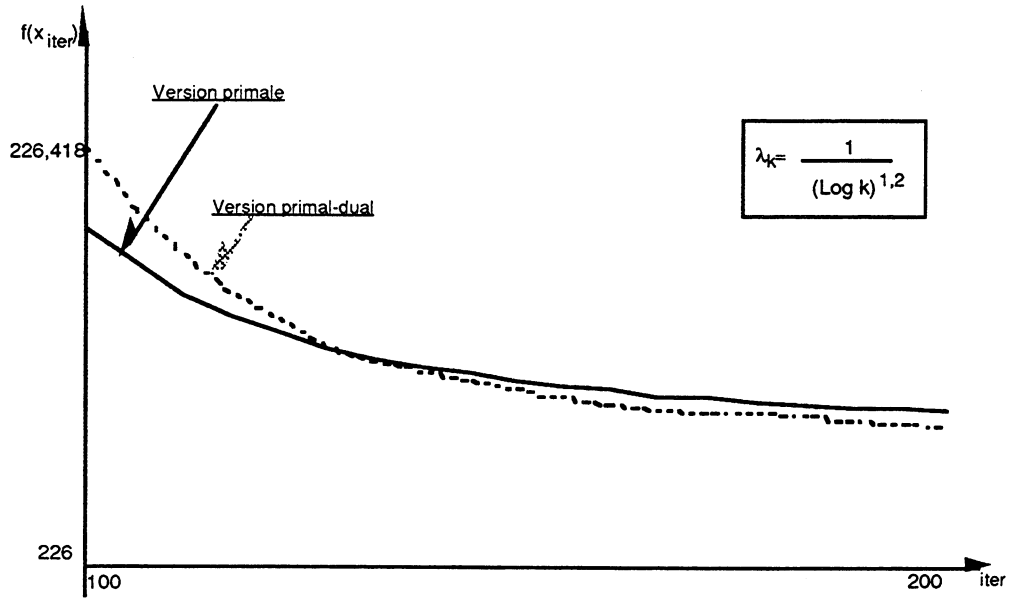
| Pour la suite [II] | | | | |
|---|---|---|---|---|
| Nombre d'itérations locales | 2 | 5 | 10 | 1 |
| Après 50 itérations, f(x)= | 226.336 | 226.263 | 226.278 | 228.823 |
| Après 100 itérations, f(x)= | 226.252 •• | 226.26 | 226.278 | 226.736 |

| Pour la suite [V] | | | | |
|---|---|---|---|---|
| Nombre d'itérations locales | 2 | 5 | 10 | 1 |
| Après 50 itérations, f(x)= | 226.302 | 226.282 | 226.286 | 227.675 |
| Après 100 itérations, f(x)= | 226.269 | 226.276 | 226.286 | 226.364 |

Comparaison de PR1 avec la version primal-duale PR3.

On voit, grâce aux deux figures suivantes, que les deux versions ont les mêmes performances, sauf que la version primal-duale nous permet d'atteindre une valeur plus proche de la valeur optimale exacte.






*Comparaison entre les versions primale et primal-duale de l'algorithme Proximal Projeté*

En ce qui concerne les versions respectives associée à l'algorithme PRC, étudié au Chapitre 3, nous constatons qu'elles ont le même commportement que les versions de l'algorithme PR.

## 5-3 Application de l'algorithme ADP2G

Dans ce paragraphe, nous allons présenter des résultats portant sur certains problèmes quadratiques convexes. Le but est de voir que pour certaines valeurs du paramètres $\lambda$, nous pouvons obtenir des résultats plus intéressants que ceux obtenus par l'algorithme de l'inverse partiel.

En ce qui concerne l'application au problème de localisation, nous allons nous limiter à la figure suivante qui résume en elle même la performance de l'algorithme de décomposition proximale par rapport à celui de l'inverse partiel. La valeur optimale $\lambda^*$ correspond ici à l'inverse de la plus grande constante de Lipschitz $L_i$ des normes $\gamma_i$ utilisées (si elles en ont). Le jeu de onnées que nous utilisons ici nous a été fourni par C. Michlot. Dans cet exemple, nous n'avons que des normes euclidiennes que que nous pondérons par des poids différents.

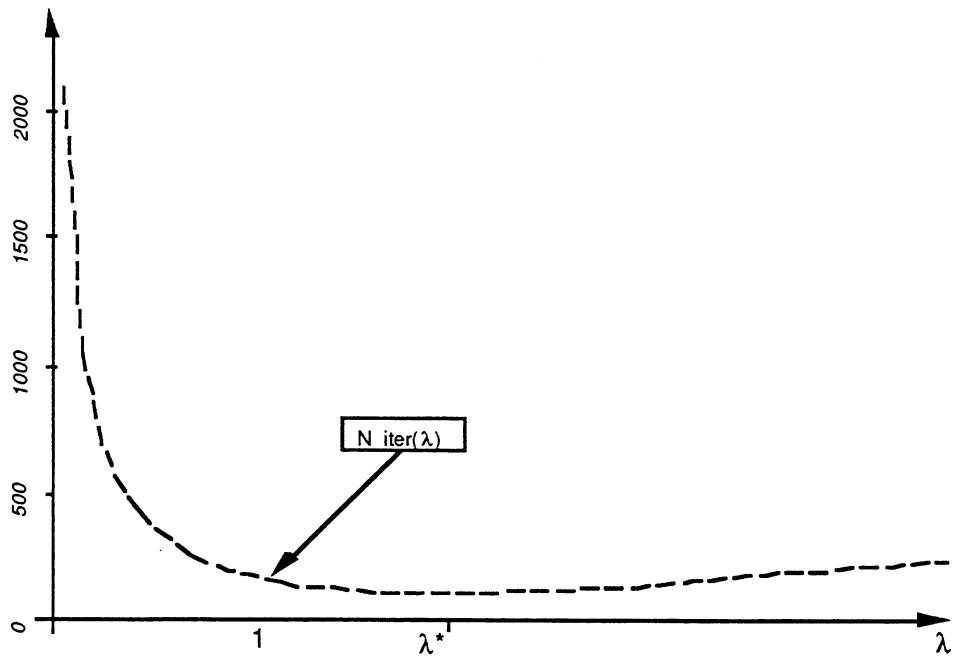



*Figure représentant le nombre d'itérations à la convergence en fonction de $\lambda$*

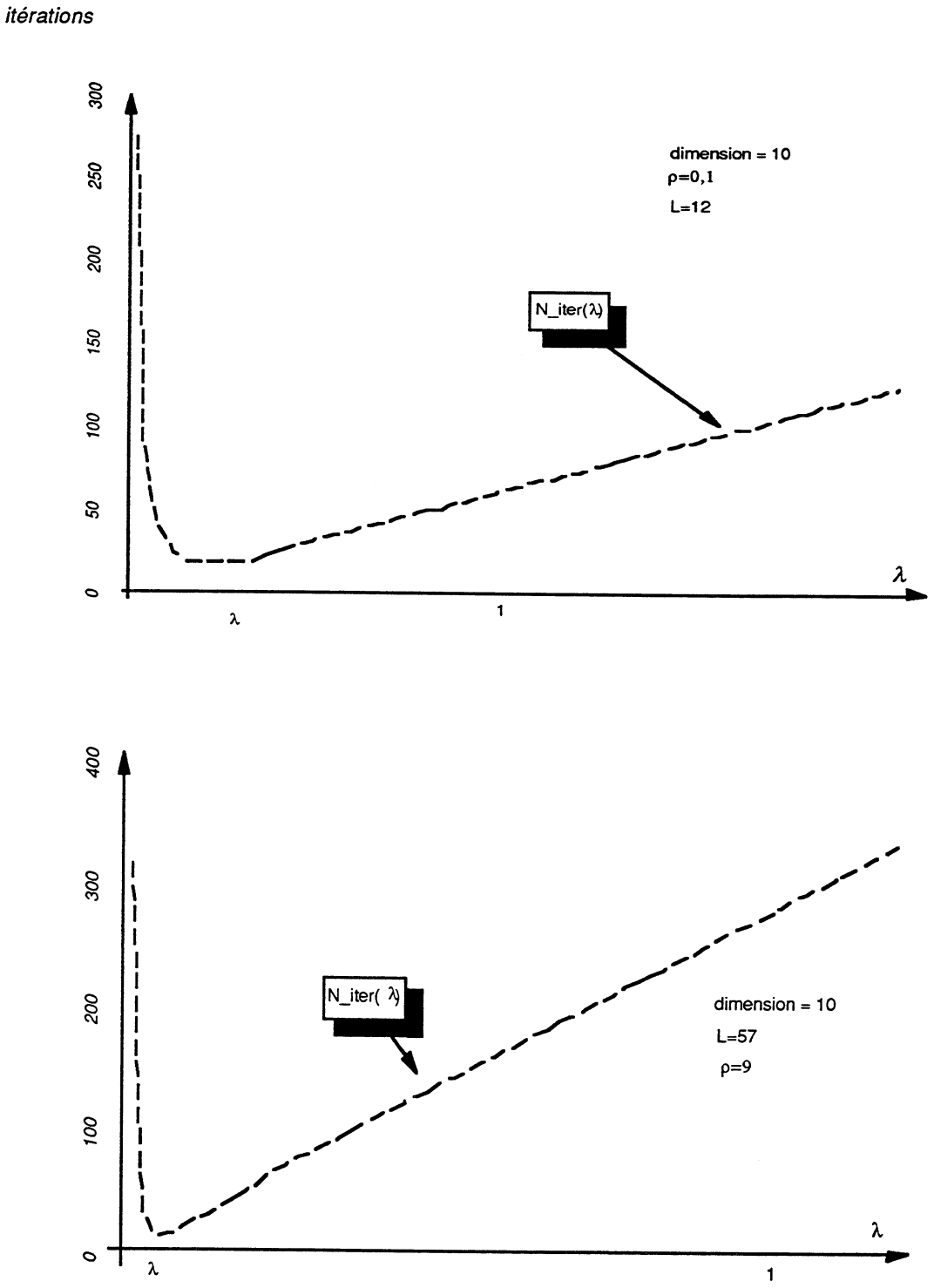
Les figures que nous allons présenter maintenant nous permettent de confirmer les résultats, que nous avons obtenus au Chapitre 3, portant sur le paramètre optimal et sa relation avec la plus grande valeur propre L de la matrice Q correspondant au problème quadratique à résoudre (dans ce cas L est la constante de Lipschitz).

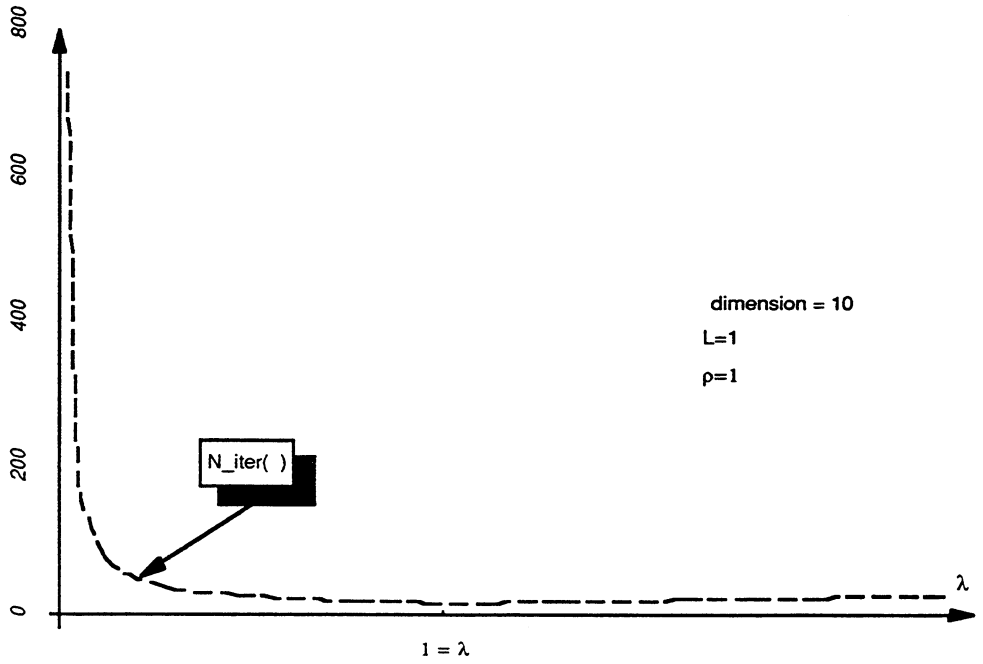
$$\min_{x \in A} \frac{1}{2}\langle x, Qx\rangle + bx$$

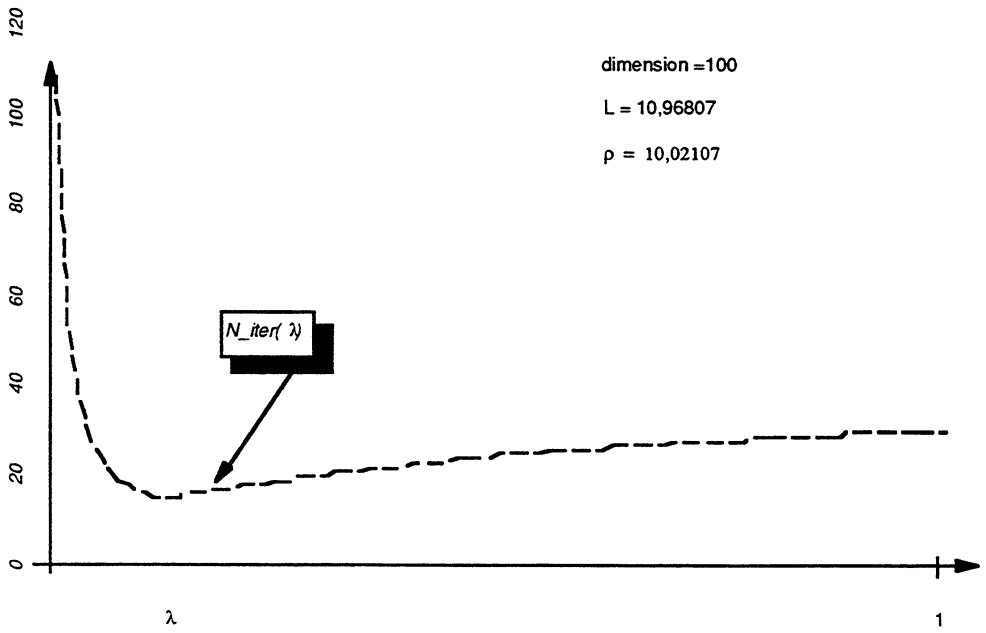
où Q est une matrice (n,n) définie positive et b un vecteur de dimension n. Ils sont choisit de façon aléatoire en imposant la valeur de L et $\rho$.

