M2AN - MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

D. DELESALLE
L. DESBAT
D. TRYSTRAM

Résolution de grands systèmes linéaires creux par méthodes itératives parallèles

M2AN - Modélisation mathématique et analyse numérique, tome 27, n° 6 (1993), p. 651-671

<http://www.numdam.org/item?id=M2AN_1993__27_6_651_0>

MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE
(Vol. 27, n° 6, 1993, p. 651 à 671)

RÉSOLUTION DE GRANDS SYSTÈMES LINÉAIRES CREUX PAR MÉTHODES ITÉRATIVES PARALLÈLES (*)

par D. DELESALLE (1), L. DESBAT (2) et D. TRYSTRAM (1)

Communiqué par R. TEMAM

Résumé. — *La méthode du Gradient Conjugué est couramment utilisée pour la résolution de grands systèmes linéaires creux. Sa parallélisation met en évidence le coût prohibitif des opérations de réduction pour le calcul des paramètres optimaux à chaque itération. Dans cet article, nous proposons une analyse de la méthode dans laquelle on remplace les paramètres du Gradient Conjugué par des constantes et rappelons le lien avec la méthode de Richardson du second ordre. Nous comparons les deux méthodes par des expérimentations séquentielles. Puis, nous étudions leur parallélisation et proposons une implémentation sur Connection Machine.*

Abstract. — Solution of large linear sparse systems by parallel iterative methods. *The Conjugate Gradient method is widely used for solving large sparse linear systems. Its parallelization shoes the prohibitive cost of the reduction operation (which consists of computing the optimal parameters at each iterations). In this paper, we propose an analysis of the algorithm derived from the Conjugate Gradient method, with constant parameters. We remind the link with the second order Richardson method and compare it with the Conjugate Gradient method. Then, we study their parallelizations, and propose an implementation on the Connection Machine.*

1. INTRODUCTION

1.1. Présentation générale

Nous présentons dans cet article une parallélisation de la méthode du Gradient Conjugué pour résoudre des grands systèmes linéaires creux tels que ceux qui proviennent de la discrétisation par éléments finis. La parallélisation des méthodes de résolution de ces systèmes a suscité récemment beaucoup de recherches [1], [14], [16], [21], [24]. Dans la

(*) Manuscrit reçu le 10 octobre 1991 et sous forme révisée le 22 octobre 1992.

(1) LMC-IMAG, algorithme parallèle et calcul formel, 46 Av. Félix Viallet, 38031 Grenoble Cedex.

(2) TIMB, UJF-CNRS-CHUG, faculté de médecine, 38700 La Tronche.

M² AN Modélisation mathématique et Analyse numérique 0764-583X/93/06/$ 4.00

section 2, nous remplaçons les paramètres du Gradient Conjugué nécessitant des calculs de réduction par des constantes. Il est bien connu [11], [15] que la méthode obtenue est une accélération polynomiale de la méthode de Richardson du premier ordre. Nous calculons la borne théorique de l'erreur de cette accélération (le taux de convergence associé aux paramètres constants optimaux correspond à la borne classique donnée pour le Gradient Conjugué). Les liens entre le Gradient Conjugué et les méthodes d'accélération de Tchebycheff sont bien établis [11], [15]. Le couple de paramètres constants optimaux conduit simplement à la méthode de Richardson du second ordre (version asymptotique de l'accélération de Tchebycheff de la méthode de Richardson du premier ordre [11]). Nous montrons par des expérimentations que ses performances sont comparables à celles du Gradient Conjugué. Dans la section 3, nous analysons les deux algorithmes et proposons une parallélisation efficace pour les matrices provenant de schémas réguliers. Nous présentons dans la section 4, une implantation sur Connection Machine et comparons les deux méthodes en fonction de la précision.

1.2. Description de la méthode du Gradient Conjugué

La méthode du Gradient Conjugué introduite dans [17], est une méthode très largement utilisée pour la résolution de systèmes linéaires. Elle est particulièrement bien adaptée au cas des larges systèmes creux [1]. Ses propriétés fondamentales ont été étudiées dans [11] et ont fait l'objet d'une très large recherche [13], [20].

Considérons le problème de la résolution du système linéaire suivant :

$$Ax = b$$

où A est SDP (Symétrique Définie Positive) de dimension n. Notons d_i ses valeurs propres ordonnées comme suit : $0 < d_1 \leqslant d_2 \leqslant \cdots \leqslant d_n$. $\kappa(A) = d_n/d_1$ désigne le conditionnement du système. Rappelons l'itération du Gradient Conjugué associé à la matrice A. A partir des vecteurs initiaux : x_0 donné et $p_0 = -r_0 \overset{\text{def}}{=} b - Ax_0$:

$$\begin{aligned} \lambda_k &= \frac{\|r_k\|^2}{\langle Ap_k, p_k \rangle} \\ x_{k+1} &= x_k + \lambda_k p_k \\ r_{k+1} &= r_k + \lambda_k Ap_k \\ \beta_{k+1} &= \frac{\|r_{k+1}\|^2}{\|r_k\|^2} \\ p_{k+1} &= -r_{k+1} + \beta_{k+1} p_k \end{aligned} \tag{1}$$

pour $k = 1, 2, \ldots$ jusqu'à convergence.

Dans cet algorithme, deux paramètres sont calculés à partir de produits scalaires : λ_k le paramètre optimal de descente le long de la direction p_k et β_k le paramètre optimal de choix d'une direction de descente dans le plan défini par le gradient et la direction de descente précédente. L'optimalité de ces deux paramètres induit la propriété fondamentale du Gradient Conjugué : les directions de descentes successives sont conjuguées 2 à 2 [11], [13]. Cette propriété garantit une convergence de la méthode en au plus n itérations.

2. PARAMÈTRES CONSTANTS

L'idée que nous développons dans la suite repose sur la remarque suivante : dans l'algorithme du gradient à pas optimal (méthode dite de la plus profonde descente), si nous remplaçons le pas de descente localement optimal à chaque étape par un pas constant optimal, on obtient une méthode itérative dont la vitesse de convergence est la meilleure borne que l'on sache donner au gradient à pas optimal par l'inégalité de Kantorovitch [20]. Les liens entre la méthode du gradient conjugué préconditionné et les méthodes d'accélération de Tchebycheff [3], [11], [15], indiquent que le même phénomène que celui rencontré dans le gradient à pas optimal se produit ici encore : la meilleure borne donnée par les polynômes de Tchebycheff est quasiment atteinte lorsqu'on remplace les paramètres λ_k et β_k par des constantes optimales. Nous rappelons ce résultat dans le paragraphe suivant.

2.1. λ et β constants

De simples manipulations algébriques à partir d'une itération du Gradient Conjugué donnent :

$$r_{k+1} = ((1 + \lambda_k/\beta_k/\lambda_{k-1})I - \lambda_k A)\, r_k - \lambda_k \beta_k/\lambda_{k-1}\, r_{k-1}\,.$$

Où I est la matrice identité d'ordre n.

