Bibliothèques pour le calcul numérique parallèle ; synthèses et perspectives

Serge G. Petiton * et B. Tourancheau †

15 janvier 1997

Résumé

Après un historique des bibliothèques et la mise en évidence de leur intérêt pour le calcul numérique, nous décrivons les bibliothèques les plus connues en indiquant leurs caractéristiques. Nous présentons l'orientation des bibliothèques les plus récentes et leurs évolutions. Nous expliquons les placements des données et la structure de la bibliothèque ScaLAPACK largement repris par les constructeurs. Les récentes bibliothèques PETSc, PARPACK et P-SPARLIB sont ensuite décrites et certaines caractéristiques de leur parallélisation sont discutées. En conclusion, nous mettons en évidence quelques problèmes non-traités par les bibliothèques actuelles et nous présentons des perspectives de recherche et développement pour les bibliothèques numériques parallèles.

1 Introduction

Lors de l'étude d'un problème de calcul numérique, après les phases de modélisation et de discrétisation, on se ramène souvent à des méthodes explicites ou implicites. Si des méthodes explicites sont utilisées, on a recourt à des bibliothèques spécifiques qui sont très ciblées. Les méthodes implicites sont très utilisées pour des problèmes complexes et coûteux en temps de calcul ce qui justifie l'utilisation de machines parallèles. Les méthodes utilisées requièrent en général plusieurs pas de temps et à chaque pas de temps, il faut résoudre un problème d'algèbre linéaire.

Les méthodes utilisées pour résoudre les très grands problèmes scientifiques demandent souvent de résoudre des problèmes d'algèbre linéaire très difficile numériquement. Le scientifique voulant résoudre de tels problèmes n'a pas toujours les compétences en analyse numérique pour développer des méthodes efficaces et numériquement stables, ni en informatique pour en tirer des algorithmes performants et des logiciels fiables. Les nouvelles bibliothèques permettent souvent, avec certaines restrictions que l'on discutera plus loin dans cet article, de libérer l'utilisateur de ces contraintes.

*. Laboratoire d'Informatique Fondamentale de Lille, URA CNRS 369, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France, Serge.Petiton@lifl.fr

†. LIGIM UCB Lyon1, 69622 Villeurbanne, France et LIP/LHPC-INRIA Rhône-Alpes, ENS-Lyon, 69364 Lyon, Bernard.Tourancheau@inria.fr

Les bibliothèques d'algèbre linéaire généralistes sont donc actuellement très utilisées et nous les présentons dans la suite de ce chapitre comme représentatives des développements actuels [1].

2 Historique des bibliothèques, de LINPACK à ScaLAPACK et PETSc

2.1 Intérêt des bibliothèques

Les bibliothèques pour le calcul numérique regroupent des ensembles de sous programmes réalisant les fonctions mathématiques et logiques de base utilisées pour la construction des logiciels scientifiques de calcul numérique.

Leur premier but était un gain de temps de développement grâce à la simplification de la création des logiciels par réutilisation de sous programmes existants. L'exemple le plus connu est celui de la bibliothèque BLAS, Basic Linear Algebra Subroutines, écrite en Fortran77.

L'expérience liée au développement et à l'utilisation de bibliothèque numérique a permis de remarquer la robustesse des fonctions de bibliothèque (car elles sont utilisées par beaucoup de programmeurs dans le monde et sont très rapidement dégagées de leurs imperfections) et leur bon niveau de performance (qui est obtenu par une programmation soignée et optimisée). On peut noter aussi souvent une très bonne qualité arithmétique et la présence de contrôle d'erreur en virgule flottante. De plus, la disponibilité des bibliothèques sur différents types d'ordinateurs entraîne une forte portabilité des programmes qui les utilisent. Par exemple, les BLAS sont disponibles en version assembleur, optimisée du point de vue des niveaux de cache, sur tous les supercalculateurs (CRAY, CONVEX, DEC, HP, NEC, SGI, ...) et pour tous les processeurs du commerce.