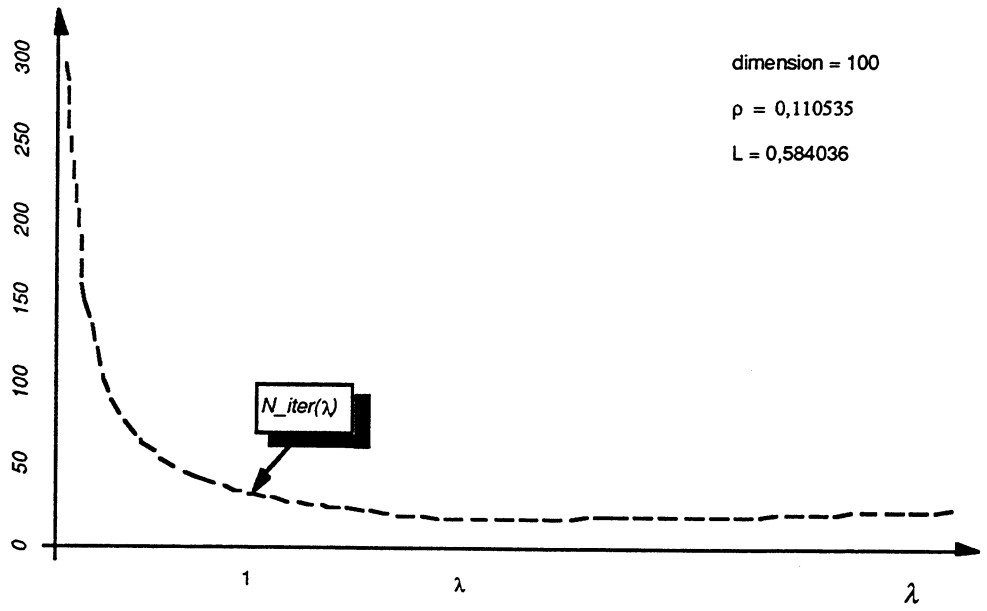
Nous constatons qu'un mauvais conditionnement implique une lenteur de

l'algorithme. Le paramètre d'échelle optimal est très petit si la plus grande valeur propre L de Q est très grande. L'influence de la constante de Lipschitz sur le nombre d'itérations a été analysée pour différents problèmes convexes quadratiques, générés de façon aléatoire, la minimisation étant faite sur un sous-espace simple. C'est ce que nous avons résumé sur les figures suivantes (L et ρ désignent respectivement la plus grande et la plus petite valeur propre de T).




800
600
400
200
0
dimension = 10
L=1
ρ=1
N_iter( )
λ
1 = λ


120
100
80
60
40
20
0
dimension =100
L = 10,96807
ρ = 10,02107
N_iter( λ)
λ
1




Ces résultats portent sur des problèmes quadratiques ; Nous les résumons dans le tableau suivant :

Sur chaque ligne apparaissent en plus que ρ (rapport de Lipschitz),

- la valeur de L : plusgrande valeur propre de la matrice associée a l'exemple correspondant,

- La valeur de 1/L,

- La valeur experimentale du paralètre optimal $\lambda^*$,

- Le nombre d'itérations à la convergence pour $\lambda=\lambda^*$ "N_iter($\lambda^*$)",

- Le nombre d'itérations à la convergence pour $\lambda=1$ (i.e. pour l'algorithme AIP),

- La dimension du problème considèré (dim),

- La tolérence (le test d'arret utilise la distance euclidienne entre l'itéré et la solution du problème).

| ρ | L | 1/L | λ | N_iter(λ*) | N_iter(1) | dim | tolerance |
|---|---|---|---|---|---|---|---|
| | 0,584 | 1,71222 | 2,05 | 17 | 34 | | |
| | 1,068 | 0,93626 | 1,05 | 17 | 20 | | |
| | 4,94 | 0,2024 | 0,26 | 18 | 29 | | |
| ρ=0,1 | 9,7807 | 0,10224 | 0,13 | 18 | 42 | | |
| | 19,4614 | 0,05138 | 0,07 | 18 | 60 | | |
| | 29,142 | 0,0343 | 0,05 | 18 | 70 | | |
| | 96,907 | 0,0103 | 0,02 | 18 | 92 | dim=100 | 0,01 |
| | 1,96807 | 0,5081 | 0,58 | 17 | 19 | | |
| | 5,84035 | 0,17122 | 0,12 | 18 | 35 | | |
| | 10,6807 | 0,0936 | 0,12 | 17 | 44 | | |
| ρ=1 | 20,3614 | 0,04911 | 0,06 | 18 | 60 | | |
| | 30,04213 | 0,0332 | 0,05 | 18 | 70 | | |
| | 49,4035 | 0,02024 | 0,03 | 18 | 82 | | |
| | 97,807 | 0,0102 | 0,02 | 19 | 92 | | |
| | 0,584 | 1,7122 | 2,04 | 16 | 32 | | |
| | 1,068 | 0,93626 | 1,21 | 16 | 18 | | |
| | 4,9403 | 0,2024 | 0,241 | 17 | 27 | | |
| ρ=0,1 | 9,7807 | 0,10224 | 0,121 | 17 | 39 | | |
| | 19,4614 | 0,05138 | 0,061 | 17 | 54 | | |
| | 29,1421 | 0,03431 | 0,041 | 17 | 63 | | |
| | 96,907 | 0,0103 | 0,021 | 17 | 75 | dim=10 | 0,001 |
| | 1,96807 | 0,5081 | 0,58 | 15 | 16 | | |
| | 5,8403 | 0,17122 | 0,211 | 16 | 29 | | |
| | 10,6807 | 0,0936 | 0,121 | 16 | 40 | | |
| ρ=1 | 20,3614 | 0,04911 | 0,061 | 17 | 54 | | |
| | 30,0421 | 0,0332 | 0,041 | 17 | 63 | | |
| | 49,4035 | 0,02024 | 0,031 | 17 | 72 | | |
| | 97,807 | 0,0103 | 0,021 | 17 | 75 | | |

L'influence du paramètre d'échelle sur le nombre d'itérations est illustrée en comparant les colonnes iter($\lambda$) (nombre d'itérations pour $\lambda$ optimal) et iter(1) (nombre d'itérations pour $\lambda = 1$). Ici, le nombre d'itérations correspond à l'algorithme (ADPG) associé au graphe de $\lambda T$.

### Remarque

Les résultats obtenus par l'algorithme de Douglas-Rachford sont très similaires à ceux que nous avons obtenus par l'algorithme de décomposition proximale appliqués au problème de localisation. Par contre, l'algorithme de Peaceman-Rachford, ne converge pas pour cet exemple.

## 5-4 Généralités sur le parallèlisme

### A- Généralités

Le parallélisme est un terme générique qui fait appel, en informatique, à la fois à une méthode de traitement de données et à l'ensemble des outils "hardware" ou "software" utilisés. Ainsi cette matière étant relativement récente, les définitions dépendent souvent des modèles ou des écoles.

Comme nous l'avons signalé à l'introduction du Chapitre 4, Le parallèlisme et le développement de ces outils, est une des motivations pour les méthodes de décomposition.

Le but des définitions introduites au début de ce paragraphe, est de présenter un aperçu des différentes notions utilisées dans le cadre du parallélisme.

### A-1 Quelques définitions

L'architecture d'un système parallèle est constituée par un ensemble de processeurs reliés entre eux. Un programme principal fait exécuter à ceux-ci les différentes tâches , programmes qui le composent. Les communications entre les processeurs sont considérées comme fiables (pour la plupart des ordinateurs de ce type).

Il faut faire la différence avec une architecture de type **distribuée**. Celle-ci est constituée de processeurs éloignés. La communication entre processeurs est alors plus problématique, avec des délais imprévisibles, et des liens de communications non sûrs. La topologie de cette architecture peut être changeante, et faiblement couplée. Les processeurs peuvent de plus avoir des activités dites privées. En fait, la coopération entre les éléments de la structure n'est pas la raison d'être de cette architecture.

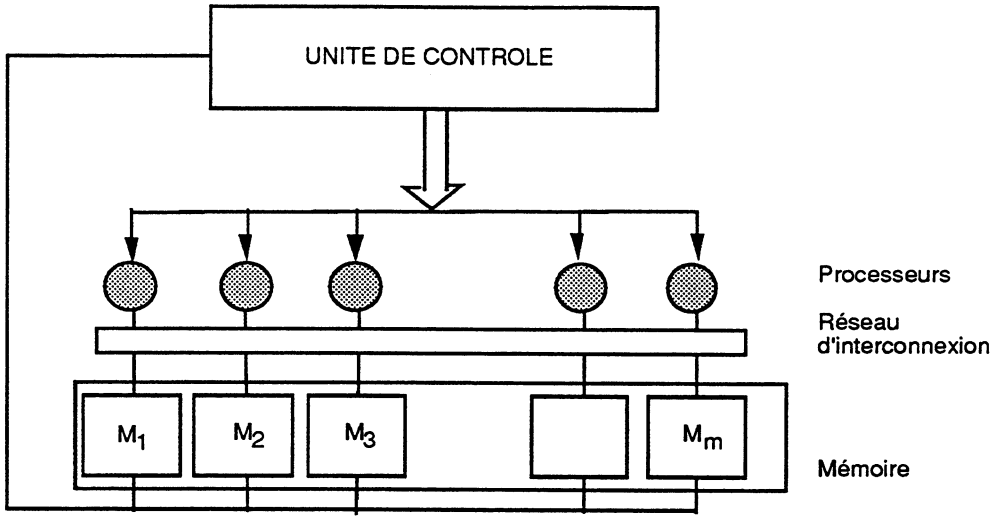
Il existe plusieurs façons d'aborder le parallélisme, ceci en fonction des problèmes et de la manière de les résoudre, par exemple l'affectation des tâches, la synchronisation de celles-ci et la communication entre les processeurs.

### A-2 Les notions de SIMD et de MIMD

Le parallélisme en informatique a été créé pour obtenir une puissance de calcul plus grande, dans bien des cas, des tâches, des traitements indépendants, similaires ou non, doivent être appliqués à des jeux de données. Le fait que ces opérations soient identiques ou non, entraîne deux sortes de parallélisme, définies ci-après (cette classification est due à Flynn (1972) ).

#### A-2-1 La notion de SIMD

Dans le premier cas, les opérations sont identiques pour des jeux de données différents, la notion de parallélisme S.I.M.D. (Single Instruction Multiple Data) sera employée. Ce système est donc bien adapté pour opérer des transformations de matrices dans lesquelles les données reçoivent un traitement identique.

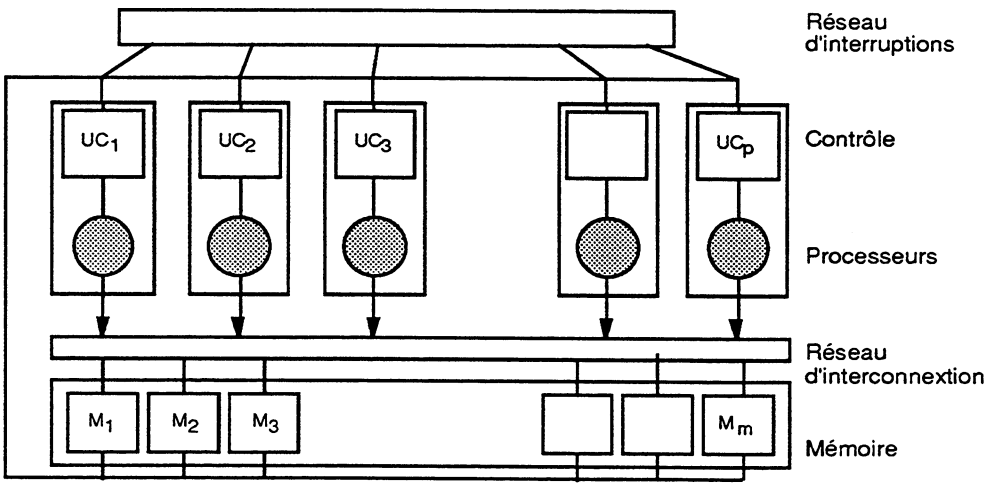



ARCHITECTURE SIMD

### A-2-2 La notion de MIMD

Dans le deuxième cas, les processeurs peuvent être employés de manière indépendante, c'est le M.I.M.D. (Multiple Instruction Multiple Data). Il faut déterminer pour chaque processeur le travail qu'il doit réaliser. Le nombre de communications dans ces machines est alors plus important.

Le contrôle des communications est ici, à gérer par les processeurs eux-mêmes. Des communications peuvent être alors complètement simultanées entre processeurs différents.




Architecture MIMD

## A-3 Quelques définitions

### A-3-1 Les grandes notions en informatique parallèle

Nous allons maintenant nous attacher à définir le vocabulaire, employé pour le parallélisme, pour décrire les outils utilisés.

**Définition A-1**

Un **processeur** est la plus petite unité capable d'effectuer les opérations arithmétiques de base, les instructions de comparaisons et de branchements, et les échanges d'informations.

(Pour un exemple détaillé de processeur, le "Transputer", cf [DR-1988])

**Définition A-2**

La distinction entre ces deux sortes de parallélisme se fait par le nombre de processeurs engagés dans l'architecture. Si celle-ci en contient bien plus que dix, le terme de **parallélisme massif** ou parallélisme à grains fins ("fine grain") sera employé. Dans le cas contraire il sera fait référence au terme de **parallélisme à gros grains** ("coarse grained").

**Définition A-3**

La **topologie** est la structure géométrique décrivant l'organisation des processeurs. La topologie est dite **réelle** (hardware) lorsqu'elle est définie par le cablage physique des processeurs. Elle est **virtuelle** (software) si elle n'est pas une sous-topologie de la topologie réelle.

**Définition A-4**

Le **pipeline** est une unité matérielle qui produit le découpage d'une opération à effectuer en étapes de durées égales qui vont s'exécuter successivement. C'est un mécanisme adapté aux structures régulières que sont par exemple les vecteurs.

La technique du pipeline consiste à anticiper sur la lecture ou l'écriture des données pendant l'exécution de certaines instructions (l'addition par exemple). C'est une optimisation dans l'ordonnancement des traitements de la machine. C'est aussi une forme de parallélisme à un niveau élémentaire qui est compatible avec le SIMD et le MIMD. L'avantage de cette technique est de ne pas nécessiter la duplication de toutes les ressources, et donc d'être peu coûteuse. par contre, elle oblige à figer les opérations une fois pour toutes.(cf [T-1988], [T-1990], et [BT-1989])

**Définition A-5**

Un **réseau systolique** est une structure composée d'un grand nombre de cellules élémentaires identiques et localement interconnectées. Chaque cellule, a au plus la complexité d'un microprocesseur et reçoit uniquement des informations de ses voisines. Elle effectue un traitement simple, et transmet un temps de cycle plus tard, le résultat à ses voisines. Ces cellules évoluent en parallèle, sous contrôle d'une horloge globale, cf Trystram(1988]).

### A-3-2 La synchronisation des opérations des différents processeurs

Si un mécanisme d'horloge globale est présent pour synchroniser les opérations sur les différents processeurs, le terme d'opérations synchronisées (ou de synchronisme) est utilisé, sinon on emploie celui d'asynchronisme. La synchronisation est, par définition, présente dans le cas d'un système SIMD. Elle implique un contrôle plus aisé du

comportement des processeurs, puisqu'ils effectuent tous la même opération à un instant donné. Les algorithmes sont alors plus simples.