Si nous remplaçons respectivement dans les formules de l'algorithme (1) les paramètres λ_k et β_k par des constantes λ et β, on obtient pour l'équation précédente la relation récurrente d'ordre 2 :

$$r_{k+1} = ((1 + \beta)I - \lambda A)\, r_k - \beta r_{k-1}\,. \tag{2}$$

Soit $A = U^t DU$ où D diag (d_i) est diagonale et U unitaire, la suite y_k convergera vers le vecteur nul si les suites

$$y_{k+1}(i) = ((1 + \beta) - \lambda d_i)\, y_k(i) - \beta y_{k-1}(i) \tag{3}$$

convergent vers 0, $\forall i = 1 \ldots n$, ou de manière équivalente si le rayon spectral de la matrice de l'itération suivante est inférieur à 1 :

$$\begin{bmatrix} y_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} (1 + \beta)I - \lambda D & -\beta \\ I & 0 \end{bmatrix} \begin{bmatrix} y_k \\ y_{k-1} \end{bmatrix}. \tag{4}$$

La proposition suivante donne la surface de convergence en β et λ de l'itération (1) à paramètres constants.

PROPOSITION 1 : *L'itération* (2) *converge pour les couples de paramètres constants* (λ, β) *tels que :*

$$-1 < \beta < 1$$
$$0 < \lambda < \frac{2(1+\beta)}{d_n}.$$

Preuve : Nous ne présentons que les grandes lignes de la démonstration, les détails se trouvent dans [5].

Soit $\Delta_i = (1 + \beta - \lambda d_i)^2 - 4\beta$ le discriminant de l'équation caractéristique $E_i : f^2 - (1 + \beta - \lambda d_i) f + \beta = 0$ associée à (3) :

• Lorsque $\beta < 0$, on peut remarquer que $\forall i = 1 \ldots n$, $\Delta_i > 0$, donc E_i a deux solutions. Notons $f_i(\lambda, \beta)$ le maximum du module des solutions de E_i, alors

$$f_i(\lambda, \beta) = \begin{cases} f_1^i(\lambda, \beta) \overset{\text{def}}{=} (1 + \beta - \lambda d_i + \sqrt{\Delta_i})/2 \text{ si } 1 + \beta - \lambda d_i \geqslant 0 \\ f_2^i(\lambda, \beta) \overset{\text{def}}{=} (\lambda d_i - 1 - \beta + \sqrt{\Delta_i})/2 \text{ si } 1 + \beta - \lambda d_i \leqslant 0 \end{cases} \tag{5}$$

$f_i(\lambda, \beta) = \max(f_1^i, f_2^i) = 1/2(|1 + \beta - \lambda d_i| + \sqrt{\Delta_i})$. Il est facile de montrer que lorsque $-1 < \beta < 0$, la vitesse de convergence est plus mauvaise que celle du gradient à pas constant optimal (*cf.* [20]).

• Lorsque $\beta \geqslant 0$ alors :

$$\Delta_i > 0 \Leftrightarrow |1 + \beta - \lambda d_i| > 2\sqrt{\beta}.$$

Le maximum des modules des solutions est la fonction :

$$f_i(\lambda, \beta) = \begin{cases} f_1^i(\lambda, \beta) & \text{si } 1 + \beta - \lambda d_i \geqslant 2\sqrt{\beta} \\ f_2^i(\lambda, \beta) & \text{si } 1 + \beta - \lambda d_i \leqslant -2\sqrt{\beta} \\ f_{12}^i(\lambda, \beta) \overset{\text{def}}{=} \sqrt{\beta} & \text{si } 1 + \beta - 2\sqrt{\beta} \leqslant \lambda d_i \leqslant 1 + \beta + 2\sqrt{\beta}. \end{cases} \tag{6}$$

Pour que l'itération vectorielle (2) converge, il faut et il suffit que

$$f_{\max}(\lambda, \beta) \overset{\text{def}}{=} \max_{i = 1 \ldots n} f_i(\lambda, \beta) < 1,$$

avec f_i donné en (5) et (6). $f_{\max}(\lambda, \beta)$ est le rayon spectral de la matrice de l'itération (4). L'étude de $f_{\max}(\lambda, \beta)$ repose sur l'ordonnancement des courbes f_1^i et f_2^i. On peut montrer que :

- $f_1^i(\lambda, \beta)$ est décroissante en λ et en i sur son intervalle de définition.

$$\lambda' \leqslant \lambda \Rightarrow f_1^i(\lambda, \beta) \leqslant f_1^i(\lambda', \beta) \text{ et } j \leqslant i \Rightarrow f_1^i(\lambda, \beta) \leqslant f_1^j(\lambda, \beta).$$

- $f_2^i(\lambda, \beta)$ est croissante en λ et en i sur son intervalle de définition.
- Pour $\beta > 0$, $f_1^i(\lambda, \beta) \geqslant \sqrt{\beta}$ et $f_2^i(\lambda, \beta) \geqslant \sqrt{\beta}$ sur leur intervalle de définition.
- Lorsque $\beta < 0$ fixé, les courbes $f_1^1(\lambda)$ et $f_2^n(\lambda)$ se croisent lorsque :

$$1 + \beta - \lambda_{\text{opt}}(\beta)\, d_1 + \sqrt{\Delta_1} = -1 - \beta + \lambda_{\text{opt}}(\beta)\, d_n + \sqrt{\Delta_n} \Rightarrow$$

$$\Rightarrow \lambda_{\text{opt}}(\beta) = \frac{2(1+\beta)}{d_1 + d_n}.$$

- Lorsque $\beta > 0$ fixé, elles se croisent en $\lambda_{\text{opt}}(\beta)$ si

$$\frac{1 + \beta + 2\sqrt{\beta}}{d_n} \leqslant \frac{1 + \beta - 2\sqrt{\beta}}{d_1} \Leftrightarrow (1 + \sqrt{\beta})^2 \leqslant \kappa(A)(1 - \sqrt{\beta})^2 \Leftrightarrow$$

$$\Leftrightarrow \sqrt{\beta} \leqslant \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}.$$

Nous pouvons conclure des propriétés précédentes que pour β fixé :

- soit $\beta \leqslant \left(\dfrac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^2$

$$f_{\max}(\lambda) = \begin{cases} f_1^1(\lambda) & \text{si} \quad 0 < \lambda \leqslant \lambda_{\text{opt}}(\beta) \\ f_2^n(\lambda) & \text{si} \quad \lambda_{\text{opt}}(\beta) \leqslant \lambda \end{cases} \tag{7}$$

