RAIRO. RECHERCHE OPÉRATIONNELLE

P. TOLLA

# Contrôle et amélioration de la précision numérique des codes de programmation linéaire continue

*RAIRO. Recherche opérationnelle*, tome 11, n° 1 (1977), p. 109-121

<http://www.numdam.org/item?id=RO_1977__11_1_109_0>

R.A.I.R.O. Recherche opérationnelle/Operations research
(vol. 11, n° 1, février 1977, p. 109 à 121)

# CONTRÔLE ET AMÉLIORATION DE LA PRÉCISION NUMÉRIQUE DES CODES DE PROGRAMMATION LINÉAIRE CONTINUE (*)

par P. TOLLA (1)

Résumé. — *La mise en œuvre sur ordinateur de codes utilisant la méthode du Simplexe de Dantzig est souvent perturbée par des phénomènes liés à résolution des systèmes linéaires : les erreurs d'arrondi ou de troncature. Le praticien est souvent désarmé devant les tracas qu'ils produisent : solution numériquement fausse, ou non réalisable, matrice de base dégénérée... C'est pourquoi nous proposons un ensemble de méthodes qui permettent de détecter les erreurs, de les réduire par réinversion de la matrice de base, de contrôler la validité des solutions trouvées après réinversion. Le but de cette étude n'est pas de résoudre des programmes linéaires de grande taille, mais d'obtenir, sans faire appel à la double-précision, des solutions satisfaisantes, du point de vue de la précision numérique, de problèmes que l'on peut traiter sur des ordinateurs dotés d'une capacité de mémoire limitée.*

## I. TOUR D'HORIZON DES MÉTHODES DE RÉSOLUTION DE PROGRAMMES LINÉAIRES

### 1. Généralités

Soit le programme linéaire :

Minimiser $c.x$ sous les contraintes

$$\begin{cases} A.x = b, \\ x \geqq 0. \end{cases}$$

où $c^t \in \mathbf{R}^n$, $x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ et où A est une matrice à $m$ lignes et $n$ colonnes.

Une itération de la méthode du simplexe comporte essentiellement la résolution de trois systèmes linéaires :

$B^{(k)}\ u^{(k)} = b$, calcul de la solution de base $u^{(k)}$;

$\pi^{(k)}\ B^{(k)} = c_B(k)$, calcul du vecteur ligne $\pi^{(k)}$ des multiplicateurs du simplexe;

$B^{(k)}\ y^{(k)} = A_{s_k}$, calcul de la colonne entrante sur la base correspondant à $B^{(k)}$.

$B^{(k)}$ est ici la matrice de base de la $k$-ième itération, $b$ le vecteur second membre, $c_{B^{(k)}}$ le vecteur ligne des coefficients de la fonction économique sur la base, $A_{s_k}$ la colonne entrant dans la base.

La résolution séparée de ces trois systèmes exigerait un temps de calcul très long, dépendant du mode d'inversion utilisée. En programmation linéaire,

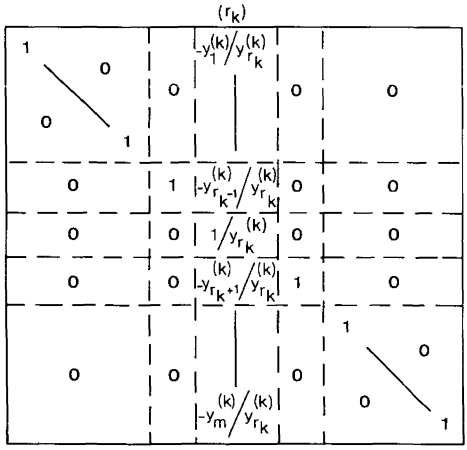
(*) Reçu septembre 1976.
(1) Institut de Programmation, Université Pierre-et-Marie-Curie, Paris.

on peut, au cours de la $k$-ième itération, exploiter l'information apportée par la $(k-1)$-ième itération pour réduire le nombre d'opérations de calcul nécessaires à la résolution des trois systèmes linéaires.

## 2. Méthodes de l'inverse explicite et P.F.I.

On dispose à chaque itération de toute l'information nécessaire au calcul de l'inverse de la matrice de base, $[B^{(k)}]^{-1}$; ceci nous permet de résoudre très simplement les trois systèmes linéaires. D'autre part les deux méthodes s'appuient sur le fait que l'inverse de la matrice de base de la $(k)$-ième itération est calculée en multipliant l'inverse de la matrice de base de la $(k-1)$-ième itération par une matrice $E_k$ d'élimination :

$E_k =$




où $r_k$ est le numéro de ligne du pivot.

D'où $[B^{(k)}]^{-1} = E_k [B^{(k-1)}]^{-1}$, ce qui évite de procéder à une inversion complète de la matrice de base à chaque itération, opération qui impose un ordre de $m^3/2$ opérations de multiplication; la mise à jour de $[B^{(k)}]^{-1}$ permet de réduire le nombre de calculs à un ordre de $m^2$ opérations de multiplication.