### A-3-3 La mémoire

**Définition A-6**

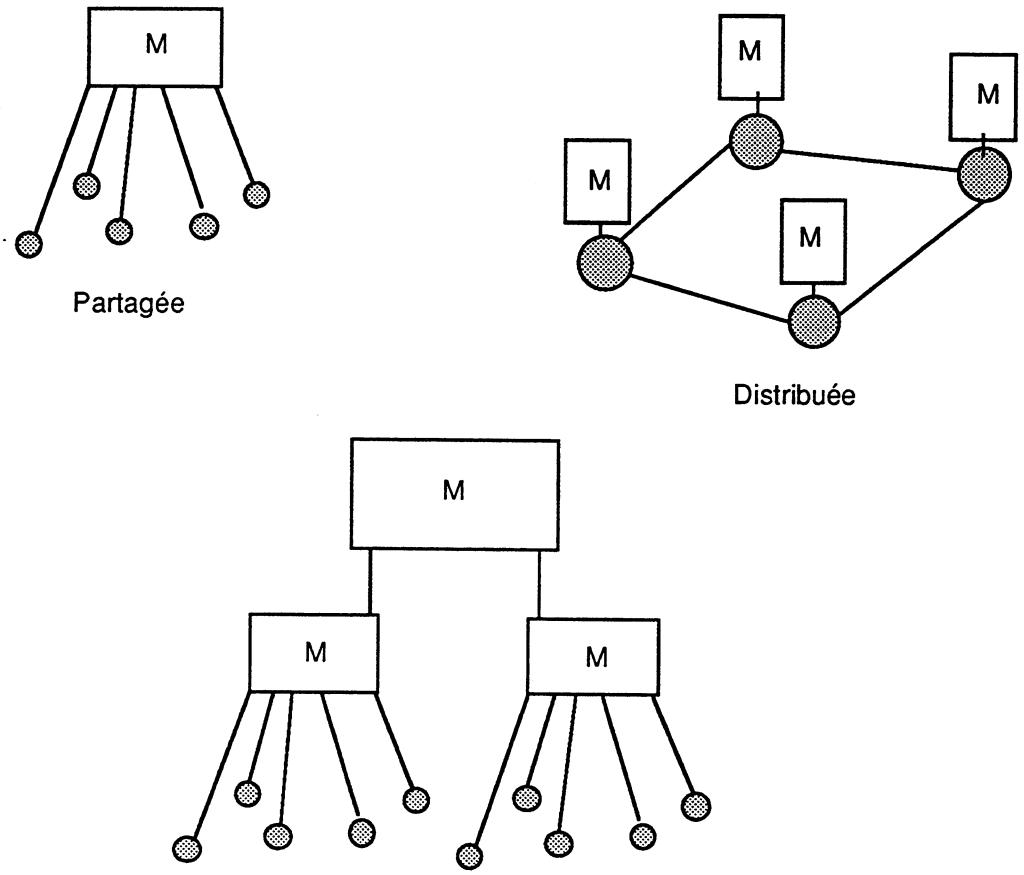
La **mémoire** est dite **partagée** ("shared memory"), si tous les processeurs ont accès à une même et unique zone mémoire. Pour pallier la gestion d'un tel cas, il est possible de les relier par un bus en temps partagé, mais les trop faibles débits des bus limitent rapidement le nombre de processeurs sur une telle machine. Une topologie en grille reliant chaque processeur à chaque banc de mémoire, réseau de type "cross-bar", peut être implantée. Le débit est alors maximal, mais extrèmement coûteux, du fait du nombre de croisements (proportionnel au produit du nombre de processeurs par le nombre de bancs de mémoire).

**Définition A-7**

La **mémoire hiérarchisée** est organisée en plusieurs niveaux.

**Définition A-8**

La **mémoire** est dite **distribuée** ou locale ("local memory"), si chaque processeur a la sienne propre. Il ne peut alors accéder aux données des autres processeurs que par l'envoi de messages.




Des situations intermédiaires sont envisageables.

## A-4 Topologie d'un réseau

### A-4-1 Les critères

La topologie d'un réseau fait référence au graphe des processeurs et de leurs liens de communication. Cette topologie est évaluée suivant la convenance qu'elle offre pour les tâches standards de communication, c'est-à-dire suivant les critères que nous définissons ci-après :

**Définition A-9**

Le **diamètre** d'un réseau est le plus petit nombre de liens, qui doivent être traversés pour relier toute paire de nœuds.

La **connectivité** d'un réseau est le nombre minimal de chemins, qu'il faut détruire pour déconnecter le réseau.

La **flexibilité** d'un réseau est la capacité de faire correspondre aux processeurs différentes topologies, afin de pouvoir lui appliquer, une grande variété d'algorithmes de façon efficace.

Le **délai de communication** est le temps D requis pour effectuer une communication standard.

### 5-4-2- Les modèles

Un modèle d'un réseau représente la façon dont sont organisés les processeurs, ceci en vue de la résolution d'un problème. En effet, suivant l'organisation des liens de communication, la gestion de l'affectation des tâches et des échanges de messages entre processeurs, n'aura pas le même degré de difficulté.

Les topologies courantes sont :

**Le graphe complet** : Tous les processeurs sont connectés entre eux. Dans ce cas, toutes les topologies peuvent être appliquées de façon virtuelle. C'est donc du point de vue flexibilité le réseau idéal. Mais en fait, la réalisation matérielle d'un tel réseau serait extrêmement volumineuse, et coûteuse. Elle existe seulement à de faibles niveaux de hiérarchie pour un petit nombre de processeurs.

**Les processeurs en ligne** : La structure de ce modèle est explicite. Les processeurs sont alignés et sont connectés au (ou) aux voisins qu'il possède. Sur une telle configuration, les communications sont extrèmement coûteuses.

**L'anneau**: Pour réaliser un anneau de processeurs, il suffit que chacun des processeurs soit connecté à deux autres. Le diamètre d'un tel réseau est alors égal à la partie entière du nombre de processeurs en jeu moins un divisé par deux. Les communications se font alors par décalage des informations d'un processeur, à chaque cycle d'horloge.

**L'arbre** : Les arbres les plus utilisés sont l'étoile et bien évidemment l'arbre binaire équilibré. L'inconvénient d'une telle structure est sa faible connectivité. La destruction d'un seul lien entraîne la création de deux groupes séparés de processeurs. Ce modèle possède un processeur racine, pour lequel tous les processeurs n'ont qu'un seul chemin pour le relier. La topologie en arbre est celle qui s'adapte le mieux aux algorithmes de tri.

**La grille** : La structure en grille ou maille ("mesh") semble la plus naturelle. En effet, dans de nombreux problèmes on traite des calculs matriciels, et la grille est le modèle qui permet de faire correspondre la disposition des informations avec celle des processeurs. Cela permet à l'utilisateur de pouvoir repérer l'adresse de ces données, en vue d'un transfert. De plus les processeurs ont couramment quatre liens de communication. Cette construction permet de les connecter facilement sur un plan. On complète souvent, ce réseau en connectant en anneau chaque ligne et chaque colonne, afin de diminuer le diamètre de cette topologie. On obtient une structure de **Tore**.

**L'hypercube** : Tous les processeurs sont connectés à autant de processeurs que la dimension de l'hypercube. Cette dimension est alors égale au diamètre, ainsi qu'au nombre de chemins de communications distincts possibles entre deux processeurs. Cette structure est de grande flexibilité, puisqu'elle contient celle d'un anneau et celle d'une grille.

Comme dans un hypercube le nombre de processeurs est une puissance de 2, $= 2^d$ , où d est la dimension de l'hypercube appelé aussi le d-cube, pour déterminer les processeurs entre eux, une adresse binaire est affectée à chacun d'eux (code de gray).

## A-5 La communication entre processeurs

### A-5-1 Les différents modes

**a) Communications générales (le routeur)**

C'est le mécanisme de contrôle des communications entre les processeurs. C'est à partir de là que sont guidés les paquets vers leur destination à travers le réseau. Le délai de sélection du chemin à parcourir doit être le plus court possible. Le tracé doit être sûr, c'est-à-dire utilisé par le minimum de paquets, pour avoir un temps optimal. C'est en fait le problème de plus court chemin dans un graphe.

**b) Communications par voisinage**

Par ce terme, on désigne dans un système SIMD, les communications de processeur à processeur qui se déroulent de proche en proche. C'est le mode le moins coûteux. Il peut être utilisé en parallèle par tous les processeurs à la fois, en un seul temps d'horloge. il nécessite de connaître l'emplacement relatif de chacun des processeurs, mais pas son adresse précise. C'est souvent le cas lors des calculs matriciels, sur une topologie au moins virtuelle, en grille.

### A-5-2 Les différents problèmes de communication

On parle de problème d'échange total ("total exchange problem" ou "**all to all**"), lorsque tous les processeurs doivent communiquer entre eux en même temps.

Le problème du nœud d'émission ("single node broadcast" ou "**one to all**") consiste à faire envoyer par un nœud un même paquet à tous les autres processeurs.

Le problème dual, du noeud de reception consiste à ce que un processeur reçoive un message venant de tous les autres processeurs s'appelle le problème du nœud d'accumulation ("single node accumulation" ou "**all to one**").

On peut aussi envoyer depuis un processeur, un message distinct à chacun des autres processeurs. Ce problème est appelé le problème du nœud de dispersion ("single node scatter" ou "**one to all personnalisé**").

Le problème inverse existe aussi : faire recevoir par un processeur, un paquet différent de chacun des autres processeurs. C'est le problème du nœud de regroupement ("single node gather" ou "**all to one personnalisé**").

### A-5-3 Le contrôle des envois de données lors des communications

Il y a deux principaux composants du contrôle des transmissions de données ("Data Link Control"). Le "framing" est un mécanisme de reconnaissance du début et de la fin des paquets. Il s'effectue par des systèmes de drapeau, appelé encore "extra bits". La

technique la plus courante est celle du bit, appelée encore rembourrage ("stuffing"). Elle consiste à mettre un drapeau à l'intérieur des paquets transmis. Le second mécanisme concerne la détection des erreurs et la retransmission ("error detection and retransmission") des paquets reçus avec des erreurs. Les techniques de détection des erreurs sont basées sur l'analyse de ces extra bits, après transmission. L'une des formes les plus simples est l'analyse de la parité ("parity check") de la somme des bits transmis. Le receveur a alors à examiner si la parité de l'extra bit correspond bien à la parité de la somme des bits transmis. Une technique plus sophistiquée est appelée "Cyclic Redundancy Check". Elle consiste à ajouter à la fin de chaque paquet une suite de bits (le reste de la division modulo deux du polynôme fixé, appelé le polynôme générateur, ayant des coefficients binaires). Si cette technique ne garantit pas une sécurité absolue, elle est cependant considérée comme étant pratiquement infaillible.

La méthode classique pour corriger les transmissions défectueuses est celle où les mauvais paquets sont retransmis jusqu'à ce qu'ils soient admis comme étant corrects. Le protocle de communication le plus simple est appelé "stop-and-wait". Le protocle le plus utilisé est appelé "go-back-n". L'idée est d'autoriser l'envoi de n paquets après le dernier paquet reçu et accepté, jusqu'à la décision de retransmission des paquets. Pour n égal à un, on retrouve le "stop-and-wait".

### A-5-4 Les différents temps

#### a) Le temps d'un processus de communication

C'est le temps **C**, requis pour préparer les informations en vue d'une transmission. Il comprend, par exemple, la mise sous forme de paquets et la sélection du lien de transmission.

#### b) Le temps de file

C'est le temps d'attente ("queueing") **Q**, en queue de file avant le départ de la transmission. Ceci est dû, par exemple, au fait que le lien est temporairement inopérant.

#### c) Le taux de transmission

C'est le taux de transmission **R** de tous les bits d'un paquet d'informations.

#### d) Le temps de propagation

C'est le temps **P**, écoulé, entre la fin de la transmission du dernier bit du paquet au processeur qui envoie, et la réception du dernier bit du paquet par le processeur qui reçoit.

Aux vues de ces définitions, on a la relation :

$$D=R \times L + Q + P + C,$$

où **L** est la longueur des paquets transmis en bits et **D** est le délai de communication.

Le modèle couramment utilisé est :

$$D = RxL + C$$

## A-6 La complexité en parallélisme

### A-6-1 Généralités

La complexité en parallélisme, tient compte d'un plus grand nombre de facteurs que dans le cas séquentiel. En effet, il faut examiner la taille du problème, le nombre de processeurs, le temps d'execution du programme, et le nombre de messages transmis lors de son déroulement. Ce dernier facteur n'est pas le moindre étant donné le coût d'un transfert d'information et les problèmes que cela pose, du point de vue technique. Il est donc difficile d'en assurer l'optimalité aussi bien en vitesse qu'en sûreté.

Soit p le nombre de processeurs, $t_i$ le temps correspondant à l'exécution d'une tâche, par le processeur i. Le temps de complexité $T_p$ d'un algorithme sur une machine ayant p processeurs, est égal au maximum des $t_i$ en supposant un ordonnancement optimal des tâches. Soit $T_\infty$ le temps de complexité égal au temps de complexité minimal, en prenant autant de processeurs que souhaité.

### A-6-2 Propositions sur le nombre de processeurs utilisés

Nous donnons ici certains résultats utiles. Les démonstrations qui ne sont pas données peuvent être retrouvées dans la référence [BT-1989] (à la section 1.2 de l'introduction).

$$T_\infty = \text{Log } n$$

où n est le nombre de processeurs d'entrée

$$T_p \leq c\, T_q$$

où c est un entier naturel, avec q=cp

$$T_1 < T_\infty + \frac{T_1}{p}$$

$$\frac{T_1}{p} \leq T_p < 2\frac{T_1}{p}$$

### A-6-3 Relations avec un algorithme séquentiel

Soit $T^*(n)$ le temps d'execution optimal d'un algorithme séquentiel pour résoudre un problème de taille n. Soient $S_p(n)$l'accélération de l'algorithme, due à la mise en parallèle et $E_p(n)$ l'efficacité de l'algorithme, alors :

$$S_p(n) = \frac{T^*(n)}{T_p(n)}$$

$$E_p(n) = \frac{S_p(n)}{p} = \frac{T^*(n)}{p\, T_p(n)}$$

La meilleure valeur est obtenue pour $E_p(n)$ égale à 1 i.e. pour $S_p(n)$ égal à p.

# B La Connection Machine

## B-1 Le choix de la machine

Les notions vues précédemment conduisent à des modes de fonctionnement très différents. Elles sont à mettre ensuite en relation avec la méthode de résolution adoptée.

Le MIMD est utilisé par le Méganode [B-1991], système sur lequel le LMC développe des outils. Au début, nous avons entrepris l'implémentation de l'algorithme de l'inverse partiel appliqué au problème de localisation, déjà traité en séquentiel par Idrissi et al [ILM-1989]. Nous avons repris cet exemple, mais nous avons été confronté à un problème de configuration optimale du réseau de communications entre les processeurs.