- soit $\sqrt{\beta} > \dfrac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}$

$$f_{\max}(\lambda) = \begin{cases} f_1^1(\lambda) & \text{si} \quad 0 < \lambda \leqslant \dfrac{1 + \beta - 2\sqrt{\beta}}{d_1} \\ \sqrt{\beta} & \text{si} \quad \dfrac{1 + \beta - 2\sqrt{\beta}}{d_1} \leqslant \lambda \leqslant \dfrac{1 + \beta + 2\sqrt{\beta}}{d_n} \\ f_2^n(\lambda) & \text{si} \quad \dfrac{1 + \beta + 2\sqrt{\beta}}{d_n} \geqslant \lambda. \end{cases} \tag{8}$$

Il est alors facile de montrer que

$$|\beta| \geqslant 1 \Rightarrow f_{\max}(\lambda) \geqslant 1,$$

et que

$$\text{si} \quad |\beta| < 1 \quad \text{alors} \quad f_{\max}(\lambda) < 1 \Leftrightarrow 0 < \lambda < \frac{2(1+\beta)}{d_n}$$

□

PROPOSITION 2 *Le couple de parametres constants*

$$\beta_{\text{opt}} = \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^2$$

$$\lambda_{\text{opt}} = \frac{2(1+\beta_{\text{opt}})}{d_1 + d_n}$$

donne une vitesse de convergence optimale

$$\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}$$

Preuve D'après ce qui précède, pour $-1 < \beta \leqslant \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^2$ fixé, la vitesse de convergence optimale est réalisée au point $\lambda_{\text{opt}}(\beta)$ et est donnée par

$$f_1^1(\lambda_{\text{opt}}(\beta), \beta) =$$

$$= 1/2 \left(1 + \beta - \frac{2(1+\beta)}{d_1 + d_n} d_1 + \sqrt{\left(1 + \beta - \frac{2(1+\beta)}{d_1 + d_n} d_1 \right)^2 - 4\beta} \right)$$

Cette fonction est décroissante en β et atteint son minimum en la borne $\beta_{\text{opt}} = \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^2$

$$f_1^1(\lambda_{\text{opt}}(\beta_{\text{opt}}), \beta_{\text{opt}}) = \sqrt{\beta_{\text{opt}}} = \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}$$

Lorsque $\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^2 < \beta < 1$ la vitesse optimale de convergence est réalisée sur l'intervalle

$$\frac{1 + \beta - 2\sqrt{\beta}}{d_1} \leqslant \lambda \leqslant \frac{1 + \beta + 2\sqrt{\beta}}{d_n}$$

et vaut $\sqrt{\beta}$ La vitesse de convergence devient donc croissante en fonction de β □

On peut illustrer les deux propositions qui précèdent par une représentation de $f_{\max}$ à travers les courbes f_1^l, f_2^l, f_{12}^l pour differents β Nous présentons le

cas d'une matrice de spectre Sp $(A) = \{2, 3, 4\}$, $(n = 3)$. On peut remarquer que lorsque β est relativement petit, nous sommes proches de la situation du gradient à pas constant. On voit dans la figure 1, que lorsque $\beta < 0$, les courbes f_1^i, f_2^i sont au-dessus des courbes $|1 - \lambda d_i|$. Lorsque β augmente, $f_1^i(\lambda_{opt}(\beta), \beta)$ diminue : la figure 2 montre que $f_{max}(\lambda_{opt}(0.02), 0.02)$ est plus petit que $f_{max}(\lambda_{opt}(-0.01), -0.01)$. La figure 3 montre la situation pour β_{opt}. Lorsque $\beta > \beta_{opt}$, $f_{max}(\lambda_{opt}(\beta), \beta) = \sqrt{\beta}$ croît. Dans la figure 4, $\beta = 0.1 > \beta_{opt}$, nous constatons que $f_{max}(\lambda_{opt}(0.1), 0.1) > f_{max}(\lambda_{opt}(\beta_{opt}), \beta_{opt})$, ceci $\forall \lambda_{opt}(0.1) \in [(1.1 - 2\sqrt{0.1})/2, (1.1 + 2\sqrt{0.1})/4]$.

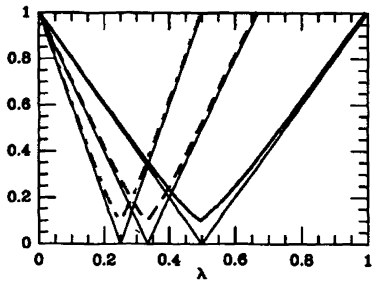

Figure 1.

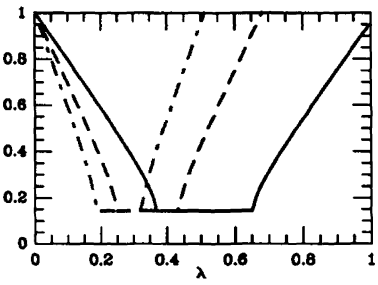

Figure 2.

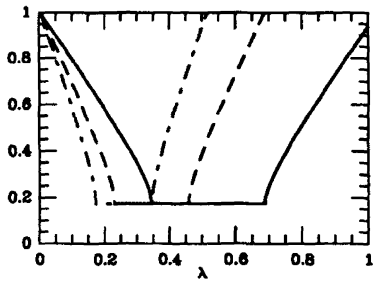

Figure 3.

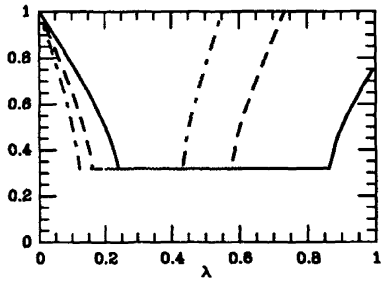

Figure 4.

2.2. Lien avec la méthode de Richardson

On peut identifier cette méthode avec celle de Richardson du deuxième ordre [11]. Elle dérive de l'accélération de Tchebycheff de la méthode du gradient à pas constant.

$$x_{k+1} = \omega_{k+1}(-\alpha r_k + x_k - x_{k+1}) + x_{k+1} . \tag{9}$$

Avec

$$\omega_1 = 1\,, \quad \omega_2 = \frac{2}{2 - \rho^2}\,, \quad \omega_{k+1} = \frac{1}{1 - \omega_k \rho^2/4}$$

où ρ est le rayon spectral de la matrice d'itération de la méthode accélérée : ici $\rho(I - \alpha A) = \dfrac{\kappa(A) - 1}{\kappa(A) + 1}$ pour le choix optimal $\alpha = \dfrac{2}{d_1 + d_n}$. On remplace les paramètres ω_k par le paramètre constant optimal :

$$\begin{aligned}\omega &= \frac{2}{1 + \sqrt{1 - \rho^2}} \\ &= \frac{2}{1 + \sqrt{\dfrac{(\kappa(A) + 1)^2 - (\kappa(A) - 1)^2}{(\kappa(A) + 1)^2}}} \\ &= \frac{2(\kappa(A) + 1)}{(\sqrt{\kappa(A)} + 1)^2}\,.\end{aligned}$$