## 3. La méthode LU classique

$B^{(k)}$ est décomposée en le produit d'une matrice triangulaire inférieure $L^{(k)}$ et d'une matrice triangulaire supérieure $U^{(k)}$ de façon que

$$|l_{ij}| \leqq 1 \quad \text{si} \quad i > j \quad \text{et} \quad l_{ii} = 1,$$

et on est ainsi ramené à la résolution de systèmes triangulaires. On met à jour cette décomposition, d'une itération à la suivante, en ôtant la $r_k$-ième colonne de $U^{(k)}$ et en la remplaçant par $[L^{(k)}]^{-1} A_{s_k}$. On triangularise la matrice

ainsi obtenue en éliminant les éléments sous-diagonaux non nuls; la matrice $[L^{(k)}]^{-1}$ est mise à jour en la prémultipliant par les matrices d'élimination. Cette méthode exige un nombre d'opérations de calcul de l'ordre de $m^3/4$ (*cf.* [3]), ce qui est très coûteux vis-à-vis de la méthode inverse explicite qui en demande seulement un ordre de $m^2$. Par contre, la liberté du choix des pivots dans l'élimination des éléments sous-diagonaux permet, selon Wilkinson [10] d'assurer une meilleure précision numérique.

## 4. La méthode LU modifiée de Bartels et Golub [2, 3]

Le principe est identique à celui de la méthode LU classique : la principale différence réside dans la mise à jour de la décomposition LU : on ôte la $r_k$-ième colonne de la matrice $U^{(k)}$, et on décale d'une position vers la gauche toutes les colonnes lui succédant; on met en dernière position $[L^{(k)}]^{-1} A_{s_k}$. On obtient ainsi une matrice, du type décrit par la figure I.1, qui est presque

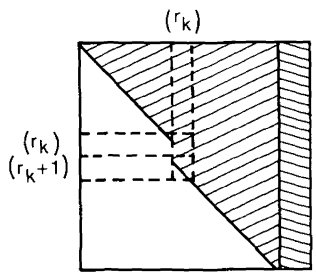



Figure I.1.

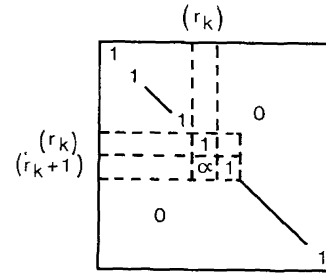



Figure I.2.

triangulaire supérieure; seuls les coefficients placés exactement sous la première diagonale à partir de la $r_k$-ième colonne ne sont pas nuls; il est donc simple de la triangulariser en la prémultipliant par des matrices d'élimination qui se réduisent à la matrice identité dans laquelle on trouve un seul élément non nul placé exactement sous la diagonale (*fig.* I.2).

On réduit donc de façon sensible le nombre d'opérations arithmétiques, puisqu'il devient de l'ordre de $m^2$, et donc très comparable à ceux des méthodes, inverse explicite et P.F.I. D'autres procédés d'actualisation de la décomposition LU ont été proposés en particulier par Tomlin [7].

## II. RÉDUCTION DES NOMBRES D'OPÉRATION DE CALCUL DANS LES MÉTHODES LU

1. $B^{(k)}\ u^{(k)} = b$

On en tire :

$$u^{(k)} = [B^{(k)}]^{-1}\ b = E_k [B^{(k-1)}]^{-1}\ b = E_k u^{(k-1)}.$$

On déduit donc la solution de base de la $k$-ième itération de celle de la $(k-1)$-ième itération à l'aide des relations suivantes :

$$u_{r_k}^{(k)} = u_{r_k}^{(k-1)} / y_{r_k}^{(k-1)},$$

$$u_i^{(k)} = u_i^{(k-1)} - u_{r_k}^{(k-1)} \frac{y_i^{(k-1)}}{y_{r_k}^{(k-1)}}, \qquad i \neq r_k.$$

Cette propriété est couramment utilisée dans les codes de programmation linéaire.

2. $B^{(k)}\, y^{(k)} = A_{s_k}$

On ne peut apporter d'amélioration à la résolution de ce système linéaire puisque le numéro de la colonne entrant dans la base est trouvé à la fin de la $(k-1)$-ième itération, et c'est la seule information dont on dispose à propos de $A_{s_k}$.

3. $\pi^{(k)}\, B^{(k)} = c_{B^{(k)}}$

$$c_{B^{(k)}} = (c_{v_1}, \ldots, c_{v_{r_k-1}}, c_{v_{r_k}}, c_{v_{r_k+1}}, \ldots, c_{v_m}),$$

$$c_{B^{(k-1)}} = (c_{v_1}, \ldots, c_{v_{r_k-1}}, c_{s_k}, c_{v_{r_k+1}}, \ldots, c_{v_m}),$$

$c_{B^{(k-1)}}$ diffère donc de $c_{B^{(k)}}$ d'un seul coefficient.