Le choix de la Connection Machine semblait plus approprié, d'autant que la structure matricielle peut s'obtenir, assez facilement. De plus, il était intéressant de pouvoir comparer la méthode que nous avons appliqué au problème de transport, avec d'autres en particulier celle de ZENIOS [Z-1991a].

## B-2 Présentation générale

La "Connection Machine" a été fabriquée par la société américaine Thinking Machine Corporation de Cambridge dans le Massachusetts. A la fin de 1987, 14 exemplaires de la CM2, modèle dont nous allons parler, avaient été vendus. Le modèle suivant : la CM5, est maintenant disponible.

Cette machine est le résultat des travaux de Scott FAHLMAN, qui avait défini un modèle de réseau sémantique pour la représentation des connaissances. Ils ont été ensuite repris par D.W.HILLIS [H-1985]. Les idées de base sont celles du traitement de l'information en parallèle. Le talon d'Achille de l'architecture de tous les ordinateurs est formé par les accès entre la mémoire centrale et le, ou les processeurs. Ce goulot d'étranglement des informations entraîne une inefficacité proportionnelle à la taille de la mémoire. La Connection Machine palie cet inconvénient par une architecture où mémoire et processeurs sont confondus pour constituer une "mémoire dynamique".

## B-3 Principes

La CM2 est un assemblage de processeurs sur lesquels se trouvent la mémoire et le traitement des données. En fonction de la nature du problème à traiter, il est possible de programmer une topologie, sous-topologie de l'hypercube, pour le réseau. De plus la structure est modulaire, afin de pouvoir augmenter facilement la taille de la machine. Elle contient une certaine redondance, afin de pourvoir pallier la défaillance d'un élément.

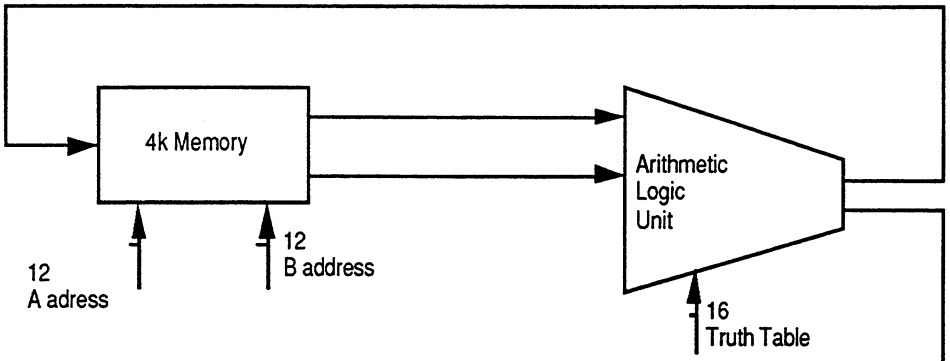
La CM2 est un réseau de processeurs, qui communiquent entre eux par des échanges d'informations gérés par un routeur. A ce titre, elle n'est pas considérée à proprement parler comme une architecture connexionniste. Cependant, son architecture et sa puissance en font un outil de simulation très efficace pour les modèles connexionnistes.

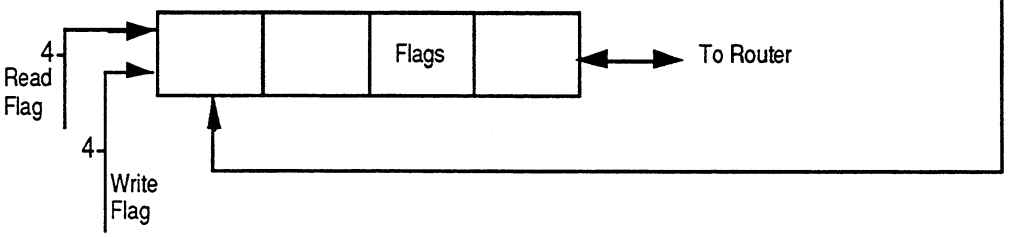
### B-4 Descriptif de la topologie de la machine

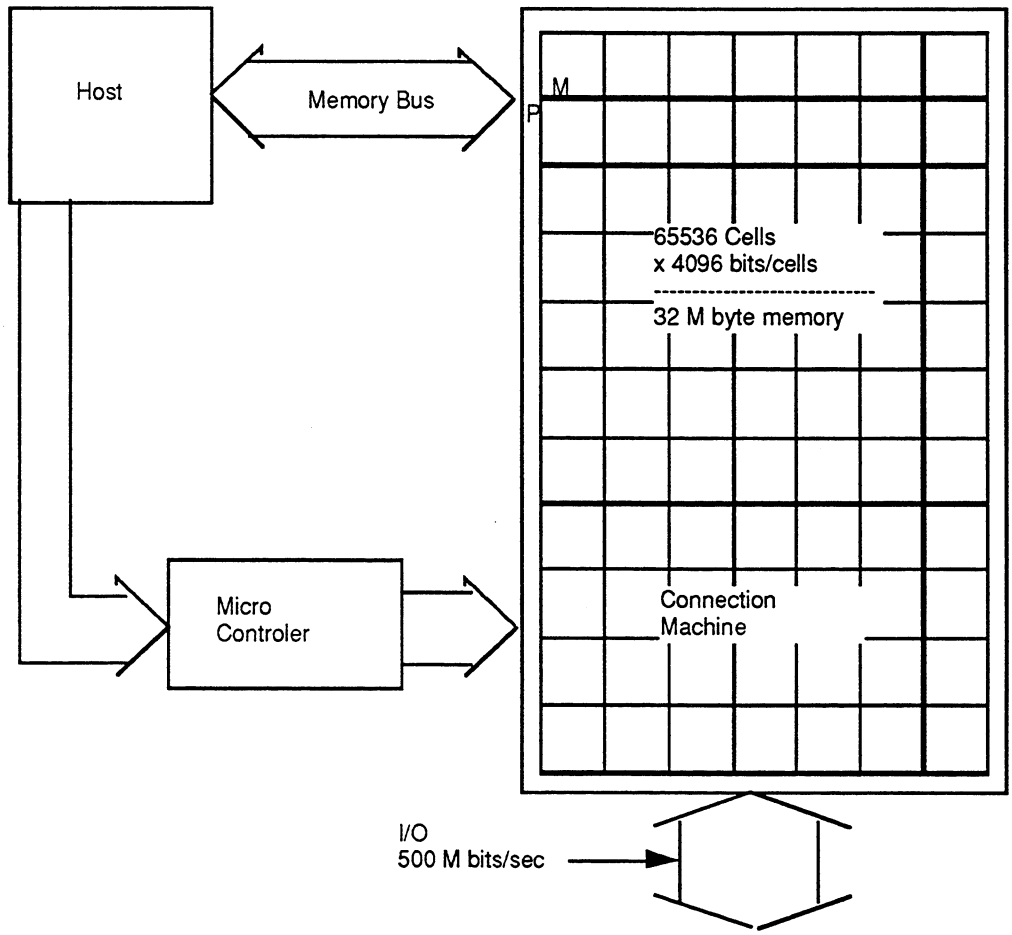
La topologie de la CM2 complète, est celle d'un hypercube de dimension 12 (4096 sommets et 24576 arêtes), où chaque sommet est constitué de 16 processeurs élémentaires 1-bit. Chaque processeur possède une mémoire locale de taille de 4 à 64 Kbits et une unité mathématique et logique, pouvant être programmée pour calculer n'importe laquelle des 256 fonctions booléennes à 16 variables ([DA-1988]). Ces 16 processeurs sont entièrement connectés entre eux par l'intermédiaire d'un réseau Cross-bar. Physiquement, ces 16 processeurs sont rassemblés sur une puce réalisée en technologie VLSI (Very Large Scale Integration), en une grille 4x4. Chaque processeur élémentaire est repéré par rapport à ses voisins sur la base d'un système d'orientation simple "NEWS", suivant la notation de la Thinking Machines Corporation (North, East, West, South).

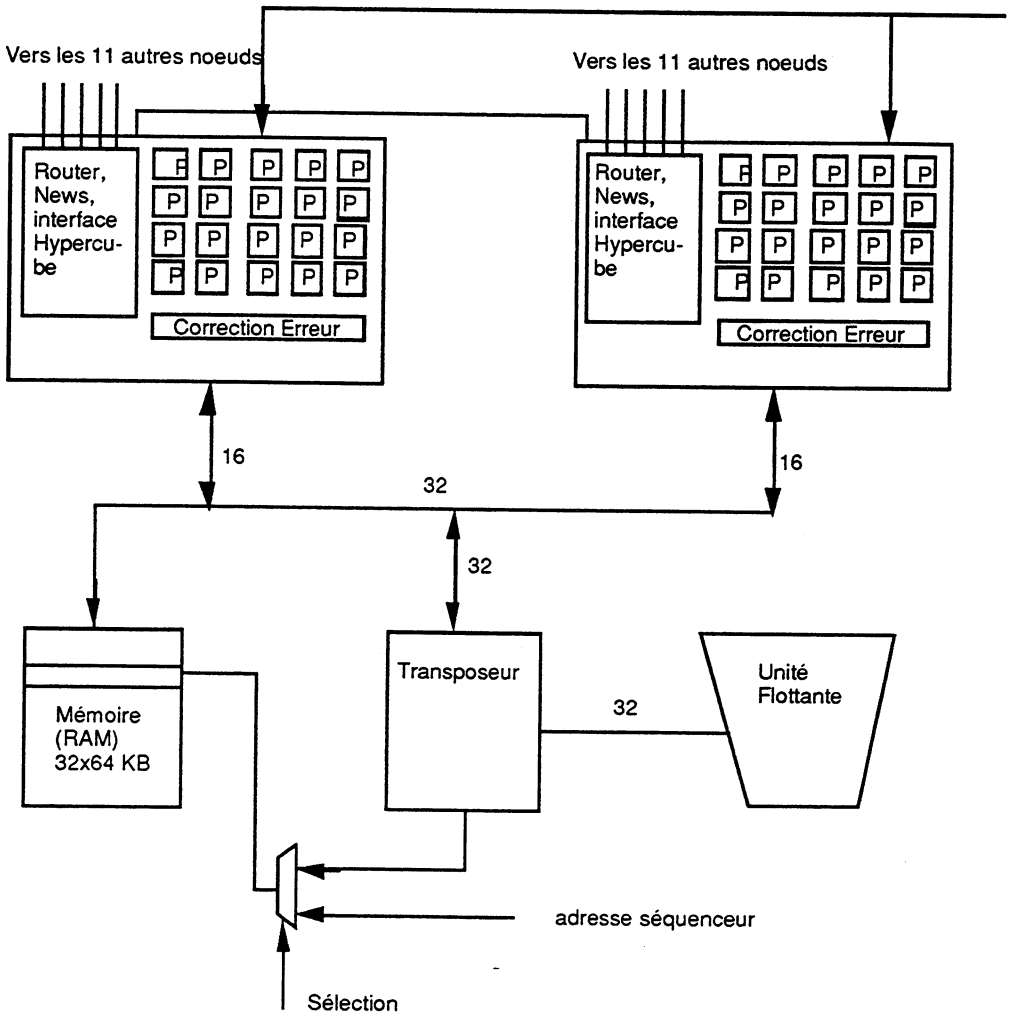
Les processeurs sont associés par 32, à une unité de calcul flottant, de format standard IEEE (23 bits de mantisse, 8 bits d'exposant et 1 bit de signe), ceci afin d'accélérer les calculs. Au lieu d'être executée bit à bit et en parallèle sur chacun des 32 processeurs, une opération portant sur les réels est executée en pipeline sur les 32 mots de 32 bits. Comme l'unité de calcul flottant a besoin de 32 bits provenant d'un même processeur élémentaire, alors que la mémoire ne donne qu'un bit de chacun des processeurs à chaque appel, il existe un circuit de transposition. Ce transposeur est en fait une matrice de 32x32 bits qui sert à stocker et à arranger les opérandes de l'unité flottante (celles-ci rentrent en série et sortent en parallèle). En réalité, il y a un transposeur par entrée de l'unité flottante, c'est-à-dire 3. Ce module de 32 processeurs est relié à l'ordinateur frontal à travers un "séquenceur" par un bus d'instructions et un bus d'adresses. Physiquement, 16 de ces modules constituent une des cartes de la CM2.

De par cette structure en hypercube, la distance entre deux sommets est au plus 12 arêtes.


4k Memory
Arithmetic
Logic
Unit
12
A adress
12
B address
16
Truth Table


4
Read
Flag
4
Write
Flag
Flags
To Router


Host
Memory Bus
M
P
65536 Cells
x 4096 bits/cells
32 M byte memory
Micro
Controler
Connection
Machine
I/O
500 M bits/sec




Organisation d'un module

## B-5 Gestion des processeurs

Le nombre des processeurs réels est fixé par la machine. L'intérêt du parallélisme massif est de donner pour chaque donnée une unité de traitement, un processeur. Par contre, si le nombre des informations dépasse le nombre des processeurs de la machine, il faut pouvoir répartir les données de façon équitable, sur tous les processeurs physiques. Pour ne pas avoir à faire la distinction entre les deux cas cités précédemment, on va définir un processeur **virtuel**. Il correspond à une donnée à traiter mise sur un processeur. Un processeur réel traite donc un ou plusieurs processeurs virtuels. Le nombre ainsi exprimé s'appelle le Virtual Processeur ratio : le **VP-ratio**.

L'un des inconvénients majeurs de cette machine est de ne pouvoir choisir un nombre de processeurs virtuels tel que le VP-ratio soit différent d'une puissance de deux.

Chaque processeur virtuel possède un bit, qui sert de drapeau (“**flag**”). Celui-ci permet d'indiquer si les opérations à effectuer affectent les données que détient le processeur. En effet, comme la machine est gérée par un système SIMD, une opération donnée doit, en principe, être effectuée par tous les processeurs virtuels. Il faut

remarquer qu'aucune autre transformation ne peut intervenir sur les informations d'un **processeur** qui resterait, dit **inactif**, car en pratique l'opération est en fait effectuée, mais le résultat n'est pas stocké. Il est important de minimiser le nombre de processeurs virtuels et le VP-ratio par un choix judicieux de la topologie.

### B-6 Les communications

#### B-6-1 Les communications avec le router

Le constructeur assure que l'envoi de messages par le router (*send-address*) est réalisé de façon quasi-optimale et quasi-sûre. L'utilisateur ne peut pas gérer la façon dont sont effectués ces transferts d'informations. De plus, il lui faut connaître l'adresse exacte de l'envoyeur et du receveur. Ce mode de communication est le plus cher. L'utilisateur doit essayer de trouver la meilleure topologie en fonction de son problème et de l'algorithme qu'il utilise, pour avoir des modes de communications moins coûteux. Ci-dessous, nous présentons quelques modes simples de communication.

#### B-6-2 Les communications par voisinage

Ce système de communication (*get-from-news, send-to-news*) nécessite une topologie en grille pour pourvoir connaître la position relative des processeurs. Les envois se font uniquement d'un processeur à un de ses voisins, suivant un paramètre de déplacement. Ce système peut être utilisé en parallèle. On remarque que la grille étant toujours en boucle, et les processeurs tous actifs, les instructions get-from-news et send-to-news sont équivalentes pour peu que leurs paramètres de déplacement soient opposés. Cependant un *"get"* est à peu près de durée équivalente à deux *"send"*. En effet, dans un *"get"* le processeur doit envoyer un message pour demander à l'autre processeur qu'il attend un message de sa part.