En multipliant l'équation (9) par A et en retranchant b, on obtient :

$$r_{k+1} = (\omega - \omega\alpha A)\, r_k + (1 - \omega)\, r_{k-1}\,.$$

Posons donc $\beta = \omega - 1$ et $\lambda = \omega\alpha$, on obtient

$$x_{k+1} = (1 + \beta - \lambda A)\, r_k - \beta r_{k-1}\,.$$

On vérifie alors que

$$\beta = \omega - 1 = \frac{2(\kappa(A) + 1) - (\sqrt{\kappa(A)} + 1)^2}{(\sqrt{\kappa(A)} + 1)^2} = \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^2$$

$$\lambda = (1 + \beta)\,\alpha = \frac{2(1 + \beta)}{d_1 + d_n}\,.$$

Remarque 1 : Nous savons (*cf.* [20]) que la méthode du gradient à pas optimal a pour taux de convergence au moins $\dfrac{\kappa(A) - 1}{\kappa(A) + 1}$, plus précisément que :

$$\|x_k^g - x^*\|_A < \left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^k \|x_0^g - x^*\|_A \tag{10}$$

avec $\|x\|_A = x^t Ax$, x_k^g le k-ième itéré de la méthode du gradient à pas optimal, x^* la solution du système linéaire à résoudre. On sait choisir λ_k constant optimal ($\lambda = 2/(d_1 + d_n)$) dans la méthode du gradient à pas constant pour avoir le taux de convergence $\dfrac{\kappa(A) - 1}{\kappa(A) + 1}$. On obtient la méthode

de Richardson du premier ordre conduisant à la même majoration (10) dans laquelle on remplace x_k^q par x_k^{co}, le k-ième itéré de la méthode du gradient à pas constant optimal. Dans le cas du gradient conjugué, l'optimalité sur les espaces de Krylov successifs conduit après quelques majorations au taux de convergence d'au moins $\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}$ et plus précisément à la majoration :

$$\left\| x_k^{gc} - x^* \right\|_A < 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \left\| x_0^{gc} - x^* \right\|_A \tag{11}$$

où x_k^{gc} est le k-ième itéré de la méthode du gradient conjugué. On vient de voir que l'on sait choisir les paramètres λ_k et β_k constants optimaux dans un algorithme semblable au gradient conjugué pour que le taux de convergence soit $\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}$.

La méthode de Richardson du second ordre est donc au Gradient Conjugué ce que la méthode de Richardson du premier ordre est à la plus profonde descente.

Puisque la méthode de Richardson du second ordre construit une itération dans les espaces de Krylov, le *GC* est plus performant en séquentiel au sens de la norme $\| \cdot \|_A$. Par contre, l'introduction de paramètres constants supprime les produits scalaires et donc l'itération est intrinsèquement plus parallèle.

2.3. Expérimentations numériques

Nous avons comparé la méthode du Gradient Conjugué et la méthode de Richardson du second ordre (paramètres constants), dans le cas où A est la discrétisation classique du Laplacien bi-dimensionnel sur une grille carrée de différentes tailles ($n \times n$ avec $n = 60, 100, 200, 300$). Pour les expérimentations, le second membre $\mathbf{b}$ est tel que la solution de $\mathbf{Ax} = \mathbf{b}$ soit le vecteur dont toutes les composantes sont égales à 1. Nous représentons dans les figures 6, 7, 8 et 9, le nombre d'itérations en fonction de la précision relative sur le résidu r. Les courbes en pointillets correspondent au Gradient Conjugué, celles en trait plein à la méthode avec paramètres constants et optimaux. Les résultats représentent le nombre d'itérations et non pas le temps séquentiel car l'objectif est l'implémentation sur une architecture parallèle distribuée (une itération de la méthode à paramètres constants coûte en séquentiel un peu moins cher qu'une itération de gradient conjugué et en parallèle théoriquement beaucoup moins cher).

Le comportement superlinéaire caractéristique du Gradient Conjugué apparaît clairement. La méthode usuelle avec β_{opt} et λ_{opt} reste linéaire. Il faut pourtant souligner que cette méthode atteint la précision de la machine avec

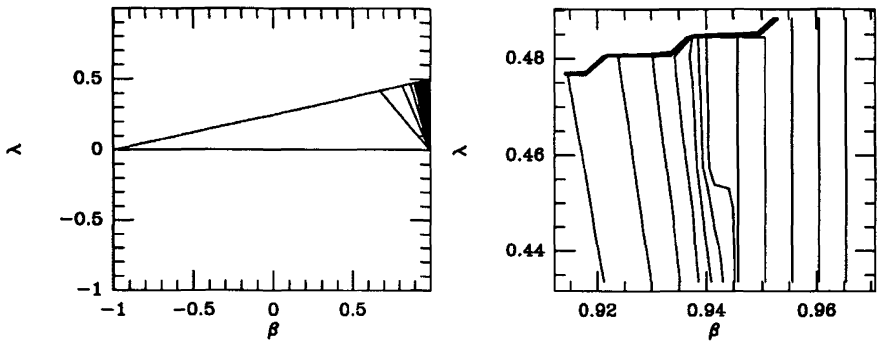

Figure 5. — $f_{max}(\beta, \lambda)$ dans le cas du Laplacien $n = 100$. A gauche nous représentons les courbes de niveau de $f_{max}(\beta, \lambda)$ de 0,97 (légèrement supérieur au minimum) à 1 par pas de 0,0025. A droite nous proposons un agrandissement autour du minimum. On peut remarquer qu'une sous-estimation de β et une surestimation λ peuvent être dramatiques (le contraire l'étant moins).

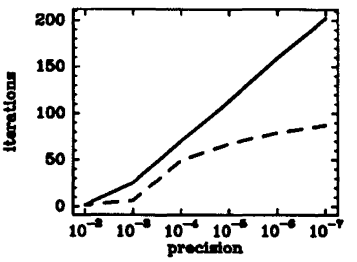

Figure 6. — *GC* et méthode à paramètres constants pour $n \times n = 3\,600$.

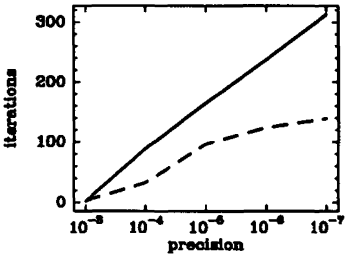

Figure 7. — *GC* et méthode à paramètres constants pour $n \times n = 10\,000$.

environ trois fois plus d'itérations seulement que le Gradient Conjugué (en double précision sur une DECstation 3100). La méthode à paramètres constants est donc très efficace. On peut probablement espérer remplacer avantageusement le Gradient Conjugué sur une machine parallèle à architec-

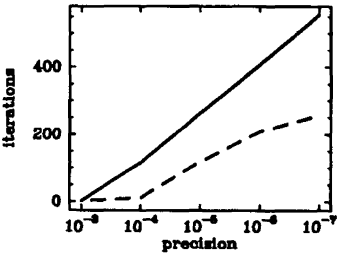

Figure 8. — *GC* et méthode à paramètres constants pour $n \times n = 40\,000$.