Or

$$\pi^{(k)} B^{(k)} = c_{B^{(k-1)}} \qquad \text{et} \qquad \pi^{(k-1)} B^{(k-1)} = c_{B^{(k-1)}}.$$

D'autre part $[B^{(k-1)}]^{-1} = [E_k]^{-1} [B^{(k)}]^{-1}$ :

$$\pi^{(k-1)} = c_{B^{(k-1)}} [E_k]^{-1} [B^{(k)}]^{-1},$$

$$\pi^{(k)} - \pi^{(k-1)} = (c_{B^{(k)}} - c_{B^{(k-1)}} [E_k]^{-1}) [B^{(k)}]^{-1}.$$

On en tire :

$$c_{B^{(k-1)}} [E_k]^{-1} = (c_{v_1}, \ldots, c_{v_{r_k-1}}, \sum_i c_{v_i} y_i^{(k)}, c_{v_{r_k+1}}, \ldots, c_{v_m}),$$

$$\pi^{(k)} - \pi^{(k-1)} = (0, \ldots, 0, c_{s_k} - \sum_i c_{v_i} y_i, 0, \ldots, 0) [B^{(k)}]^{-1}.$$

Dans les méthodes de type LU, les calculs sont ainsi réduits à la résolution de deux sous systèmes linéaires de taille $m - r_k$ au lieu de $m$. Nous avons montré dans [6] que ces diverses remarques permettant de réduire le nombre d'opérations arithmétiques dans des proportions sensibles; en particulier pour la méthode LU modifiée ou un calcul du nombre moyen d'opérations produit donne

$$\frac{14}{3} m^2 + \frac{1}{2} m - \frac{1}{6}$$

lorsqu'on n'utilise pas les remarques ci-dessus,

$$\frac{17}{6}m^2+4m-\frac{5}{6}$$

quand on les utilise. La propriété exposée au paragraphe (3) permet de réduire en moyenne le nombre d'opérations de multiplication nécessaires à la résolution du système $\pi^{(k)} B^{(k)} = c_{B^{(k)}}$ de $\frac{3}{2}m^2+\frac{1}{2}m$ à $\frac{2}{3}m^2+m+\frac{1}{2}$.

La propriété exposée au paragraphe (1) permet quant à elle de réduire le nombre d'opérations de multiplication nécessaires à la résolution du système $B^{(k)} u^{(k)} = b$ de $\frac{3}{2}m^2+m$ à $m$.

## III. CONTRÔLE DE LA SOLUTION DES SYSTÈMES LINÉAIRES

### 1. Majorants d'erreur d'arrondi

Les moyens de contrôle des erreurs d'arrondi en programmation linéaire étaient très peu développés lorsque, s'appuyant sur les travaux de Wilkinson, [10], R. Bartels a proposé un calcul de majorants d'erreur d'arrondi commise dans la résolution des différents systèmes linéaires. Nous avons montré dans [6] que Bartels avait très sous-estimé ces majorants. En fait la notion de majorant d'arrondi est peu utilisable pour le contrôle de la solution des systèmes linéaires en programmation linéaire; en effet il est très difficile de cerner de près l'erreur avec des majorants qui lui sont très supérieurs en réalité. Par contre nous avons proposé dans [6] des majorants qui tiennent compte de la position des pivots en ligne, et reflètent non l'erreur d'arrondi, mais plutôt son évolution.

Supposons qu'après une réinversion qui a décomposé la matrice de base $B$ en $L$ et $U$, on procède à $k$ itérations supplémentaires; la matrice de base $B^{(k)}$ de la $k$-ième itération s'écrira :

$$B^{(k)} = LG^{(k)} U^{(k)},$$

avec

$$G^{(k)} = \{\Pi_{r_1}^{(1)}[\Gamma_{r_1}^{(1)}]^{-1}\ldots\Pi_{m-1}^{(1)}[\Gamma_{m-1}^{(1)}]^{-1}\}$$
$$\times\ldots\times\{\Pi_{r_k}^{(k)}[\Gamma_{r_k}^{(k)}]^{-1}\ldots\Pi_{m-1}^{(k)}[\Gamma_{m-1}^{(k)}]^{-1}\}\,U^{(k)},$$

où les matrices $\Pi_i^{(l)}$ sont, pour la $l$-ième itération, la matrice identité si le pivot correspondant est sur la diagonale de la matrice à triangulariser, la matrice de permutation des $i$-ième et $(i+1)$-ième lignes sinon; les

matrices $\Gamma_l^{(l)}$ sont les matrices d'élimination des éléments sous-diagonaux dans la méthode de Bartels.