#### B-6-3 Les communications avec combinaison

Ce système consiste en une combinaison puis une redistribution des informations contenues dans un ensemble de processeurs mis en ligne. On peut ainsi effectuer la somme des éléments d'une ligne par sous-ensembles, et avec diffusion à tous les éléments du groupe. Ces types de communications sont au nombre de trois.

***Spread*** *:* Cette instruction effectue une des opérations de calcul élémentaire, sur les éléments actifs d'une ligne, et redistribue à chacun le résultat. Cette opération comme la suivante ne fait pas intervenir l'ordre des processeurs.

***Reduce*** *:* Cette instruction effectue une des opérations de calcul élémentaire, sur les éléments actifs de la ligne, mais ne transmet l'information qu'à un seul des processeurs.

***Scan*** *:* Cette instruction effectue une des opérations de calcul élémentaire, sur les données des processeurs actifs et redonne le résultat au cours du développement le long de la ligne.

Ces trois instructions possèdent des variantes. Il peut y être introduit le concept de segmentation. Dans ce cas l'opération est effectuée indépendamment pour des groupes de processeurs voisins donnés le long de la ligne. Des processeurs (actifs ou inactifs),

fixés jouent le rôle de frontières entre ces sous-ensembles. Pour l'opération scan l'addition peut être effectuée dans un sens ou dans l'autre, les termes "croissant" et "décroissant" sont alors utilisés. De même, il peut être souhaité de donner le résultat de l'opération au processeur de la dernière opérande, ou au processeur de l'opérande précédente, les termes "inclusif" et "exclusif" sont alors employés.

### B-6-4 Les communications avec l'hôte

Les intructions qui régissent les échanges d'informations entre les processeurs et l'hôte sont les suivantes :

***Read-from et Write-to :*** Ces intructions permettent à une variable de l'hôte de prendre pour valeur celle d'un processeur virtuel actif ou inactif de la CM2, et réciproquement d'y placer une valeur.

***Read-from-news-array et Write-to-news-array***: Dans ce cas, ce n'est plus une information, comme précédemment, mais un tableau entier qui est transmis.

***global :*** L'instruction global associée à une opération de combinaison permet de récupérer, dans une variable de l'hôte, la combinaison des valeurs d'un champ de tous les processeurs actifs de la CM2.

### B-4 C/ParIS

La CM2 admet quatre langages usuels : le C/ParIS, le C*, le *Lisp, et le CM-Fortran, qui sont des extensions de langages courants. Le C/ParIS peut être comparé à un assembleur très évolué. Il demande à l'utilisateur une attention plus précise pour contrôler le déroulement des intructions. Cependant, il est bien plus efficace que les autres langages, assez pratique à l'usage et clair même pour un non utilisateur.

Ayant pour idée de comparer l'algorithme utilisé, avec les autres déjà connus, il était intéressant de programmer dans le même langage. Zenios et son équipe ayant travaillés en C*, et en C/ParIS, l'un des deux langages était à retenir. En fait, Zenios nous a envoyé ses codes pour que nous puissions implémenter notre méthode après les modifications nécessaires.

### B-4-1 Les caractéristiques du C/ParIS

Le C/ParIS (Parallel Instruction Set) (cf [C.M.1], [C.M.2]) est un langage qui combine le langage C standard, pour toutes les opérations avec l'ordinateur frontal et un langage de programmation parallèle pour les traitements effectués par la CM2. Le C/ParIS ressemble à un assembleur dans le sens où il est nécessaire de gérer le contrôle des types et le découpage des expressions mathématiques. C'est d'ailleurs le langage de programmation de plus bas niveau auquel un utilisateur a accès sur la CM2. Il ne permet de décrire que la partie parallèle du code, la partie séquentielle étant écrite en C. Mais il est évolué, puisque la gestion de l'occupation mémoire ou celle des processeurs virtuels n'est pas à faire.

Les intructions ParIS sont constituées :

*) d'un ensemble de fonctions qui permettent de gérer les virtuels VP-set, leurs

géométries et la mémoire de ces processeurs.

**) d'un ensemble d'opérations modifiant un champ de destination à partir de différents champs sources. Ces opérations sont soit des opérations arithmétiques, soit des communications.

***) d'un petit nombre de constantes du système qui permettent d'écrire des programmes indépendants de la taille de la machine.

Nous allons donner deux exemples pour illustrer ce langage :

**1 -** Supposons que nous devions additionner à tous les processeurs la valeur 2 au champ x et mettre le résultat dans le champ y , ce qui correspondà l'opération **y = x + 2**, nous aurions alors l'instruction :

**CM_f_add_const_always_3_1L(y, x, 2.0, S, E)**

où **S** représente la mantisse et **E** l'exposant. Ceci s'analyse comme suit :

**CM** : ceci est une opération à effectuer sur la Connection Machine

**f**: nous travaillons sur des réels
**add**: c'est une addition
**const** : la dernière opérande est une constante
**always** : ceci sera effectué sur tous les processeurs actifs ou non.
**3**: trois opérandes sont en jeu
**1L**: la taille des champs est la même pour les trois opérandes, sinon il aurait fallu ajouter les paramètres de longueur de chacun des champs.

**2 -** Supposons que nous voulions maintenant donner la valeur 2 à y, uniquement sur les processeurs dont la valeur du champ x est négative, nous obtenons :

```
CM_f_lt_zero_1L(x, S, E);
CM_logand_context_with_test ();
CM_f_move_constant_1L(y, 2.0, S, E);
CM_load_context (context-initial);
```

Dans la première instruction, un test est effectué sur la valeur du champ du processeur. Le bit "drapeau test" des processeurs est mis à 0 ou 1 en fonction du résultat du test appliqué.

Dans la deuxième instruction, ce bit est évalué en fonction du contexte courant, bit drapeau actif/inactif. A partir de-là, le contexte est modifié pour toutes les instructions qui suivent.

Dans la troisième instruction, le terme **move** indique un changement de la valeur d'un champ.

La dernière inntruction permet de retrouver le contexte initial, appelé ici pour l'exemple context-initial.

## 5-5 Application de l'algorithme de décomposition proximale problème de transport

Nous examinons dans ce paragraphe la décomposition distribuée du problème général de transport. Nous voulons dire par là qu'à chaque itération, les calculs sont fait en parallèle sur tous les arcs (origine/destination), contrairement à une approche similaire, due à Zenios(1991), dans laquelle chaque processeur traite toute une ligne de la matrice associée au problème. Nous avons besoin d'un processeur pour chaque élément non nul de cette matrice , i.e. 3pq processeurs, où p est le nombre de sommets origines et q le nombre de destinations.

Le modèle qui nous intéresse dans ce travail est un problème de transport non linéaire avec des capacités sur les arcs. Soit $\mathcal{O}$ un ensembles d'origines et $\mathcal{D}$ un ensemble de destinations et soient $p=|\mathcal{O}|$ et $q=|\mathcal{D}|$. On note par $\mathcal{A}$ l'ensemble des arcs (i,j), où $i\in\mathcal{O}$ et $i\in\mathcal{D}$ :

$$(\mathbf{T}) \quad \begin{cases} \text{Minimiser} \sum_{i,j \in A} f_{ij}(x_{ij}) \\ \sum_{j\in J_i} x_{ij} \le s_i, \forall\, i\in\mathcal{O} \\ \sum_{i\in I_j} m_{ij}x_{ij}=d_j, j\in\mathcal{D} \\ 0 \le x_{ij} \le c_{ij}, (i,j) \in \mathcal{A} \end{cases}$$

Chaque fonction $f_{ij}$ est dans $\Gamma^0(\mathbb{R})$ puisqu'elles dont quadratiques, les $s_i$ représentent les stocks disponibles en chaque sommet origine et les $d_j$ les demandes en chaque sommet destination. Chaque $x_{ij}$ représente la quantité des biens transportés entre i et j et elle est bornée par une capacité fixe $c_{ij}$.

L'idée est de construire un dual séparable de (T) se basant sur les résultats de Rockafellar(1970c) et ceux que nous présentons dans l'annexe de ce document. Le problème dual est par conséquent construit à partir d'une bifonction séparable, où les variable de perturbation sont dans un sous-espace particulier pour retrouver le problème originel.

Dans une seconde étape, nous donnons une formulation primal-duale dans le but d'appliquer une décomposition proximale (§ Chapitre 3).

Nous allons utiliser la notion de dualité séparable comme nous l'avons décrite dans l'annexe, afin d'obtenir un sous-problème différent pour chaque élément non nul de la matrice de transport. Par conséquent, chaque $x_{ij}$ apparait trois fois dans le modèle. La première fois dans la fonction objectif, puis dans les blocs de contraintes origines et destinations.

*) Nous créons alors trois copies de chaque variable, notées respectivement $x^f_{ij}$, $x^o_{ij}$ et $x^d_{ij}$ de telle sorte que la variable globale soit dans le sous-espace suivant :

$$L=\{x=(x^f, x^o, x^d)\in \mathbb{R}^{3pq} \mid x^f=x^o=x^d\}$$

Nous obtenons un problème primal perturbé en ajoutant :

*) Une perturbation $u^o_{ij}$ pour chaque arc ayant pour origine le sommet i et une perturbation $u^d_{ij}$ pour tout arc ayant pour destination un sommet j de sorte que la variable de perturbation soit dans le sous-espace suivant :

$$A = \left\{u = (u^o, u^d) \in \mathbb{R}^{2pq} \mid \sum_j u^o_{ij} = 0, \forall i, \sum_i u^d_{ij} = 0, \forall j\right\}$$

Soient $s_{ij}$ et $d_{ij}$ des allocations fixes telles que

$$\sum_j s_{ij} = s_i, \quad \forall i\in \mathcal{O} \text{ et } \sum_i d_{ij} = d_j, \quad \forall j\in \mathcal{D}$$

La bifonction Fu associée définie, pour $u=(u^o,u^d)\in \mathbb{R}^{2pq}$ sur le sous-espace L est donnée par :

$$(Fu)(x) = \begin{cases} \sum_{ij} f_{ij}(x^f_{ij}) \text{ si } x^o_{ij} \le s_{ij} + u^o_{ij} \\ \qquad\qquad m_{ij}\, x^d_{ij} = d_{ij} + u^d_{ij} \\ \qquad\qquad 0 \le x^o_{ij} \le c_{ij} \\ \qquad\qquad 0 \le x^d_{ij} \le c_{ij} \\ +\infty \text{ sinon} \end{cases}$$

Les perturbations doivent être dans le sous-espace A pour retrouver le problème de départ. Le problème (T) est équivalent au suivant :

$$\mathrm{Inf}_{x\in L, u\in A}(Fu)(x)$$

Soient

*) $u^{*f}$, $u^{*o}$ et $u^{*d}$ les variables duales du problème. Elles sont obtenues par copies des variables duales originales et doivent être dans le sous-espace orthogonal de A.

*) $x^{*f}$, $x^{*o}$ et $x^{*d}$ les perturbations duales ; elles doivent être dans le sous espace orthogonal de L.

En appliquant l'algorithme ADPG1, traité au Chapitre précédent, à ce problème, nous obtenons les trois sous-problèmes, $S^f_{ij}$, $S^o_{ij}$ et $S^d_{ij}$ ci-dessous, pour chaque arc (i,j)

$$(s^f_{ij}) \begin{cases} \mathrm{Inf}_{x^f_{ij}} \left\{ f_{ij}(x^f_{ij}) + \frac{1}{2\lambda}(x^f_{ij} - (\overline{x}^f_{ij} + \lambda \overline{x^*}^f_{ij}))^2 \right\} \\ 0 \le x^f_{ij} \le c_{ij} \end{cases}$$

$$(s^o_{ij}) \begin{cases} \mathrm{Inf}_{(x^0_{ij},\, u^0_{ij})} (x^o_{ij} - (\overline{x}^o_{ij} + \lambda \overline{x^*}^o_{ij}))^2 + (u^o_{ij} - (\overline{u}^o_{ij} + \lambda \overline{u^*}^o_{ij}))^2 \\ x^o_{ij} \le s_{ij} + u^o_{ij} \\ 0 \le x^o_{ij} \le c_{ij} \end{cases}$$

$$(s^d_{ij}) \begin{cases} \mathrm{Inf}_{(x^d_{ij},\, u^d_{ij})} (x^d_{ij} - (\overline{x}^d_{ij} + \lambda \overline{x^*}^d_{ij}))^2 + (u^d_{ij} - (\overline{u}^d_{ij} + \lambda \overline{u^*}^d_{ij}))^2 \\ x^d_{ij} \le d_{ij} + u^d_{ij} \\ 0 \le x^d_{ij} \le c_{ij} \end{cases}$$

Remarquons que ces sous-problèmes sont faciles à résoudre et sont à une ou deux dimensions. Les variables sont alors facilement calculables grâce aux relations suivantes :

$$x^f_{ij} + \lambda x^{*f}_{ij} = \overline{x}^f_{ij} + \lambda \overline{x^*}^f_{ij}$$

$$x^0_{ij} + \lambda x^{*0}_{ij} = \overline{x}^0_{ij} + \lambda \overline{x^*}^0_{ij}$$

$$u^0_{ij} + \lambda u^{*0}_{ij} = u^0_{ij} + \lambda \overline{u^*}^0_{ij}$$

$$x^d_{ij} + \lambda x^{*d}_{ij} = \overline{x}^d_{ij} + \lambda \overline{x^*}^d_{ij}$$

$$u^d_{ij} + \lambda u^{*d}_{ij} = \overline{u}^d_{ij} + \lambda \overline{u^*}^d_{ij}$$

Finalement, les projections sur les sous-espaces sont données pour chaque arc $(i,j) \in \mathcal{A}$ par :

$$\overline{\overline{x}}^f_{ij} = \overline{\overline{x}}^0_{ij} = \overline{\overline{x}}^d_{ij} = \frac{1}{3}(x^f_{ij} + x^0_{ij} + x^d_{ij})$$

$$\overline{\overline{x^*}}^f_{ij} = x^{*f}_{ij} - \frac{1}{3}(x^{*f}_{ij} + x^{*0}_{ij} + x^{*d}_{ij})$$

$$\overline{\overline{x^{*0}_{ij}}} = x^{*0}_{ij} - \frac{1}{3}(x^{*f}_{ij} + x^{*0}_{ij} + x^{*d}_{ij})$$

$$\overline{\overline{x^{*d}_{ij}}} = x^{*d}_{ij} - \frac{1}{3}(x^{*f}_{ij} + x^{*0}_{ij} + x^{*d}_{ij})$$

$$\overline{\overline{u^{0}_{ij}}} = u^{0}_{ij} - \frac{1}{|J_i|}\sum_{j \in J_i} u^{0}_{ij}$$