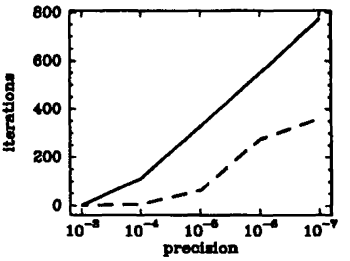

Figure 9. — *GC* et méthode à paramètres constants pour $n \times n = 90\,000$.

ture distribuée, car elle ne nécessite que des communications locales. En particulier si la précision désirée n'est pas très grande, au aura intérêt à l'utiliser puisque les deux méthodes ont un comportement presque similaires (au moins dans cet exemple où le spectre est uniformément réparti).

Le problème de l'estimation des valeurs propres extrêmes peut être résolu par différentes méthodes. Une première approche consiste à mettre en œuvre des méthodes hybrides analogues à celles présentées dans cet article. En général on commence par un petit nombre d'itérations du Gradient Conjugué, puis on poursuit par des itérations de type Tchebycheff. D'autre part, des méthodes concurrentes du type polynômes de moindres carrés donnent de meilleurs résultats pour la localisation des valeurs propres [19]. Une majoration de l'intervalle des valeurs propres de type Gershgorin est alors suffisante. Dans le cas du Laplacien, l'intervalle [0, 8] peut-être utilisé. On peut remarquer dans la figure 5, qu'une surestimation de β et une sous-estimation de λ est meilleure que le contraire.

3. ANALYSE PARALLÈLE DE LA MÉTHODE À PARAMÈTRES CONSTANTS

3.1. Décomposition de l'algorithme du Gradient Conjugué

Le projet LINPACK (LINear algebra PACKage) [6] créé il y a une vingtaine d'années, a pour objectif de fournir à la fois des outils de mesures des programmes scientifiques et des outils généraux de production de gros

logiciels. Il a conduit à la création d'une bibliothèque de procédures en algèbre linéaire. On distingue 3 niveaux d'opérations fondamentales (noyaux des méthodes numériques) notées BLAS1, BLAS2 et BLAS3 en fonction de l'ordre du nombre d'opérations effectuées (n, n^2, n^3) [7]. Cette identification permet une implantation efficace et simple à mettre en œuvre grâce à sa conception modulaire. Cette bibliothèque comprend tous les éléments de base de l'algèbre linéaire : le Dot, le AXPY (BLAS1), le produit matrice-vecteur ou la modification de rang 1 (BLAS2), l'élimination de Gauss ou encore le produit matrice matrice (BLAS3). Le Dot et le AXPY sont les procédures de base de l'algèbre linéaire. Leurs définitions sont les suivantes :

Dot
$s \leftarrow 0$
pour $i \leftarrow 1$ jusqu'à n
$s \leftarrow s + x(i) * y(i)$

AXPY
pour $i \leftarrow 1$ jusqu'à n
$s(i) \leftarrow a * x(i) + y(i)$.

Ces deux opérations requièrent le même nombre d'opérations en séquentiel. On peut décrire l'algorithme du Gradient Conjugué en utilisant la notion de BLAS et ainsi le décomposer en procédures qui se parallélisent facilement [8]. L'itération de base de l'algorithme du Gradient Conjugué s'écrit alors :

1 Produit Matrice-vecteur : Calcul de $A * p_k$
1 Dot suivi d'un calcul scalaire
2 AXPY pour la mise à jour de x_k et r_k
1 Dot suivi d'un calcul scalaire
1 AXPY pour la mise à jour de p_{k+1}.

Excepté le produit matrice-vecteur, toutes les procédures utilisées sont de niveau BLAS1. Ce produit qui appartient au niveau BLAS2 peut être lui-même décomposé en n étapes de niveau 1 (cela dépend du stockage employé).

3.2. État de l'art

Le Gradient Conjugué est très utile pour résoudre de grands systèmes linéaires. Il permet d'utiliser au mieux la structure creuse des matrices contrairement aux méthodes directes comme l'élimination de Gauss qui ne permettent pas en général de respecter le stockage initial. Plusieurs solutions ont été proposées pour paralléliser le Gradient Conjugué. La plupart de ces travaux portent sur un découpage macroscopique en blocs sur des machines à gros grain. Soit par exemple en utilisant une décomposition locale en sous-domaines [21], [16], [14], soit avec une approche plus centralisée qui

correspond simplement à la parallélisation du produit matrice-vecteur [2], [10], [26]. Saad a proposé une approche plus fine [24]. Il a mis en évidence que la chute de l'efficacité d'une itération du Gradient Conjugué était due aux deux points de synchronisation (conséquence des 2 Dot). Des manipulations algébriques sur les formules de base conduisent à une solution où l'on peut remplacer les 2 Dot par trois consécutifs soit un seul point de synchronisation par itération. Cependant, cette méthode n'est pas stable numériquement.

En parallélisme massif distribué où chaque élément de la matrice est placé sur un processeur différent, seule une étude microscopique peut être employée. Si l'on analyse le niveau BLAS1, on s'aperçoit que le Dot nécessite des communications entre les processeurs pour rassembler toute l'information, entraînant ainsi un coût supplémentaire. Par contre, un AXPY ne fait que des opérations locales à un processeur sans communication. Un Dot est donc plus coûteux en parallèle qu'un AXPY en raison des communications supplémentaires.

3.3. Parallélisation de la méthode à paramètres constants

La méthode à paramètres constants évite les deux Dot en supprimant dans l'itération de base les calculs des scalaires λ_k et β_k, et en les remplaçant par des paramètres constants.

Par contre, il subsiste le problème du calcul du produit matrice-vecteur. Dans le cadre du parallélisme massif sur une grille à deux dimensions, le produit matrice-vecteur plein demande l'utilisation de n Dot réalisés en parallèle. De plus, le vecteur résultat n'est pas en général, placé de la même manière que le vecteur initial, il est donc nécessaire de déplacer des éléments pour effectuer les AXPY ce qui entraîne des communications supplémentaires, soit un coût plus important qu'un Dot. Le cas du produit matrice-vecteur plein devient donc inutile à étudier car le coût en communication est trop élevé, et le gain obtenu sur les deux points de synchronisation sera négligeable.

Le cas creux est intéressant à étudier pour trouver un stockage approprié avec la structure à la fois du problème à traiter mais aussi de l'architecture de la machine cible qui obtienne un gain de temps sensible [25].