Lorsqu'on résout $B^{(k)} v = q$ à l'aide de cette décomposition, on obtient les vecteurs $f$, $w$ et $t$ vérifiant :

$$(L+\delta L)w = q,$$
$$(G^{(k)}+\delta G^{(k)})f = w,$$
$$(U^{(k)}+\delta U^{(k)})t = f.$$

$t$ est alors la valeur calculée de $v$, et sera solution du système

$$(B^{(k)}+\varepsilon^{(k)})t = (L+\delta L)(G^{(k)}+\delta G^{(k)})(U^{(k)}+\delta U^{(k)})t = q.$$

Comme $B^{(k)}+\delta B^{(k)} = LG^{(k)}\,U^{(k)}$, on obtient :

$$\varepsilon^{(k)} = \delta B^{(k)} + LG^{(k)}\delta U^{(k)} + L\delta G^{(k)}\,U^{(k)} + \delta L G^{(k)}\,U^{(k)} + L\delta G^{(k)}\delta U^{(k)}$$
$$+\delta L G^{(k)}\delta U^{(k)} + \delta L\delta G^{(k)}\,U^{(k)} + \delta L\delta G^{(k)}\delta U^{(k)}.$$

Nous avons proposé dans [6] des majorants de $\| \delta B^{(k)} \|$, $\| L \|$, $\| G^{(k)} \|$, $\| U^{(k)} \|$, $\| \delta L \|$, $\| \delta G^{(k)} \|$, $\| \delta U^{(k)} \|$ qui permettront de calculer un majorant de $\| \varepsilon^{(k)} \|$.

$$\| \delta L \| < m\left\{(m+2)\varepsilon_2' + \frac{\varepsilon_1}{1-\varepsilon_1}\left[1+(m-1)\varepsilon_2'\right]\right\},$$

où

$$\varepsilon_2' = \frac{1}{m+2}\left\{\exp\left[\frac{(m+2)\,\varepsilon_1}{1-\varepsilon_1}\right]-1\right\}$$

où $\varepsilon_1$ est un majorant en valeur absolue de l'erreur commise lorsqu'on fait le produit de deux termes en simple précision

$$\| \delta G^{(k)} \| < T^{(k)},$$

où

$$T^{(k)} = \left\{\sum_{k'=1}^{k}\left\{\prod_{k''<k'}, \beta_{r_{k''}}^{m}\left[\sum_{l=r_{k''}}^{m}\beta_{r_{k''},\,l}'^{m}\,\beta_{l+1}'^{m}\right]\prod_{k''>k'}\beta_{r_{k''}}'^{m}\right\}\right\}\left[\frac{(3+\varepsilon_1)^2_{+1}}{(1-\varepsilon_1)^2}\right]\varepsilon_1,$$

où

$$\beta_{r_k}^{m} \approx \beta_{r_k}'^{m} = \left[2m-r_k\right]^{1/2}$$

et

$$\beta_{r_k,\,l}^{m} \approx \left[m-1+2l-2r_k\right]^{1/2},$$

$$\| \delta U^{(k)} \| \leqq m\left\{(m+1)\varepsilon_2' + \frac{(2\varepsilon_1-\varepsilon_1^2)}{(1-\varepsilon_1)^2}\left[1+(m-1)\varepsilon_2'\right]\right\}\max_{i<j}\{|U_{ij}|\},$$

$$\| L \| \leqq m,$$

$$\| U^{(k)} \| \leqq m \max_{i,\,j}\{|U_{ij}^{(k)}|\},$$

$$|| G^{(k)} || \leqq \prod_{i=1}^{m} \beta'^{m}_{r_i}.$$

Un majorant de $|| \delta B^{(k)} ||$ est calculé de façon itérative : on trouvera son expression dans [6]. En faisant de même pour $T^{(k)}$, on obtient pour le calcul de $|| \varepsilon^{(k)} ||$ un nombre d'opérations produit de l'ordre de 1/2 km.

### 2. Contrôle statistique des erreurs d'arrondi et de troncature

La Porte et Vignes dans [8] et [9], proposent un contrôle statistique d'erreur d'arrondi ou de troncature qui s'est avéré très efficace : Soit $Ax = b$ le système à résoudre; $\rho = b - Ax$, le vecteur des résidus calculés; on détermine une évaluation du vecteur résidu théorique $\hat{\rho}$ avec :

$$\hat{\rho}_i = 2^{-n} \sqrt{N_i (\sum a_{ij}^2 x_j^2 + b_i^2)} \quad \text{pour les erreurs de troncature,}$$

$$\hat{\rho}_i = 2^{-n} N_i \sqrt{\sum a_{ij}^2 x_j^2 + b_i^2} \quad \text{pour les erreurs d'arrondi,}$$

$N_i$ est le nombre de coefficients $a_{ij}$ non nuls dans la $i$-ième ligne de $A$, et $n$ le nombre de bits de la mantisse pour l'ordinateur utilisé.

Soit $\rho_i^* = | \rho_i | / \hat{\rho}_i$ le résidu normé pour chaque ligne; trois cas peuvent être envisagés :

(*a*) Les résidus normés sont tous de l'ordre de grandeur de l'unité : la solution est considérée comme informatiquement satisfaisante.