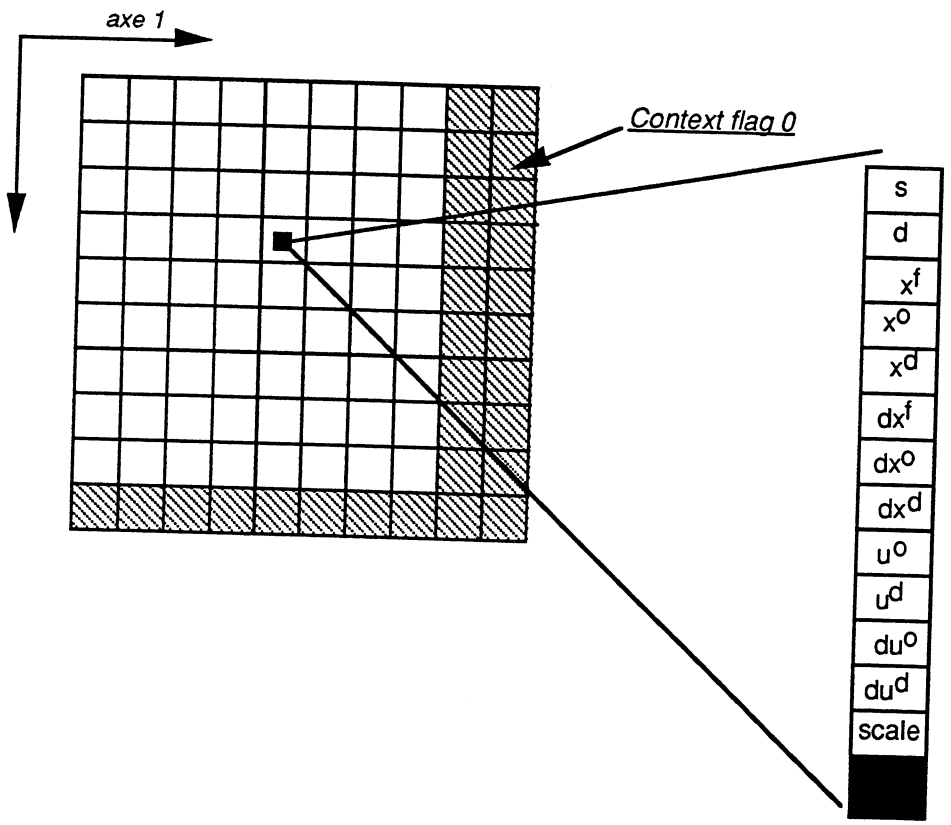
$$\overline{\overline{u^{d}_{ij}}} = u^{d}_{ij} - \frac{1}{|I_j|}\sum_{i \in I_j} u^{d}_{ij}$$

$$\overline{\overline{u^{*o}_{ij}}} = \frac{1}{|J_i|}\sum_{j \in J_i} u^{*0}_{ij}$$

$$\overline{\overline{u^{*d}_{ij}}} = \frac{1}{|I_j|}\sum_{i \in I_j} u^{*d}_{ij}$$

La notation $\overline{\overline{x}}$ représente la nouvelle estimation de x après projection.

Nous avons implémenté sur la Connection Machine (CM2) cette adaptation de l'algorithme de décomposition proximale au problème de transport avec un coût quadratique sur un jeu de données utilisé par Zenios(1991a) et qu'il appelle dans son travail par "tseng1". Ce dernier nous a également fourni le code d'un programme basé sur une méthode de relaxation par les lignes (row-action) que nous avons comparée à la décomposition proximale. Nous nous sommes inspirés de ce code qui est mieux adapté à l'algorithme qu'il a utilisé. Nous nous sommes interessé pour le moment à l'imlémentation dense, l'implémentation creuse sera l'objet d'un travail en cours d'élaboration. La figure suivante représente la configuration en grille du réseau de processeurs ainsi que le partitionnement de la mémoire de chaque processeur. Le nombre de processeurs actifs est 5063, qui est exactement le nombre d'arcs.




Configuration en grille "NEWS" et partitionnement de la mémoire pour 'implémentation dence de l'algorithme ADPG1 applique au problème de transport avec un coût quadratique

Ci-dessous les étapes importantes du module correspondant à l'algorithme que nous avons appliqué.

**Proximal step**

```
/* Scaling of bounds : xf = (xf/lambd - cf2 + dxf) / (2*cf1 + 1/lambd) */
        CM_f_divide_3_1L( xf, x, lambd, S, E);
        CM_f_subtract_2_1L( xf, cf2, S, E);
        CM_f_add_2_1L( xf, dxf, S, E);
        CM_f_divide_2_1L( xf, iwl, S, E);

/* test on the value of xf */
        . . .

/* scaling of dxf = (lambd*dxf + x - xf) /lambd */
        CM_f_multiply_2_1L( dxf, lambd, S, E);
        CM_f_add_2_1L( dxf, x, S, E);
        CM_f_subtract_2_1L( dxf, xf, S, E);
        CM_f_divide_2_1L( dxf, lambd, S, E);

/* Scaling of the origin nodes : xo = (xo + s + uo + lambd*(dxo + duo)) /2 */
        CM_f_add_3_1L( xo, dxo, duo, S, E);
        CM_f_multiply_2_1L( xo, lambd, S, E);
```

```
            CM_f_add_2_1L( xo, x, S, E);
            CM_f_add_2_1L( xo, s, S, E);
            CM_f_add_2_1L( xo, uo, S, E);
            CM_f_divide_constant_2_1L( xo, 2.0, S, E);

/* test on the value of xo */
            . . .

/* scaling of uo = xo - sij */
            CM_f_move_1L( scale, uo, S, E);
            CM_f_subtract_3_1L( uo, xo, s, S, E);

/* scaling of dxo = (lambd*dxo + x - xo) /lambd */
            CM_f_multiply_2_1L( dxo, lambd, S, E);
            CM_f_add_2_1L( dxo, x, S, E);
            CM_f_subtract_2_1L( dxo, xo, S, E);
            CM_f_divide_2_1L( dxo, lambd, S, E);

/* scaling of duo = (lambd*duo + uo_old - uo) /lambd */
            CM_f_multiply_2_1L( duo, lambd, S, E);
            CM_f_add_2_1L( duo, scale, S, E);
            CM_f_subtract_2_1L( duo, uo, S, E);
            CM_f_divide_2_1L( duo, lambd, S, E);

/* Scaling of the destination nodes : xd = (xd + d + ud + lambd*(dxd + dud)) /2 */
            CM_f_add_3_1L( xd, dxd, dud, S, E);
            CM_f_multiply_2_1L( xd, lambd, S, E);
            CM_f_add_2_1L( xd, x, S, E);
            CM_f_add_2_1L( xd, d, S, E);
            CM_f_add_2_1L( xd, ud, S, E);
            CM_f_divide_constant_2_1L( xd, 2.0, S, E);

/* test on the value of xd */
            . . .

/* scaling of ud = xd - dij */
            CM_f_move_1L( scale, ud, S, E);
            CM_f_subtract_3_1L( ud, xd, d, S, E);

/* scaling of dxd = (lambd*dxd + x - xd) /lambd */
            CM_f_multiply_2_1L( dxd, lambd, S, E);
            CM_f_add_2_1L( dxd, x, S, E);
            CM_f_subtract_2_1L( dxd, xd, S, E);
            CM_f_divide_2_1L( dxd, lambd, S, E);

/* scaling of dud = (lambd*dud + ud_old - ud) /lambd */
            CM_f_multiply_2_1L( dud, lambd, S, E);
            CM_f_add_2_1L( dud, scale, S, E);
            CM_f_subtract_2_1L( dud, ud, S, E);
            CM_f_divide_2_1L( dud, lambd, S, E);
```

**Projection step**

```
/* Dual variables  dxi = dxi - ((dxf + dxo + dxd) /3)*/
            CM_f_add_3_1L( scale, dxf, dxo, S, E);
            CM_f_add_2_1L( scale, dxd, S, E);
            CM_f_divide_constant_2_1L( scale, 3.0, S, E);
            CM_f_subtract_3_1L( dxf, dxf, scale, S, E);
            CM_f_subtract_3_1L( dxo, dxo, scale, S, E);
            CM_f_subtract_3_1L( dxd, dxd, scale, S, E);
```

```
/* Primal variables  xf = xo = xd = (xf + xo + xd) /3 */
        CM_f_add_3_1L( scale, xf, xo, S, E);
        CM_f_add_2_1L( scale, xd, S, E);
        CM_f_divide_constant_2_1L( scale, 3.0, S, E);CM_f_move_1L( xf, scale, S, E);
        CM_f_move_1L( xo, scale, S, E);
        CM_f_move_1L( xd, scale, S, E);

/* Primal perturbation variables */
        CM_f_move_1L( scale, uo, S, E);
        CM_spread_with_f_add_1L( scale, scale, axis_news[1], S, E);
        CM_f_divide_2_1L( scale, compts, S, E);
        CM_f_subtract_2_1L( uo, scale, S, E);
        CM_f_move_1L( scale, ud, S, E);
        CM_spread_with_f_add_1L( scale, scale, axis_news[0], S, E);
        CM_f_divide_2_1L( scale, compid, S, E);
        CM_f_subtract_2_1L( ud, scale, S, E);

/* Dual perturbation variables */
        CM_spread_with_f_add_1L( duo, duo, axis_news[1], S, E);
        CM_f_divide_2_1L( duo, compts, S, E);
        CM_spread_with_f_add_1L( dud, dud, axis_news[0], S, E);
        CM_f_divide_2_1L( dud, compid, S, E);
```

Pour cet exemple, l'algorithme que nous avons proposé converge, ce qui est rassurant, mais il effectue plus d'itérations. Nous donnons à titre d'exemple les résultats suivants :

Informations sur l'exemple et la machine

```
=======> NONLINEAR TRANSPORTATION CODE <=======
Uncap.Transp.Problem  1000 nodes ; 5063 arcs ;

****** Rows of adjacency matrix = 500
****** Columns of adjacency matrix = 500
****** NEWS grid size set at 512 x 512 ******
****** VP Ratio 32 ******
Reading data ...
Creating geometry ...
Initializations ...
****** Memory requirements 20 32-bit words ******
****** Number of active VP 5063 ******
****** Start Main Algorithm ******
Calibrating CM timer...Done. CM speed = 7.00 MHz
```

Résultats donnés par l'algorithme de Zenios après 1620 itérations

```
 ****** Timing information ******
Real time: 44.30 sec; CM time: 32.74 sec; Front end virtual time: 27.04 sec;
CM utilization: 74%; Front end utilization: 61%
****** Final Error Check on the CM ******
   x Max = 2825.33057
   x min = 0.75007
   d Max = 2825.33008
   d min = 0.75000
----> Max. row error = 0.04 percent.

   x Max = 2784.16016
   x min = 0.01000
   d Max = 2784.15991
   d min = 0.01000
----> Max. column error = 0.00 percent.

   xij Max = 794.19012
   xij min = -0.00195
----> Max. bound  error = 0.18 percent.
****** Quadratic function value:  243201920.000000 ******
```

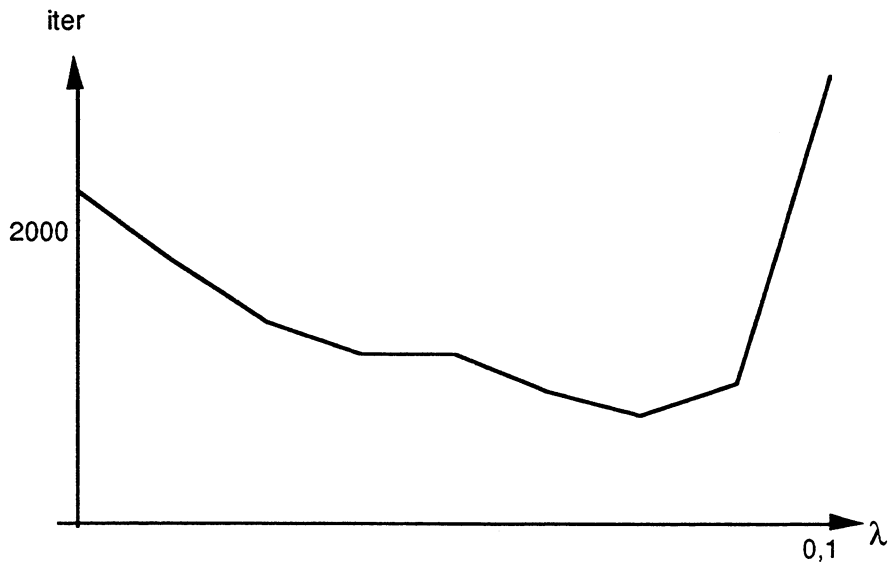
Résultat donné par notre algorithme après 4000 itérations

```
 ***** Timing information *****
Real time: 545.04 sec; CM time: 540.94 sec; Front end virtual time: 522.98 sec;
CM utilization: 99%; Front end utilization: 96%
***** Final Error Check on the CM *****
 lambd  = 0.09000
 cf1  = 5.00000
 upp Max = 99999.000
 xij Max = 794.190
 xij min = 0.000
----> Max. bound  error = 0.00 percent.

 sij Max = 527.397
 sij min = 0.075
  compts = 66.000
  compts = 3.000
   x Max = 2825.330
   x min = 0.750
   s Max = 2825.330
   s min = 0.750
----> Max. row error = 0.00 percent.

 dij Max = 383.917
 dij min = 0.002
  comptd = 21.000
  comptd = 2.000
   x Max = 2784.160
   x min = 0.010
   d Max = 2784.160
   d min = 0.010
----> Max. column error = 0.22 percent.
```

Lorsque nous relaxons le test d'arret, nous obtenons pour différentes valeurs de $\lambda$ la figure suivante :




La lenteur de l'algorithme de décomposition proximale est due à des problèmes d'echelle dans les données. En effet, certaines demandes sont de l'ordre de $10^{-2}$ alors que d'autres sont de l'ordre de $10^{3}$.

Il semble nécessaire de poursuivre la mise au point du code afin qu'il soit mieux adapté à l'algorithme (ADPG).

# Annexe :

## Quelques notions sur la dualité symétrique

## A-1 Rappel sur les bifonctions

Dans le chapitre 4 portant sur la décomposition, nous avons vu qu'il y a plusieurs techniques dont on se sert afin de rendre le problème séparable. En effet, on arrive à éclater les contraintes de couplage en rajoutant éventuellement des variables supplémentaires, que nous notons dans ce chapitre par u (ou v). Afin d'établir la dépendance entre la fonction objective d'un problème de minimisation convexe et un vecteur correspondant à une perturbation de ce dernier, R.T.Rockafellar(1970c) a défini la notion de bifonction.

**Définition A-1** : Une bifonction F "de $\mathbb{R}^m$ dans $\mathbb{R}^n$" est définie comme étant une application faisant correspondre à tout $u \in \mathbb{R}^m$ un fonction Fu sur $\mathbb{R}^n$ ayant ses valeurs dans $[-\infty , +\infty]$ :

$$\mathbb{R}^m \longrightarrow 2^{(\mathbb{R}^n)}$$

$$u \longmapsto [\, Fu : \mathbb{R}^n \longrightarrow \bar{\mathbb{R}}]$$

Fu étant l'image de u par la bifonction F.
La fonction définie sur $\mathbb{R}^m \times \mathbb{R}^n$ par :

$$(u,x) \rightarrow (Fu)(x)$$

est appelée la fonction graphe de la bifonction F.