L'utilisation de bibliothèques augmente donc la portabilité des programmes, diminue le temps de développement et de validation et permet souvent un gain en performances.

2.2 Les bibliothèques scalaires

2.3 Matrices denses

Les bibliothèques scalaires développées depuis les années 70 sont souvent à l'origine des bibliothèques parallèles actuelles. La première réellement utilisée fut la bibliothèque LINPACK, dès 1974. C'est par là même, la première apparition des BLAS favorisée par les débuts de la programmation structurée et des langages procéduraux (souvent Fortran). Rapidement, les constructeurs ont proposé des implémentations spécialisées, voir câblées, des BLAS sur leurs machines pour offrir plus de performances aux utilisateurs. Les méthodes implémentées sont principalement les méthodes directes pour des matrices denses ou des matrices bandes. Cette bibliothèque a été développée par DONGARRA,

1. Par manque d'espace, nous ne pouvons traiter dans ce texte les bibliothèques de FFT, méthodes explicites, mailleurs, méthodes de Newton, éléments spectraux, ...

Bunch, Moler et Stewart avec l'aide du numéricien Wilkinson. C'est aussi la première apparition du MACHEPS (Epsilon Machine) utilisé pour les tests de convergence numérique. Vers la même époque, EISPACK est apparue, c'est un peu le pendant *valeur propre* de LINPACK. Elle permet le calcul de vecteurs et de valeurs propres pour des matrices denses essentiellement en utilisant la méthode QR. Il n'y avait pas de méthode itérative de type Krylov qui sont à l'heure actuelle omniprésentes dans les bibliothèques parallèles aussi bien pour la résolution de systèmes linéaires que pour le calcul de vecteurs et valeurs propres. Cette bibliothèque a été développée entre autres par Garbow, Boyle, Dongarra et Moler. Ces deux bibliothèques ont connu un succès remarquable vers la fin des années 70. Elles ont été diffusées dans le monde entier et utilisées de manière intensive car, en particulier, elles sont gratuites. Puis est apparue LAPACK, implémentée en Fortran. On trouve aussi maintenant une version C (CLAPACK) et une version C++ (LAPACK++). Ces versions sont moins complètes que la version Fortran. D'un point de vue fonctionnel, LAPACK est la réunion de LINPACK et de EISPACK à laquelle ont été rajoutées des méthodes numériques nouvelles des années 70. On trouve aussi des versions par bloc d'algorithmes classiques, c'est à dire des versions orientées vers les machines possédant une hiérarchie mémoire, comme les machines à processeurs vectoriels. Les BLAS sont aussi présentes dans les versions 1, 2 et 3. Elles ont été repensées et spécifiées de manière très rigoureuse pour définir quelles étaient les opérations à fournir, quels devaient être leurs paramètres, leur noms etc. Les BLAS 1 concernent les opérations entre vecteurs. Les BLAS 2 font intervenir des matrices et des vecteurs. Les BLAS 3 concernent les opérations entre matrices. ARPACK est apparue plus récemment. Elle est basée sur une version implicite d'Arnoldi introduite par Sorensen de l'université de Rice. Elle utilise aussi les BLAS 1, 2 et 3 et LAPACK. ARPACK et LAPACK sont diffusées actuellement en version parallèle. Nous discutons plus en détails de plusieurs de ces bibliothèques par la suite. Par ailleurs, il existe d'autres bibliothèques, par exemple celles développées par CRAY pour des domaines plus spécifiques.