4. IMPLANTATION

4.1. Description de la Connection Machine

La Connection machine, selon la classification de Flynn [9], [18], est une machine de type SIMD, c'est-à-dire qu'elle exécute une même instruction sur un ensemble de données. Elle est constituée de cartes identiques, regroupant 16 modules de 32-Processeurs Élémentaires 1-bit (PE). Dans sa configuration maximale, elle contient 65 536 PE. Un module contient deux

circuits constitués de 16 PE plus une unité de communication, auxquels s'ajoutent une mémoire locale et une unité de calcul flottant. L'ensemble des modules est relié à l'ordinateur frontal à travers un séquenceur par un bus instructions et un bus d'adresses. Les circuits sont reliés entre eux par un réseau de type hypercube (de degré 12 pour la configuration maximale). Les liaisons internes à un groupe de 16 PE sont réalisées par un réseau d'interconnexion complet. Des détails peuvent être trouvés dans [4], [23].

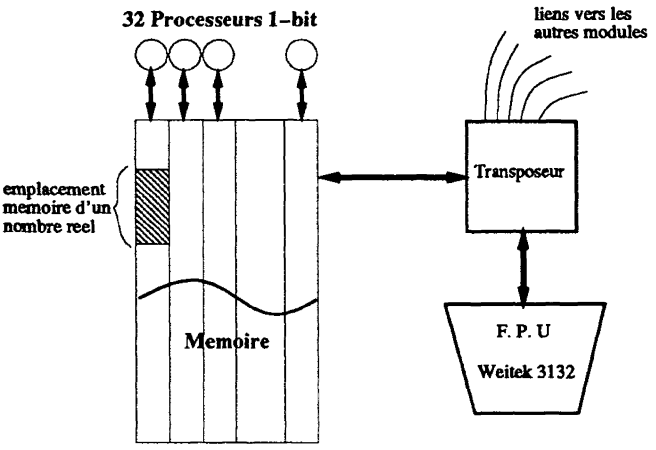

Figure 10. — Schéma d'un module.

Comme l'essentiel des applications testées sur la CM2 sont de type scientifique, une unité flottante a été ajoutée afin d'accélérer les calculs flottants. Ceux-ci sont réalisés au format IEEE simple précision. Sur les dernières versions de la machine, il est possible d'utiliser des processeurs travaillant au format double précision (64 bits) pour améliorer la précision des calculs. Le gain obtenu grâce à l'apport de l'unité flottante est de l'ordre de 20 par rapport au temps série sur les PE.

La programmation de la CM2 peut se faire soit à l'aide de langages de haut niveau tels que le C* ou StarLisp, soit en utilisant le langage ParIS (Parallel Instruction Set) qui est de plus bas niveau mais demande une plus grande attention.

Tous les langages utilisent la notion de processeurs virtuels, c'est-à-dire qu'un processeur physique divise sa mémoire pour simuler de nouveaux processeurs. Cette virtualisation permet ainsi de traiter des problèmes de grande taille sans modifier l'algorithme. On appelle VP-ratio le nombre de processeurs virtuels simulés par un seul processeur physique. Il est à noter que le VP-ratio influe beaucoup sur les performances.

Les communications possibles sont de trois types :

a) Les communications générales qui servent à envoyer un message d'un processeur à n'importe quel autre. Elles utilisent une unité de communication interne à la CM2 ce qui occasionne une perte sensible au niveau des performances.

b) Les communications NEWS qui permettent d'envoyer un message sur un processeur voisin dans la topologie choisie (par exemple nord, est, ouest ou sud pour une grille 2-D). Les processeurs exécutant la même instruction, il n'existe pas de conflit de liens. Ces communications sont donc très rapides, elles coûtent environ le temps de 4 opérations arithmétiques de base.

c) Les communications avec recombinaisons. Elles permettent de communiquer entre les processeurs d'une même dimension de la topologie choisie. Elles associent à la fois communication et opérations de recombinaison binaires des messages. Ces communications utilisent les liens de l'hypercube reliant tous les circuits. Par exemple, elles permettent de réaliser un produit scalaire sur une ligne de matrice. Elles coûtent environ 10 fois plus que les communications NEWS.

4.2. Implantation

Le cas d'une matrice pleine ne permettant pas d'espérer un gain de temps en raison du faible rapport entre une itération de Richardson et une itération du Gradient Conjugué, nous nous sommes uniquement intéressés au cas creux qui représente l'essentiel des utilisations de la méthode du Gradient Conjugué. Nous considérons la résolution du problème du Laplacien dans les mêmes conditions que dans le cas séquentiel.

Il faut donc dans le cas creux créer un nouveau produit matrice-vecteur qui soit plus rapide qu'un produit scalaire. Dans le cas du Laplacien, la matrice est issue du maillage en grille du problème considéré. A chaque pas, un nœud du maillage reçoit une valeur de ces 4 voisins pour calculer sa nouvelle valeur comme le montre la figure suivante.

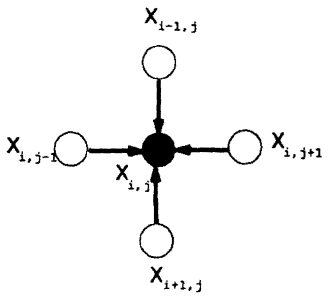

Figure 11. — Calcul au niveau d'un nœud.

C'est ce schéma que nous avons directement reproduit sur la Connection Machine configurée en grille 2-D. Un processeur correspond à un nœud du maillage. Cette méthode a deux avantages importants : un gain de place et un gain de temps.

• Le gain de place se traduit par une absence complète de matrice, il ne reste que le vecteur à stocker. Cela permet donc de résoudre des problèmes de taille très supérieure et donc de pouvoir traiter des problèmes réels.

• Le gain de temps est dû à la structure régulière du maillage du Laplacien. Une itération du produit matrice-vecteur demande un échange de messages entre tous les voisins sur la grille. Ce type de communication correspond au niveau le plus rapide : les communications NEWS sur la Connection Machine. De manière plus générale, on peut appliquer le même raisonnement surtout les schémas « réguliers ». Des exemples d'implantations se trouvent dans [22].

Le seul problème à signaler provient du caractère SIMD de la machine qui empêche de réaliser cette communication en une seule instruction. En effet, ne pouvant effectuer qu'une opération à la fois, il faut la décomposer en quatre communications, une selon chaque point cardinal, soit quatre instructions.

L'algorithme du produit matrice-vecteur creux s'écrit :

$x_{k+1} = 4 * x_k$

si le nœud n'appartient pas au bord Nord
alors reçoit x_k du voisin nord dans x_{trans}

$$x_{k+1} = x_{k+1} - x_{\text{trans}}$$

si le nœud n'appartient pas au bord Sud
alors reçoit x_k du voisin sud dans x_{trans}

$$x_{k+1} = x_{k+1} - x_{\text{trans}}$$

si le nœud n'appartient pas au bord Ouest
alors reçoit x_k du voisin ouest dans x_{trans}

$$x_{k+1} = x_{k+1} - x_{\text{trans}}$$

si le nœud n'appartient pas au bord Est
alors reçoit x_k du voisin est dans x_{trans}

$$x_{k+1} = x_{k+1} - x_{\text{trans}} .$$

Comme nous l'avons décrit précédemment, les communications NEWS sont les plus rapides sur la machine car elles n'entraînent aucun conflit sur les liens. On a donc un gain de temps sensible par rapport au cas plein qui nécessite le calcul de n produits scalaires (communication avec recombinaison). Mais la question importante subsiste : une itération de Richardson coûte-t-elle beaucoup moins chère qu'une itération du Gradient Conjugué et peut-elle compenser l'économie du produit scalaire ?