(*b*) L'un au moins des résidus normés est nettement supérieur à l'unité, mais nettement inférieur à $2^n$ : la solution n'est pas pleinement satisfaisante et il est conseillé d'utiliser l'affinage itératif qui consiste à résoudre $A . \Delta x = \rho$, puis prendre comme nouvelle solution $x + \Delta x$.

(*c*) L'un au moins des résidus normés est de l'ordre de grandeur de $2^n$ : la solution est mauvaise; elle ne peut généralement pas être améliorée; cela signifie que la méthode de résolution est mal adaptée au système proposé.

## IV. DÉTECTION DES ERREURS ET AMÉLIORATION DE LA PRÉCISION EN PROGRAMMATION LINÉAIRE

Il est peu utile et très coûteux de contrôler l'erreur au cours de chaque itération de la méthode du simplexe; il faut donc disposer de moyens de détection de l'erreur, d'amélioration de la précision par réinversion de la matrice de base et de contrôle de la précision numérique de la solution trouvée par réinversion.

### 1. Détection d'erreur

Un critère simple consiste à calculer les coefficients de coût relatif sur la base, $\Delta_{B^{(k)}}$; comme $\Delta_{B^{(k)}} = 0$, les valeurs obtenues sont exactement les erreurs commises sur ces coefficients.

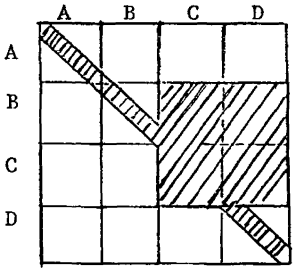
On peut alors procéder de deux façons différentes :

(*a*) Dans les premières itérations, on calcule simultanément $\Delta_{B^{(k)}}$ et le majorant correspondant de $\| \varepsilon^{(k)} \|$; lorsqu'une au moins des composantes de $\Delta_{B^{(k)}}$ est supérieure en valeur absolue à $10^{-3}$ par exemple, le majorant calculé est conservé comme seuil d'erreur, et on ne recalcule plus $\Delta_{B^{(k)}}$ dans les itérations suivantes, mais uniquement le majorant; lorsque celui-ci dépasse le seuil ainsi fixé, la connaissance des numéros des colonnes de base permet de reconstituer $B^{(k)}$ et de procéder à une décomposition de $B^{(k)}$ en $L'^{(k)}$ et $U'^{(k)}$ avec maintenant la possibilité de choisir des pivots susceptibles d'assurer une meilleure précision numérique.

(*b*) On se fixe un seuil d'erreur, par exemple $10^{-3}$, et on calcule au cours de chaque itération $\Delta_{B^{(k)}}$; lorsqu'une au moins des composantes de $\Delta_{B^{(k)}}$ dépasse en valeur absolue ce seuil, on procède à une décomposition de $B^{(k)}$ de façon analogue à celle exposée au paragraphe (*a*).

## 2. Réinversion

La réinversion doit être adaptée au mode d'actualisation de $[B^{(k)}]^{-1}$ ou $L^{(k)}\ U^{(k)}$ et suffisamment précise. Pour cela, on peut utiliser le creux de la matrice et l'existence de singletons en ligne et en colonne. On trouve fréquemment des singletons dans $B^{(k)}$ en raison de la présence de colonnes de la matrice de base canonique initiale. Nous avons proposé dans [6] un mode d'organisation de $B^{(k)}$ tenant compte des singletons.




On trouve dans le bloc *AA* les singletons en ligne et en colonne, dans *BB* les singletons en colonne et dans *DD* les singletons en ligne. Cette disposition est très utile, en particulier pour la méthode LU, car elle permet de ne trianguläriser que le bloc *CC*.

On peut d'autre part tenir compte du « creux » de ce bloc en essayant par exemple d'en trianguläriser une partie par simple permutation de lignes et de colonnes, ce qui ne modifie pas la structure globale de la matrice. On contrôle ensuite les résultats à l'aide du procédé de Vignes et La Porte exposé au paragraphe III.2.

Orchard-Hays propose dans [5] une organisation un peu différente telle que :

- les blocs $AA$ et $CC$ sont triangulaires inférieurs;
- les singletons sont répartis sur la diagonale du bloc $DD$;
- les blocs $BA$, $CA$, $DA$, $DB$ ont un faible taux de coefficients non nuls;
- les blocs $AB$, $AC$, $AD$, $BC$, $BD$ et $CD$ ne comportent que des zéros.

Cette disposition permet, dans la triangularisation, d'obtenir une matrice $L$ dont le taux de remplissage des blocs $BA$, $CA$, $DA$ et $DB$ ne soit pas trop augmenté, et de faire des choix partiels de pivots.