Une bifonction peut alors être vue comme la première étape d'une application à deux étapes ; la première étant de $\mathbb{R}^m$ dans $\mathbb{R}^n$ et la seconde de $\mathbb{R}^n$ dans $\mathbb{R} \cup \{\infty\}$. On peut donc facilement voir qu'on a une correspondance bijective entre l'ensemble de bifonctions de $\mathbb{R}^m$ dans $\mathbb{R}^n$ et l'ensemble des fonctions sur $\mathbb{R}^{m+n}$ .

**Définition A-2 :**

**1 -** Une bifonction est dite convexe, propre ou fermée si et seulement si sa fonction graphe est convexe, propre ou fermée respectivement.

**2-** Le *G-domaine* d'une bifonction F est le *domaine de sa fonction graphe.*

Le *domaine de la bifonctionf F*, noté domF, est défini par :

$$domF = \{ u : / Fu \not\equiv +\infty \}.$$

Pour plus de détails voir R.T.Rockafellar.

## A-2 Bifonctions et optimisation convexe

Soit ($\mathcal{P}$)un problème d'optimisation convexe ayant la forme suivante

$$\begin{cases} \min f_0(x) \\ G(x) \leq 0 \end{cases} \qquad (\mathcal{P})$$

où G et $f_O$ sont des fonctions propres convexes sur X ($=\mathbb{R}^n$), et séparable par rapport à la partition $X_1, ..., X_r$, ($X_j = \mathbb{R}^{n_j}$, j=1, ..., r) et $n_1+...+n_r=n$. G est donc de la forme suivante :

$$G(x) = \sum_{j=1}^{r} g_j(x_j)$$

où, $x_j \in X_j$.

On suppose que la fonction $f_0$ est aussi séparable, par rapport à la même partition que G, i.e.

$$f_0(x) = \sum_{j=1}^{r} f_{0j}(x_j)$$

Où les fonctions $f_{0j}$ et $g_j$, $1 \le j \le r$, sont propres convexes fermées définies respectivement sur les espaces $X_j$ correspondants.

Ce problème peut être perturbé de plusieurs manières. Selon Rockafellar (1970c), a perturbation pour laquelle le problème perturbé coïncide avec le problème ($\mathcal{P}$), apprtenant au sous-espace trivial {0} de $\mathbb{R}^r$. Nous nous interessons au cas où la perturbation est dans un sous-espace vectoriel non trivial de l'espace des perturbations. Nous signalons que ce dernier type de perturbation a une grande utilité, dans beaucoup de situations, notamment si les contraintes couplantes du problème de départ sont séparables comme dans le problème ($\mathcal{P}$) ci-dessus.

Nous allons nous restreindre au cas d'une seule contrainte afin de simplifier les notations. Le cas de plusieurs contraintes peut être traité d'une façon analogue sans rien changer aux résultats que nous présentons ici.

Vue la structure du problème ($\mathcal{P}$), nous pourrons lui faire correspondre deux bifonctions ;

$$\Phi : \mathbb{R} \to (\mathbb{R}^n \to \mathbb{R})^{\Phi v}$$

et

$$\Psi : \mathbb{R}^r \to (\mathbb{R}^n \to \mathbb{R})^{\Psi u}$$

Ces deux bifonctions correspondent aux deux problèmes perturbés suivants :
Le premier problème est :

$$\begin{cases} \min f_0(x) \\ G(x) \le v \end{cases} \qquad (\mathbf{R_v})$$

qui coincide avec le problème ($\mathcal{P}$) quand v = 0 (cf R.T.Rockafellar [R ]).
Le second problème est :

$$\begin{cases} \min f_0(x) \\ g_j(x) \le u_{ij} \ ; \ 1 \le j \le r \end{cases} \qquad (S_u)$$

qui coïncide avec le problème (P) pour un certain $u = (u_j)\ 1 \le j \le r$, appartenant au sous-espace défini par :

$$A = \left\{ u \in \mathbb{R}^r : \sum_{j=1}^{r} u_j = 0 \right\}$$

Les deux bifonctions $\Phi v$ et $\Psi u$ sont donc définies comme suit :

$$(\Phi v)(x) = R_0(v,x) + R_1(v,x)$$

où les les fonctions $R_i$ sont définies sur $\mathbb{R} \times \mathbb{R}^n$ par :

$$R_0(v,x) = f_0(x)$$

$$R_1(v,x) = \chi(x \mid \{x : G(x) \le v\}). \qquad (1^{**})$$

et

$$(\Psi u)(x) = k_0(u,x) + \sum_{j=1}^{r} k_j(u_j, x_j)$$

où les fonctions $k_j$ sont définies sur $\mathbb{R}^t \times \mathbb{R}^n$ par :

$$k_0(u,x) = f_0(x) \ ;$$

$$k_j(u,x_j) = \chi(x_j \mid \{x_j : g_j(x_j) \le u_j\}) \ ; \ 1 \le j \le r. \qquad (2^{**})$$

où $\chi(x|S(v))$ est la fonction indicatrice de l'ensemble S(v) .

Pour une bifonction F nous rappelons la notation de R.T.Rockafellar[ ] suivante :

$$(\inf F)(u) = \inf Fu = \inf_x (Fu)(x)$$

Si A est un sous-espace vectoriel de l'espace des perturbations, nous ajoutons la notation suivante :

$$(\inf F)_A(u) = \inf_{u \in A} (\inf F)(u) = \inf_{(u \in A,\, x)} (Fu)(x)$$

Par construction des deux bifonctions $\Phi$ et $\Psi$, nous avons :

$$(\inf \Psi)_A(u) = (\inf \Phi)(0) = \alpha$$

où $\alpha$ est la valeur optimale de $(\mathcal{P})$

**Théorème A-3**:

Les deux bifonctions $\Phi v$ et $\Psi u$ sont convexes, ainsi que les fonctions inf $\Psi u$ et inf $\Phi v$ sur leur espace de définition respectifs et si la valeur optimale de $(\mathcal{P})$ est finie, alors il existe

$$u^* \in A \text{ et } w^* \in A^{\perp} \qquad \textbf{(A1)}$$

tels que :

$$(\inf \Psi)(u^*) = (\inf \Phi)(0) = \alpha \qquad \textbf{(A2)}$$

et

$$-w^* \in \partial(\inf \Psi)(u^*) \qquad \textbf{(A3)}$$

De plus, si on note $\gamma$ un vecteur de Kuhn-Tucker associé à $(\mathcal{P})$ alors

$$w^* = (\gamma, \gamma, ..., \gamma) \in \mathbb{R}^r.$$

**Preuve :** La convexité des fonctions de perturbation inf$\Phi$ et inf$\Psi$ est due au même raisonnement que dans ( R.T.Rockafellar[ ] Th 29-1). Nous savons, d'une part, que d'après ce même théorème, si $\gamma$ est un vecteur de Kuhn-Tucker pour le problème $(\mathcal{P})$, alors $-\gamma$ est un sous-gradient de (inf$\Phi$) en 0 donc

$$(\inf \Phi)(\zeta) \geq (\inf \Phi)(0) + \langle \gamma, \zeta \rangle \quad \text{pour tout } \zeta \in \mathbb{R} \qquad \textbf{(A4)}$$

D'après la forme des deux bifonctions, on en déduit que :

$$(\inf \Psi)(u) \geq (\inf \Phi)(\zeta(u)) \qquad \textbf{(A5)}$$

Où $u = (u_1,...,u_r)$ et $\zeta(u)$ est défini par :

$$\zeta(u) = \sum_{j=1}^{r} u_j$$

Pour montrer la relation (A5), il suffit de remarquer que

$$\bigcup_{j=1}^{r} \{x | g_j(x_j) \leq u_j\} \subseteq \left\{x | \sum_{i=1}^{r} g_i(x_i) \leq \sum_{j=1}^{r} u_j\right\}$$

D'après les relations (A4) et (A5), on déduit que :

$$(\inf \Psi)(u) \geq (\inf \Phi)(0) + \langle \gamma, \zeta(u) \rangle \quad \text{pour tout } u \in \mathbb{R}^t$$

D'autre part, si la valeur optimale $\alpha$ de $(\mathcal{P})$ est finie, alors il existe u* élèment de A tel que

$$(\inf \Psi)(u^*) = (\inf \Phi)(0) = \alpha.$$

On peut alors écrire que :

$$(\inf \Psi)(u) \geq (\inf \Psi)(u^*) + < \gamma , \zeta(u) >, \text{ pour tout } u \in \mathbb{R}^r \quad \textbf{(A6)}$$

Si nous définissons w* comme dans l'énoncé du théorème A-1-3, la relation (A6), ci-dessus est équivalente à :

$$(\inf \Psi)(u) \geq (\inf \Psi)(u^*) + < w^* , u >, \text{ pour tout } u \in \mathbb{R}^r \quad \textbf{(A7)}$$

or $u^* \in A$ donc

$$< w^* , u^* > = 0$$

et :

$$(\inf \Psi)(u) \geq (\inf \Psi)(u^*) + < w^* , u - u^* > \text{ pour tout } u \in \mathbb{R}^r$$

Nous pouvons ainsi dire que la relation (A3) est vraie, i.e. :

$$-w^* \in \partial(\inf \Psi)(u^*)$$

**CorollaireA-4 :**

Supposons que la valeur optimale de (P) est unique, alors :

(P) admet un vecteur de Kuhn-Tucker unique $\alpha^*$ si et seulement si la bifonction $\Psi$ est différentiable en $u = u^*$. Dans ce cas, nous avons :

$$-\alpha^* = \frac{\partial}{\partial u_i} (\inf \Psi)|_{u=u^*}$$

**Preuve :** Ce résultat peut être déduit facilement du théorème précédent et de R. T. Rockafellar [ ], théorème 25-1.

# Conclusion

Nous pensons avoir fait le lien dans ce travail entre certains résultats théoriques et leur utilité dans les applications numériques. C'est d'autant plus clair dans le cas de la méthode proximale qui s'avère extrèmement sensible à la modélisation de certains problèmes et à l'introduction de paramètres d'accélération de la convergence. Plus précisément en ce qui concerne la méthode de l'inverse partiel, si la contribution de Spingarn reste fondamentale du point de vue théorique, les qualités numérique n'ont été que peu observées, si ce n'est dans les travaux de Michelot sur les problèmes de localisation. Le rôle du paramètre $\lambda$ introduit pour accélérer la convergence est maintenant éclairci ainsi que le lien avec la vieille méthode de Douglas-Rachford. Dans ce dernier sens, on on peut considérer notre étude comme une suite aux travaux essentiels développes en France depuis J.J. Moreau, en particulier ceux de B. Martinet et de P.L. Lions.

On peur considérer que notre étude est essentiellement inspirée des travaux récents de Spingarn tout en faisant le lien avec ceux plus anciens de Martinet et Lions.

D'autres méthodes comme la méthode proximale projetée nous semblent aujourd'hui plus attrayantes et même si le choix de la suite des paramètres reste encore empirique, c'est une alternative raisonnable dans certains cas.

Les méthodes de décomposition étudiées dans ce travail caractérisent l'évolution actuelle vers une coordination distribuée et régularisée. Les pionniers sont encore apparus en France, Martinet, Lions-Mercier, Pierra et autres et l'ecole américaine dominée par l'équipe du MIT de D. Bertsekas a pris le relais en mettant l'accent sur la parallélisation et les applications aux problèmes d'optimisation dans les réseaux. L'implémentation massivement parallèle de ces méthodes reste encore très délicate et l'évolution du matériel vers le (MPMD) (CM-5) devance visiblement les efforts d'adaptation des algorithmes. Nous estimons ces premiers résultats obtenus à Grenoble sur la CM-2 encourageants et prévoyons que l'importance des algorithmes de décomposition proximale devra croître dans les années qui viennent.

# Bibliographie

P. **Alart** et B. **Lemaire(1991)**, " Penalization in non classical convex programming via variational convergence", Mathematical Programming, vol. 51, N° 3, pp 307-332.

K.J. **Arrow** et L. **Hurwics(1959)**, " Decentralization and computation in resource allocation", dans Essay in Economics and Econometrics, ed. R.W. Pfonts, University of California Press, Los Angeles.

H. **Attouch (1984)**, Variational Convergence for Functions and Operators", Appl. Math. series, Pitman, London

J. P. **Aubin** et I. **Ekeland (1984b)**, Applied Nonlinear Analysis, John Wiley & Sons, Inc.

A. **Auslender** , J. P. **Crouzeix** et P. **Fedit (1987)**, "Penalty-Proximal methods in convex programming", Journal of Optimization Theory and Applications, Vol. 55, N° 1, pp 1-21.

J. B. **Baillon (1978)**, "Un exemple concernant le comportement asymptotique de la solution du problème $0 \in du/dt + \partial\varphi(u)$", Journal of Functional Analysis 28, pp 369-376.

J. F. **Benders (1962)**, "Partitionning procedure for solving mixed variables programming problems", Numerische Mathematik, 4, pp 238-252.

D. P. **Bertsekas (1976)**, "On the Goldstein-Levitin-Polyak gradient projection method", IEEE Trans. Aut. control. Vol 21, N° 2, pp 174-184.

D. P. **Bertsekas (1987)**, P. A. Hosein et P. Tseng, "Relaxation methods for network flow problems with convex arc costs", SIAM J. Control and Optimization. Vol 25, N°. 5, pp 1219-1243

D. P. **Bertsekas** et J. N. **Tsitsiklis (1989)**, Parallel and Distributed Computation, Prentice-Hall int. Ed.

P. **Bouvry et al (1991)**, "Manuel du méganode" Rapport technique 63, Avril 1991 Laboratoire de Modélisation et de Calcul (URA397) IMAG.

L.M. **Bregman (1965)**, "The method of successive projection for finding a common point of convex sets". Akad. Nauk. SSSR Dokl., 162, pp 487-490. Traduction anglaise, dans Soviet Math. Dokl., 162, pp 688-692.

H. **Brezis (1973)**, Operateurs Maximaux Monotones et Semi-groupes de Contractions dans les Espaces de Hilbert, Mathematics Studies 5, North Holland

H. **Brezis** et A. **Haraux** **(1976)**, "Image d'une somme d'opérateurs maximaux monotones et applications", Israel Journal of Mathematics, Vol. 23. N° 2, pp 165-186

H. **Brezis** et P. L. **Lions** **(1978a)**, "Produit infini de résolvantes", Israel Journal of Mathematics, Vol. 29. N° 4, pp 329-345

F. E. **Browder** **(1964)**, "Continuity properities of monotone nonlinear operators in Banach spaces", Bull. Amer. Math. Soc., vol. 70, pp 551-553.

F. E. **Browder** **(1965)**, "Multi-valued monotone nonlinear mappings and duality mappings in Banach spaces". Trans. Amer. Math. Soc., vol. 118, pp 338-351.