Avant de repondre a cette question par les experimentations, il faut remarquer qu'il existe une difference pour réaliser un Dot entre le cas d'un stockage plein et le cas creux N'ayant plus de matrice à stocker, le vecteur X est stocké un élément par nœud de la grille Le produit scalaire s'effectue alors en deux étapes un premier produit scalaire sur les lignes suivi d'un deuxième sur les colonnes pour obtenir le résultat final

4.3. Experimentations

La programmation des deux méthodes a été réalisée en deux langages différents StarLisp et ParIS Les expérimentations ont été faites en fonction de la taille du problème à traiter Les résultats obtenus à nombre d'itérations identiques (n = 2 000), sont donnés dans les figures 12 et 13

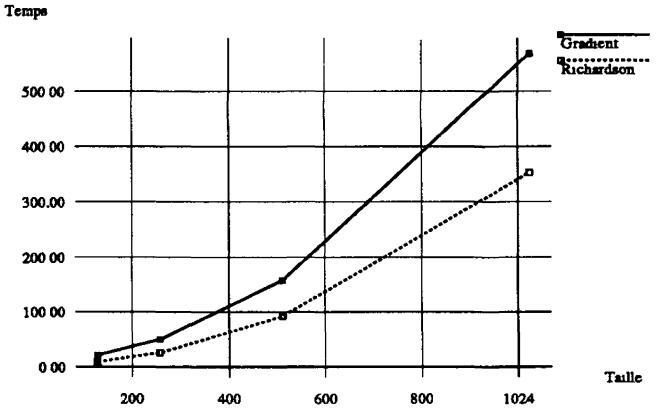

Figure 12. — Comparaison des temps d'execution en StarLisp

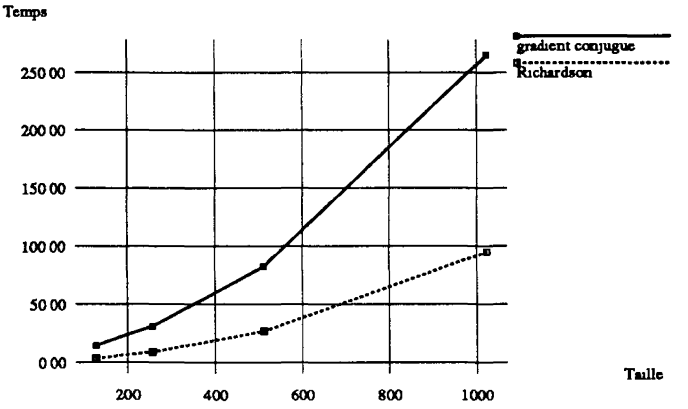

Figure 13 — Comparaison en ParIS

On remarque que dans les deux cas, la méthode à paramètres constants est meilleure que celle du Gradient Conjugué. Le gain est nettement supérieur à celui obtenu dans le cas plein dont les résultats ne sont pas donnés ici. Cela s'explique par le fait que le produit matrice-vecteur creux prend beaucoup moins de temps que dans le cas plein (1 produit scalaire + communications pour remettre le vecteur en place).

On observe également que proportionnellement le rapport entre les deux méthodes sur les deux figures précédentes diminue avec la taille. Ce phénomène est dû au VP-ratio de la CM2, car son influence est d'ordre linéaire sur le produit matrice vecteur tandis qu'elle est d'ordre logarithmique sur les produits scalaires. Une comparaison rapide des deux courbes permet de montrer une accélération sensible des performances entre les deux langages. Il paraît assez clair que le langage StarLisp nécessite des tests supplémentaires vu son caractère de langage de haut niveau. Cette constatation est encore plus flagrante en regardant uniquement la courbe de la méthode de Richardson du second ordre qui est proportionnellement meilleure (facteur 4). Sachant que les deux méthodes n'ont pas le même type de convergence, la méthode à paramètres constants est-elle meilleure que le Gradient Conjugué pour ce type d'implantation sur des schémas réguliers ?

Pour répondre à cette question, nous avons tout d'abord étudié la convergence des deux méthodes pour différentes précisions. Connaissant le résultat exact, il est facile de compter le nombre d'itérations nécessaires pour obtenir une précision donnée en utilisant la norme infinie :

$$\text{precision} = \|x_k - x^*\|_\infty .$$

Les résultats obtenus sont présentés sous la forme du rapport du nombre d'itérations obtenues pour la méthode à paramètres constants sur celui de la méthode du Gradient Conjugué. En raison à la fois de la taille du problème et surtout de la taille du processeur flottant (32-bit), nous avons constaté expérimentalement que la précision maximum ne pouvait dépasser 10^{-5} dans le meilleur des cas. Cependant, il existe des modèles avec des unités flottantes de 64 bits.

On remarque que comme dans le cas séquentiel pour une faible précision, le rapport entre les deux méthodes reste faible, et plus on désire une précision élevée plus le rapport devient important. On constate de plus que la taille du problème considéré n'influe en rien sur la convergence des méthodes. Cette étude de la convergence étant faite, observons ces influences sur le gain de temps obtenu entre les deux méthodes. Nous nous restreindrons au cas du programme ParIS en raison de ses meilleures performances. En regroupant les deux dernières figures, on obtient la courbe 15 .

L'influence du VP-ratio, observée dans la figure 12, est encore plus flagrante sur cette courbe. Excepté le cas 128 qui permet d'obtenir un gain satisfaisant quelle que soit la précision demandée, les performances se

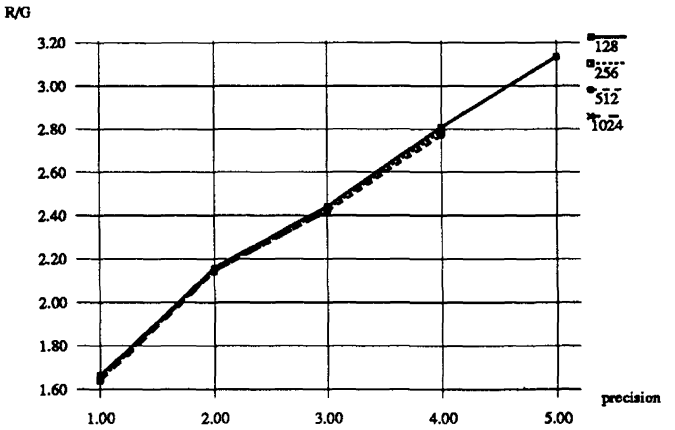

Figure 14. — Rapport entre les deux méthodes à itération fixée.