## V. PROGRAMMES ET RÉSULTATS

### 1. Les codes

Nous avons programmé deux codes :

Le premier code utilise la méthode inverse explicite; au cours de chaque itération on calcule le vecteur $\Delta_{B^{(k)}}$ : lorsque la plus grande composante, en valeur absolue, de ce vecteur est supérieure à un seuil fixé, par exemple $10^{-3}$, le code appelle le sous-programme REINV 1 de réinversion qui n'est autre qu'une méthode de Jordan avec recherche, en colonne, de pivot de plus grande valeur absolue.

Le second code, basé sur la méthode LU avec la mise à jour de la décomposition LU de la matrice de base proposée par Bartels, comporte le test d'erreur sur les $\Delta_{B^{(k)}}$ exposé au paragraphe IV.1.*b.* avec calcul du majorant de $\| \varepsilon^{(k)} \|$ proposé au paragraphe III.1; lorsque le majorant dépasse le seuil fixé, le programme fait appel au sous-programme REINV 2 de réinversion, qui décompose la matrice de base $B^{(k)}$ en $L^{(k)}$ et $U^{(k)}$ selon la méthode de Gauss avec recherche, en colonne et sous la diagonale, de pivot de plus grande valeur absolue.

Le manque de précision des méthodes est du essentiellement au fait que, dans la méthode du simplexe, les pivots sont fixés par le deuxième critère de Dantzig de façon à fournir à chaque itération une solution de base réalisable; l'intérêt de la réinversion est que, la connaissance des numéros des colonnes de base permet de recréer la matrice de base $B^{(k)}$, et de procéder à une inversion ou à une décomposition LU de celle-ci tenant compte uniquement de la valeur des coefficients de $B^{(k)}$, ce qui est censé assurer une précision numérique supérieure. Cependant, si la précision est meilleure, elle n'est peut-être, pas encore, satisfaisante; c'est pourquoi, dans les deux codes, le sous-programme CØNTRØ contrôle la validité de la solution de base obtenue immédiatement après la réinversion, à l'aide du critère des résidus normés de La Porte et Vignes : lorsque le plus grand des résidus normés est supérieur à l'unité, on procède à l'affinage itératif de la solution jusqu'à ce que tous les résidus soient inférieurs à l'unité.

Il est possible de contrôler et améliorer à l'aide de CØNTRØ, au cours de chaque itération, la solution de chacun des trois systèmes, mais le calcul des résidus normés, et l'affinage itératif demandent pour chacun d'eux un ordre de $2\,m^2$ opérations de multiplication au minimum, ce qui revient à tripler le temps de calcul dans l'hypothèse où on ne procède qu'à une seule opération d'affinage itératif pour chaque système !

A l'optimum, la solution de base est de nouveau contrôlée et améliorée par le sous-programme CØNTRØ.

## 2. Les résultats

Les deux codes ont été principalement testés sur deux exemples : un problème d'investissements de l'E.D.F., à 44 contraintes et 53 variables, et un problème pétrolier à 33 contraintes et 33 variables dû à Maurin [4].

*Évolution de l'erreur d'arrondi :* nous avons étudié l'évolution de l'erreur d'arrondi dans la méthode LU modifiée, en ôtant les sous-programmes de réinversion et de contrôle d'erreur, sur environ 80 itérations, et en considérant, à chaque itération le plus grand coefficient, en valeur absolue, de $\Delta_{B^{(k)}}$; comme l'indique la figure V.1, l'augmentation, quoique nette dans les premières itérations, devient ensuite assez désordonnée; il est alors évident qu'un majorant qui est croissant avec les itérations va mal refléter ces erreurs.

*Comparaison des méthodes :* pour contrôler les résultats, nous disposions de la solution du problème d'investissement de l'E.D.F., fournie par le code G.R.G. d'Abadie, [1], sur un ordinateur CDC 6600 qui comporte en simple-précision quatorze chiffres significatifs; le problème pétrolier avait été traité sur un ordinateur CDC 3600 qui comporte le même nombre de chiffres significatifs que le CDC 6600. Nous avons, quant à nous, utilisé les deux codes sur un CII 10070 qui comporte seulement six à sept chiffres significatifs en simple précision.

– Pour comparer la précision numérique des méthodes de mise à jour de l'inverse de la matrice de base $B^{(k)}$ et de la décomposition de $B^{(k)}$ en $L^{(k)}$ et $U^{(k)}$, nous avons testé les deux codes sans y adjoindre les sous-programmes de contrôle d'erreur et de réinversion : la méthode LU a fourni dans les deux cas une base optimale exacte, tandis que les composantes de la solution de base correspondante avaient toutes au moins un chiffre significatif exact; par contre la méthode inverse explicite a donné une base fausse dans la résolution du problème pétrolier, alors que, dans le cas du problème de l'E.D.F., la base était exacte, mais certaines composantes de la solution de base, tout en étant d'un bon ordre de grandeur, ne comportaient aucun chiffre significatif exact.