F. E. **Browder** **(1968)**, "Nonlinear maximal monotone operators in Banach space". Math. Anal., vol. 175, pp 89-113.

F. E. **Browder** et W. V. **Petryshyn** **(1966)**, "The solution by iteration of nonlinear functional equations in Banach spaces". Bull. Amer. Math. Soc., vol. 72, pp 571-575.

F. E. **Browder** et W. V. **Petryshyn** **(1967)**, "Construction of fixed points of nonlinear mappings in Hilbert space", Journal of Mathematical Analysis and applications 20, 197-228.

R. E. **Bruck** **(1975)**, "Asymptotic convergence of nonlinear contraction semigroups in Hilbert space", Journal of Functional Analysis 18, pp 15-26.

R. E. **Bruck** **(1978)**, "On the almost-convergence of iterates of a non expansive mapping in Hilbert space and the structure of the weak $\omega$-limit set", Israel Journal of Mathematics. Vol. 29. N° 1, pp 1-16.

Y. **Censor** **(1981)** et A. **Lents**, "An iterative row-action method for interval convex programming", Journal of Optimization Theory and Applications : vol 34, N° 3, 321-353,

**C.M.1(1989)** Thinking Machines Corporation "Connection Machine Programming in C/Paris Version 5" ;Cambridge, Massachusetts.

**C.M.2** **(1989)** Thinking Machines Corporation, "Connection Machin Parallel Instruction Set", Cambridge, Massachusetts.

G. **Cohen** et D.L. **Zhu(1983)**, "Decomposition coordination methods in large scale optimization problems. The nondifferentiable case and the use of the augmented lagrangians", dans Advences in Large Scale Systems, J.B. Cruz ed. vol. I, JAI Press.

M. C. **Crandall** et A. **Pazy** **(1969)**, "Semi-group of nonlinear contractions and dissipative sets", Journal of Functional Analysis 3, pp 376-418.

G. B. **Dantzig** et P. **Wolfe** **(1961)**, "The decomposition algorithm for linear programming", Economtrica, vol. 29, N° 4, pp 767-778.

E. **Decamp** et B. **Amy** **(1988)** "Neurocalcul et réseaux d'automates. Le point sur les recherches et les applications", EC2.

J.P. **Derycke** et P. **Regache** **(1988)** "Le Transputer", Rapport de projet année spéciale ENSIMAG.

J. **Eckstein** **(1989)**, "Splitting methods for monotone operators with applications to parallel optimization", PhD thesis, MIT.

I. **Ekeland** et R. **Temam** **(1974)**, Analyse Convexe et Problèmes variationnels, Dunod, ed. Gauthier-Villars, Paris.

M. C. **Ferris(1991)**, "Finite termination of the proximal point algorithm", Mathematical Programming, vol. 50, N° 3, pp 359-366.

M.J.**Flynn** **(1972)** "Some computer organisations and their effectiveness", IEEE Transactions on Computers.

M. **Fortin** et R. **Glowinski** **(1982)**, Méthodes de Lagrangien Augmenté, Bordas, Paris 1982.

D. **Gabay** et B. **Mercier** **(1976)**, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation", Comp. and Maths. with Appls. Vol. 2, p 17-40, Pergamon Press.

A. **Ghinzzetti** ed. **(1969)**, Theory and Applications of Monotone Operators, A. Ghinzetti ed., Proc. of a NATO Advanced Study Institute, Venise/Italy, pp 1-33, 1968. Edition ODERISI.

K. **Goebel** et W.A. **Kirk** **(1990)**, Topics in Metric Fixed Point Theory", Cambridge Studies in Advenced Mathematics, 28, Cambridge University Press.

E. G. **Gol'shtein(1986)**, "The block method of convex programming", Soviet Math. Dokl. Vol. 33, pp 584-587.

E. G. **Gol'shtein** et N. V. **Tret'yakov** **(1979)**, "Modified lagrangians in convex programming and their generalizations", Mathematical Programming Study 10, pp 86-97.

**Gong Chen** et M. **Teboulle** **(1992)**, "Convergence analysis of a proximal-like minimization algorithm using Bregman functions", To appear in SIAM J. Optimization.

O. **Güler** **(1991)**, "On the convergence of the proximal point algorithm for convex minimization", SIAM J. Control and Optimization, Vol. 29, N° 2, pp 403-419.

Cu D. **Ha** **(1990)**, "A generalization of the proximal point algorithm", SIAM J. Control and Optimization, Vol. 28, N° 3, pp 503-512.

M. R. **Hestenes** **(1969)**, "Multiplier and gradient methods", J. Optimization Theory and applications", Vol. 4, pp 303-320.

D.W. **Hillis(1985)** "The Connection Machine", Thinking Machine Corporation Press 1985.

H. **Idrissi**, O. **Lefebvre** et C. **Michelot (1989)**, "Duality for constrained multifacility location problems with mixed norms and applications", Annals of Operations Research, 18, pp 71-92.

S. **Kaczmarz (1937)**, "Angegherte auflosung von systemn linearer gleichungen". Bull. International de l'accademie Polonaise Sci. A, 1937, pp 355-357.

J.E. **Kelley** Jr.**(1960)**, "The cutting plane method of convex programming", SIAM, N° 8, pp 703-712.

M.A. **Krasnosel'skii (1955)**, "Two observation about the method of successive approximations". Uspehi Math. Nauk., vol. 10, N° 1, pp 123-127.

L. S. **Lasdon (1970)**, Optimization Theory for Large Systems, Macmillan Series in Operation Research, New York.

P. J. **Laurent (1972)**, Approximation et Optimisation, Hermann, Paris.

J. **Lawrence** et J. E. **Spingarn(1986)**, "On fixed points of non expansive piecewise isometric mappings", Proc. London Math. soc.

O. **Lefebvre** et C. **Michelot (1988)**, "About finite convergence of the proximal point algorithm", International Series of Numerical Mathematics, Vol. 84, pp 154-161, (c) 1988, Birkhäuser Verlag, Basel.

T. **Leighton (1990)**, "Parallel computation", Technical report 227/90 Departement of computer science University of Toronto.

B. **Lemaire (1988)**, "Coupling optimization methods and variational convergence", International Series of Numerical Mathematics, Vol. 84, pp 163-179, (c), Birkhäuser Verlag, Basel.

B. **Lemaire (1990)**, "About the proximal point algorithm". 1er congrès Franco-Soviétique d'Optimisation, Luminy.

C.**Lemarechal(1980)**, "Extensions diverses des méthodes de gradient et applications", Thèse de Doctorat ès Sc., Université de Paris IX.

C.**Lemarechal(1991)**, "Lagrangian decomposition and nonsmooth optimization : Bundle Algorithm, Prox iteration, Augmented lagrangian", dans Proceedings Erice, F. Giannesi, ed.

P. L. **Lions** et G.I. **Marchouk(1974)** (éditeurs), Sur les Méthodes Numériques en Sciences Physiques et Economiques", Dunod, Paris.

P. L. **Lions** et B. **Mercier (1979)**, "Splitting algorithms for the sum of two nonlinear operators", SIAM J. Numer. Anal. Vol. 16, N° 6.

H.P.L. **Luna(1978)**, Les techniques de décomposition-coordination dans les modèles économiques d'optimisation", Thèse de Doctorat de l'Université Paul Sabatier de Toulouse.

F. J. **Luque (1984)**, "Asymptotic convergence analysis of the proximal point algorithm", SIAM J. Control and Optimization, Vol. 22, N° 2, pp 277-293.

P. **Mahey (1990)**, "Méthodes de Décomposition pour la Programmation Mathématique", Thèse d'Habilitation à diriger des recherches, Institut National Polytechnique de Grenoble.

P. **Mahey**, V.H. **Nguyen**, S. **Oualibouch** et D.T. **Pham (1992a)**, " Proximal techniques for convex programming". Rapport de recherche du laboratoire ARTEMIS/ IMAG. (RR 877-M).

P. **Mahey** et D.T. **Pham (1992b)**, "Partial regularization of the sum of two maximal monotone operators". $M_2AN$, Vol. 27.

P. **Mahey**, S. **Oualibouch** et D.T. **Pham (1992c)**, "Proximal decomposition on the graph of a maximal monotone operator". a paraitre dans SIAM J. Optim.

B. **Martinet (1970)**, "Régularisation d'inéquations variationnelles par approximations successives", Revue Française d'Informatique et de Recherche Opérationnelle, vol. 4 (R-3), pp 154-158.

B. **Martinet (1972)**, "Determination approchée d'un point fixe d'une application pseudo-contractante. Cas de l'application prox". Compte rendu de l'Academie des Sciences, Paris, série A274, pp 163-165.

B. **Martinet (1972)**, "Algorithmes pour la résolution de problèmes d'optimisation et de minimax", Thèse d'état, Univ. de Grenoble.

D. **Medhi(1987)**, "Decomposition of structured large scale optimization problems and parallel optimization", Ph. D. Thesis, University of Wisconsin - Madison.

M. **Minoux (1983)**, "Programmation Mathématique, Théorie et Algorithmes", Tomes 1 et 2, Dunod ed., Paris.

G. J. **Minty (1961)**, "On the maximal domain of a monotone function", Michigan Mathematical Journal, 8, pp 135-137.

G. J. **Minty (1962)**, "Monotone (nonlinear) operators in Hilbert space", Duke Mathematics Journal, 29, pp 341-346.

G. J. **Minty (1964)**, "On the monotonicity of the gradient of a convex function", Pacific Journal of Mathematics, 14, pp 243-247.

J.J.**Moreau (1965)**, "Proximité et dualité dans un espace de Hilbert", Bulletin de la

societé Mathématique de France, 93, pp 273-299.

K. **Mouallif**, V. H. **Nguyen** et J. J. **Strodiot (1991)**, "A perturbed parallel decomposition method for a class of nonsmooth convex minimization problems", SIAM J. Control and Optim, Vol. 29, pp 829-847.

Z.**Opial(1967)**, "Weak convergence of the sequence of successive approximations for non expansive mappings". Bull. Amer. Math. Soc., vol. 73, pp 591-597.

G.B.**Passty (1979)**, "Ergodic convergence to a zero of the sum of two monotone operators in Hilbert space", Journal of Mathematical Analysis and Applications, 72, pp 383-390.

G.**Pierra (1984)**, "Decomposition through formalization in a product space", Mathematical Programming 28, pp 96-115.

B.T.**Polyak** et E.S. **Levitin (1966)**, "Constrained minimization methods". USSR Comp. Math. and Math. Phys. vol. 6, N° 5, pp 1-50.

M.J.D.**Powell (1969)**, "A method for nonlinear constraints in minimization problems", dans Optimization, R. Fletcher ed., New York, Academic Press.

Di S.**Reich (1977)**, "On infinite product of resolvants", Lincei - Rend. Sc. Fis. Mat. e Nat. - Vol. LXIII - pp 338-340.

R.T.**Rockafellar (1966)**, "Characterization of the subdifferentials of convex functions". Pacific Journal of Math., vol. 17, N° 3, pp 497-510.

R.T.**Rockafellar (1969)**, "Convex functions, monotone operators and variational inequalities", Proceeding of a NATO advanced study institute, Venise/Italy, pp 36-65, 17-30 Juin 1968. Edition ODERISI, Gubbio.

R.T.**Rockafellar (1970a)**, "Monotone operators associated with saddle-function and minimax problems", Symp. in Pure Mathematics, Vol. 18, part 1, AMS, F. Browder, ed., pp 241-250.

R.T.**Rockafellar (1970b)**, "On the maximality of the sum of nonlinear operators", Trans. of the American Math. Soc., Vol. 149, pp 75-88.

R.T.**Rockafellar (1970c)**, "Convex Analysis", Princeton University Press.

R.T.**Rockafellar (1974)**, "Augmented lagrange multiplier function and duality in nonconvex programming", SIAM J. Control, Vol. 12, N° 2, pp 268-285.

R.T.**Rockafellar (1976a)**," Monotone operators and the proximal point algorithm", SIAM J. Control, Vol. 14, N° 5, pp 877-898.

R.T.**Rockafellar (1976b)**, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming", Mathematics of Operations Research, Vol. 1, N° 2, pp 97-116.

H.**Schaefer (1957)**, "Über die Methode sukzessiver Approximationen". Jber. Deutsch. Math. -Verein. Vol. 59, pp 131-140.

N.Z.**Shor (1968)**, "On the rate of convergence of the generalized gradient method", Kibernetika, vol. 4, N° 3.

J.E.**Spingarn (1982)**, "Submonotone mappings and the proximal point algorithm", Numer. Funct. Anal. and Optimiz., Vol. 4, N° 2, pp 123-150.

J.E.**Spingarn (1983)**, "Partial inverse of a monotone operator", Appl. Math. Optim., Vol. 10, pp 247-265.

J.E.**Spingarn (1985)**, "Applications of the method of the partial inverse to convex programming : Decomposition", Math. Progr., Vol. 32, pp 199-223.

M.**Teboulle(1992)**, "Entropic proximal mappings with applications to nonlinear programming", Math. Oper. Res. Vol. 17, N° 3, pp 670-690.

A. **Tikhonov** et V. **Arsenine (1976)**, "Méthodes de résolution de problèmes mal posés". Edition MIR de Moscou, traduction française.

A. **Titli(1975)**, Commande Hiérarchisée et Optimisation des Processus Complexes", Dunod, Paris.

P. **Tseng (1992)**, "On the convergence of the products of fermement non expansif mappings". SIAM J. Optim., vol. 2, N° 3, pp 425-434.

P. **Tseng** et D. P. **Bertsekas (1992)**, "On the convergence of the exponential multiplier method for convex programming", à parître dans Mathematical Programming.

P. **Tseng** et D. P. **Bertsekas (1992)**, "Partial proximal minimization algorithm for convex programming", 4$^{th}$ SIAM Conf. on Optimization.

P. **Tossings (1990)**, "Sur les zéros des opérateurs maximaux monotones et applications", Thèse de Doctorat, Université de Liège, 1990.

D. **Trystram (1988)**, "Quelques résultats de complexité en algorithmique parallèle et systolique",Thèse de doctorat de l'INPG, TIM3.

D. **Trystram (1990)**, "Supercalculateurs", LMC-IMAG

J.**Von Neumann (1950)**, "Functional operators, Vol. II : The geometry of orthogonal spaces". Annals of Math. Stud. N° 22, Princeton Univ. press, Princeton.

K.P. **Wismer(1970)** (éditeur), Optimization Methods for Large scale Systems, McGraw Hill, New York.

K. **Yosida (1968)**, "Functional Analysis", Springer-Verlag New York Inc.

S.A.**Zenios (1991a)**, "Massively parallel row-action algorithms for some nonlinear transportation problems", SIAM J. Optimization, Vol. 1, N° 3, pp 373-400.

S.A.**Zenios (1991b)**, "On the fine-grain decomposition of multicommodity transportation problems", SIAM J. Optimization, Vol. 1, N° 4, pp 643-669, 1991.

D.L.**Zhu (1982)**, "Optimisation sous-différentiable et Méthodes de Décomposition", Thèse de doctorat de l'Ecole Nationale Supérieure des Mines de Paris, Paris 1982.