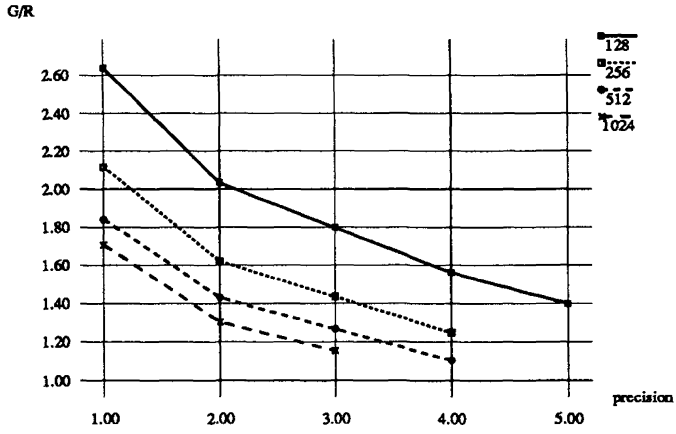

Figure 15. — Gain final obtenu par la méthode à paramètres constants.

dégradent rapidement avec la taille. Des expérimentations sur une CM2 de taille plus importante, atténueraient certainement cette dégradation. On remarque cependant que pour toutes les mesures effectuées, la méthode à paramètres constants reste meilleure que celle du Gradient Conjugué.

5. CONCLUSION

De toutes les expérimentations effectuées sur la Connection Machine dans le cas du Laplacien et pour tous schémas réguliers, on peut extraire deux grandes lignes : premièrement le manque de précision de la machine, et

deuxièmement l'effet du VP-ratio. Le phénomène de précision est un cas particulier dû à l'architecture même de la Connection Machine, et peut être pondéré par l'utilisation de processeurs 64 bits. L'effet du VP-ratio est aussi lié au problème de l'architecture des machines SIMD, mais son atténuation semble possible avec les projets de construction de machines avec un million de processeurs. On peut donc envisager l'avenir avec beaucoup d'espoir. Mais pour l'instant la méthode à paramètres constants est très efficace pour une classe de problèmes correspondant à la résolution de grands systèmes linéaires creux avec relativement faible précision comme en traitement du signal ou comme noyau de base de méthodes non linéaires.

REMERCIEMENTS

Nous tenons à remercier les *referees* pour leurs remarques constructives, en particulier sur les méthodes de recherche des valeurs propres extrêmes que nous avons utilisées lors de la rédaction finale.

RÉFÉRENCES

[1] S. F. ASHBY, T. A. MANTEUFFEL, P. E. SAYLOR, 1990, A Taxonomy For Conjugate Gradient Methods, *Siam J. Numer. Anal.*, 27.

[2] J. Y. BLANC, 1991, *Contribution du parallélisme à la résolution d'un problème de répartition de charge dans les réseaux électriques,* Thèse de l'institut polytechnique de Grenoble.

[3] P. CONCUS, G. GOLUB, D. P. O'LEARY, 1984, A Generalized Conjugate Gradient for the Numerical Solution of Elliptic Partial Differential Equations, in *Numerical Analysis,* série SIAM.

[4] D. DELESALLE, D. TRYSTRAM, D. WENZEK, 1990, *Tout ce que vous voulez savoir sur la Connection Machine,* Rapport de Recherche LMC-IMAG.

[5] L. DESBAT, 1990, *Critères de Choix des Paramètres de Régularisation : Application à la déconvolution,* Thèse de l'université Joseph Fourier, Annexe B : *Gradient Conjugué et parallélisme.*

[6] J. DEMMEL, J. J. DONGARRA, J. DUCROZ, A. GREENBAUM, S. J. HAMMARLING, D. C. SORENSEN, 1988, *A project for developing a Linear Algebra Library for high-performance computer,* Aspect of computation on asynchronous parallel processors, M. Wright.

[7] J. J. DONGARRA, I. S. DUFF, D. C. SORENSEN, H. A. VAN DER VORST, 1991, *Solving Linear Systems on Vector and Shared Memory Computers,* Siam.

[8] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH, G. W. STEWART, 1979, *LINPACK user's guide,* Siam philadelphia.

[9] M. J. FLYNN, 1972, Some computer organisations and their effectiveness, *IEEE Trans. on Computers C-21,* 9.

[10] G. FOX *et al.*, 1988, *Solving problems on concurrent processors : General techniques and regular problems* (vol. I), Prentice-Hall.

[11] G. H. GOLUB, G. MEURANT, 1983, *Résolution numérique des grands systèmes linéaires,* Eyrolles Paris, collection CEA/EDF.

[12] G. H. GOLUB, R. S. VARGA, 1961, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second other Richardson iterative methods,* Part I et Part II, Numerische Mathematik.

[13] G. H. GOLUB, C. F. VAN LOAN, 1989, *Matrix Computation,* Second edition, Johns Hopkins.

[14] J. GUSTAFSSON, G. LINDSKOG, 1986, A preconditioning technique based on element matrix factorisations, *Comp. Meth. Appl. Mech. Engng.*, 55.

[15] A. L. HAGEMAN and D. M. YOUNG, 1981, *Applied Iterative Methods,* Academic Press.

[16] G. L. HENNIGAN *et al.*, 1989, *A proposed domain decomposition technique for finite element on FPS T-serie,* Proceedings of 4th Conf. Hypercube.

[17] M. HESTENES, E. STIEFEL, 1952, Methods of Conjugate Gradient for Solving Linear Systems, *Journal Res. Nat. Bur. Stan.*, vol. 49.

[18] K. HWANG, F. A. BRIGGS, 1984, *Computer Architecture and Parallel Processing,* McGraw-Hill.

[19] O. G. JOHNSON, C. A. MICCHELLI and G. PAUL, 1983, Polynomial Preconditionnings for Conjugate Gradient Calculations, *SIAM J. Numer. Anal.*, vol. 20, pp. 362-376.

[20] P. LASCAUX, R. THEODOR, 1987, *Calcul matriciel appliqué à l'art de l'ingénieur,* Masson.

[21] P. LAURENT-GENGOUX, D. TRYSTRAM, 1988, *Parallel conjugate gradient algorithm with local decomposition,* Rapport de recherche TIM3-IMAG.

[22] O. A. MCBRYAN, 1989, *Connection Machine Application Performance,* Boulder Research report.

[23] THINKING MACHINE CORPORATION, 1991, *Connection Machine CM-200 Serie,* Technical Summary.

[24] Y. SAAD, 1983, *Practical use of polynomial preconditionings for the conjugate gradient method,* Yale Research report YALEU/DCS/RR-282.

[25] J. SALTZ, S. PETITON, H. BERRYMAN and A. RIFKIN, 1991, *Performance effects of irregular communications patterns on massively parallel multiprocessors,* NASA Contractor Report 187514.

[26] C. TONG, 1989, The Preconditioned Conjugate Gradient Method on the Connection Machine. *Int. Jour. of Hight Speed Comp.*, vol. 1.