– Les deux codes ont donné de bons résultats lorsqu'on y a incorporé les sous-programmes de réinversion et de contrôle d'erreur. Cependant la méthode LU s'est avérée plus précise, puisque dans tous les cas, les compo-

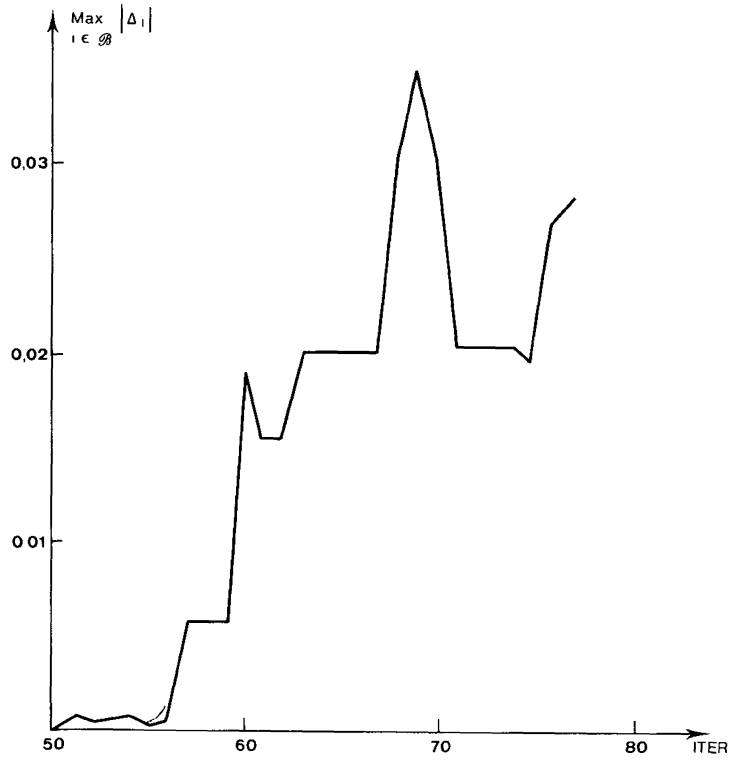



**Figure V.1.**

**Méthode LU sans réinversion sur le problème de la raffinerie. Évolution de la plus grande composante en valeur absolue de $\Delta_{B^{(k)}}$. $\mathscr{B}$ est l'ensemble des indices des variables de base.**

santes de la solution de base optimale comportaient au moins cinq chiffres significatifs exacts contre deux pour la méthode inverse implicite. Par contre, il arrive que la méthode LU fournisse des solutions intermédiaires très fausses : on a pu alors remarquer que certaines composantes du vecteur $\Delta_{B^{(k)}}$ deviennent très grandes devant 0 (de l'ordre de 15); la réinversion à ce niveau permet de réduire l'erreur de façon appréciable puisque les composantes de $\Delta_{B^{(k)}}$ deviennent nulles au cours de l'itération suivante (*fig.* V.2). Le critère qui consiste à vérifier la nullité des composantes de $\Delta_{B^{(k)}}$ semble donc très efficace.

– L'utilisation comme pivot du coefficient de plus grande valeur absolue en colonne, dans les deux sous-programmes de réinversion, a permis d'obtenir des solutions de base précises; il faut remarquer que le sous-programme de réinversion de la méthode LU n'a jamais fait appel à l'affinage itératif; dans le sous-programme de réinversion de la méthode inverse explicite, le procédé qui consiste à prendre les éléments de la diagonale de $B^{(k)}$ comme pivots, a

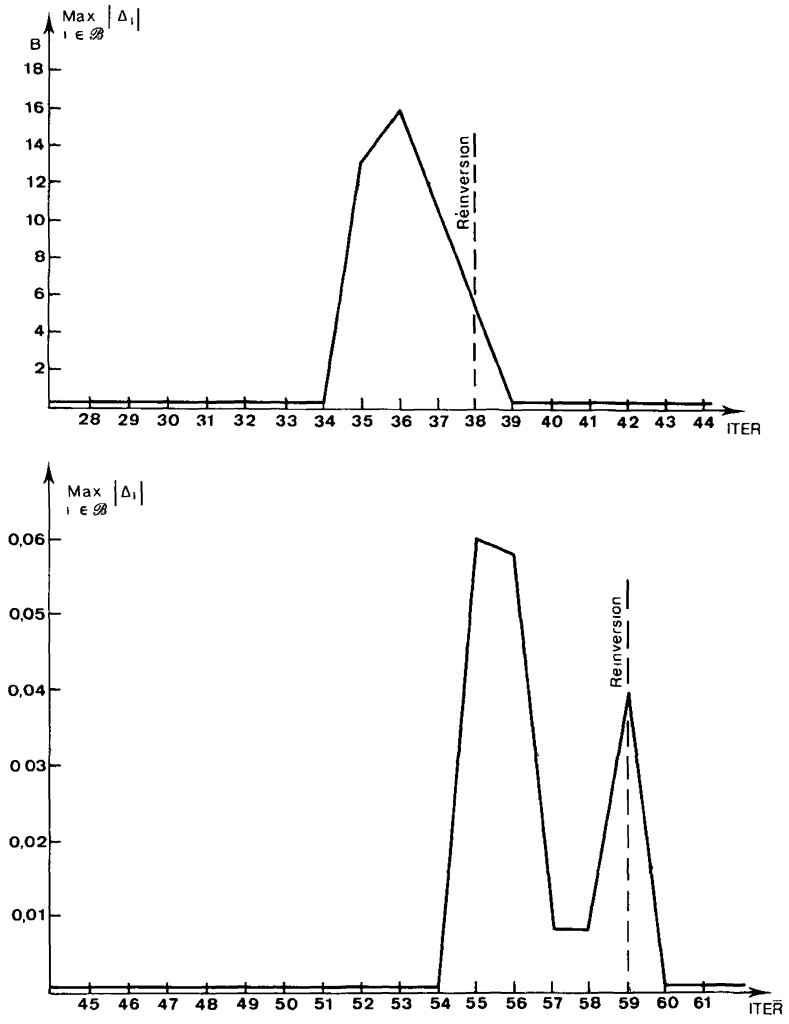



**Figure V.2.**

**$\mathscr{B}$ est l'ensemble des indices des variables de base. Méthode LU avec réinversion sur le problème E.D.F. Évolution de la plus grande composante en valeur absolue de $\Delta_{B^{(k)}}$.**

abouti à une solution inacceptable et non améliorable au sens de La Porte et Vignes, alors que le choix du pivot maximal en colonne permet de trouver une solution acceptable après trois appels successifs à l'affinage itératif.

— L'organisation de la matrice de base à l'aide du sous-programme ØRDØ avant chaque réinversion, a permis, lorsqu'on utilisait le code LU, de calculer

exactement des composantes de la solution de base qui, dans le cas contraire, comportaient seulement quatre chiffres significatifs exacts.

## VI. CONCLUSION

Les codes que nous avons mis au point ont résolu de façon satisfaisante des problèmes dont les solutions obtenues à l'aide de codes classiques étaient peu précises ou fausses. On peut aller encore plus loin dans la recherche de la précision en utilisant la méthode de permutation de La Porte et Vignes pour calculer statistiquement le nombre de chiffres significatifs exacts des composantes de la solution; cependant, ce procédé est très coûteux car on doit résoudre plusieurs systèmes linéaires supplémentaires pour déterminer les nombres de chiffres significatifs exacts, et de plus il faut valider à chaque résolution la solution obtenue à l'aide du critère des résidus normés. D'autre part, dans la réinversion, la méthode de Gauss, avec recherche de pivot maximal en colonne est numériquement peu stable. C'est pourquoi nous développons actuellement des méthodes destinées à fournir des solutions de base dont les composantes ont un nombre de chiffres significatifs exacts suffisant lorsque la méthode de Gauss avec recherche de plus grand pivot en colonne échoue. Nous envisageons d'autre part d'utiliser les méthodes, que nous avons exposé ici, dans des codes de résolution de programmes non linéaires, et en particulier le code G.R.G. d'Abadie, [1], qui possède déjà à plusieurs niveaux, des procédures de traitement de la dégénerescence, phénomène qui peut s'avérer fort difficile à résoudre si les solutions de base sont peu précises.

## BIBLIOGRAPHIE

1. J. ABADIE, *Integer and Non Linear Programming*, North Holland, 1970.
2. R. H. BARTELS, *A Numerical Investigation of the Simplex Method*, Thèse Stanford, 1968.
3. R. H. BARTELS, G. H. GOLUB et M. A. SAUNDERS, *Numerical Techniques in Mathematical Programming* dans *Non Linear Programming*, J. B. ROSEN, O. L. MANGASARIAN et K. RITTER éd., Academic Press, New York, London, 1970.
4. H. MAURIN, *Programmation linéaire appliquée*, Technip, 1967.
5. W. ORCHARD HAYS, *Advanced Linear Programming Computing Techniques*, McGraw-Hill, 1968.
6. P. TOLLA, *méthodes de triangularisation et programmes d'ordinateur assurant l'amélioration de la précision en programmation linéaire*, Thèse de 3e cycle, 1974, Université Pierre-et-Marie-Curie.
7. J. A. TOMLIN, *Maintaining a Sparse Inverse in the Simplex Method*, Technical Report n° 70-16, novembre 1970, Operation Research House. Stanford University.
8. J. VIGNES et M. LA PORTE, *Évaluation statistique des erreurs numériques dans les calculs sur ordinateur, application au contrôle des solutions des systèmes linéaires et des équations algébriques*, Canadian Conférence, 1972.
9. J. VIGNES et M. LA PORTE, *Étude statistique des erreurs dans l'arithmétique des ordinateurs. Application au contrôle des résultats d'algorithmes numériques*, Numer. Math. vol. 23, 1974, p. 63-72.
10. J. H. WILKINSON, *Rounding Errors in Algebric Processes*, London, H.M.S.O. 1963.