2.4 Matrices creuses

Les matrices creuses sont très répandues dans le calcul numérique car elles proviennent souvent de maillages liés aux méthodes à éléments finis. Une des premières bibliothèques manipulant des matrices creuses a été le Yale Sparse Package diffusée dès 70. C'était une bibliothèque complète d'algèbre linéaire pour les matrices creuses. Elle imposa aussi la représentation des matrices creuses sous le format CSR (Compress Sparse Row). Sous le format CSR, tous les éléments non nuls de la matrice sont stockés dans un tableau à une dimension de taille nnz (Number of Non Zero). Un autre tableau contient les indices des colonnes des éléments non nuls pour un ordre de ligne donné. Un dernier tableau de taille n contient les indices de colonne des premiers éléments de ligne en suivant l'ordre choisi. C'est devenu un format très classique. La bibliothèque ELLPACK utilise un format particulier de stockage des matrices creuses dans un tableau à 2 dimensions dont une est compressée. Puis est apparue la bibliothèque SPARSKIT, bibliothèque scalaire pour calcul sur matrice creuse développée à la NASA et à l'université de Minneapolis par Y. SAAD. Elle utilise aussi le format CSR et surtout on trouve un ensemble d'outils de manipulations de matrices qui offrent à l'utilisateur le choix entre transformer ses matrices au format CSR avant les appels aux routines de la bibliothèque (de nombreuses routines de conversion sont fournies) ou travailler directement en format CSR en utilisant les outils de manipulation et de visualisation de matrices fournis. La bibliothèque fournit en particulier des conversions depuis COO vers CSR. Le format COO est simple : tous les éléments non nuls sont stockés avec leurs coordonnées i et j. Il est à remarquer que toutes ces bibliothèques (denses ou creuses) ont à chaque fois été introduites et développées par des numériciens très réputés dans leur domaines de recherche et ont été diffusées dans le monde entier.

2.5 Les bibliothèques vectorielles

L'utilisation efficace de processeurs vectoriels demande le maintien du remplissage des différents niveaux de hiérarchie mémoire avec les opérandes successifs (souvent les éléments de vecteurs) pour ne pas interrompre le calcul pipeliné par une lecture en mémoire centrale coûteuse. Les solutions algorithmiques pour la vectorisation proposent une vision du calcul par blocs de données (vecteurs, matrices) et une optimisation des itérations dépendante de l'architecture (cache, profondeur du pipeline).

La première bibliothèque à avoir été vectorisée fut LINPACK, basée sur les BLAS qui optimisaient l'utilisation des unités vectorielles. L'ensemble de la bibliothèque étant basé sur les BLAS, quand celles-ci sont "vectorisées", on obtient la puissance vectorielle de la machine. Les BLAS vectorielles ont souvent été optimisées par les constructeurs pour leurs machines. Ceci a permis de réutiliser les codes développés en LINPACK pour des machines scalaires et de les mettre en oeuvre sur des machines vectorielles. Ensuite sont apparues des versions où la totalité de certains algorithmes a été reprise et vectorisée. Une des premières bibliothèques complètement vectorielles et générales, c'est à dire dépassant le cadre des systèmes linéaires, a été MAGEV développée par Control Data. Elle se basait sur les primitives Q8 qui étaient un équivalent des BLAS micro-codées pour les machines de Control Data. On trouvait en particulier des calculs de valeurs propres pour des matrices symétriques et l'ensemble des routines pour systèmes linéaires. Parallèlement, la bibliothèque EIVPACK [2] développée à Control Data France traitait les problèmes de valeurs propres non Hermitiennes. On voit apparaître à cette occasion les premières méthodes de KRYLOV, telle que la méthode d'Arnoldi pour résoudre des problèmes de valeurs propres et aussi une des premières versions de la méthode QR pour les valeurs propres en vectoriel. Pour le cas creux en général, on avait souvent recourt aux Gather/Scatter qui permettaient de gérer les indirections liées aux systèmes creux. Certaines machines, telles certains CRAY X-MP, possédaient des accélérateurs matériels spécialement développés à cet effet.

2.6 Les premières bibliothèques parallèles

L'utilisation efficace d'ordinateurs multiprocesseurs demande le maintien d'une certaine localité des données pour ne pas avoir besoin de trop d'accès distants très couteux et une bonne répartition du travail entre les processeurs afin d'équilibrer la charge de calcul et de pouvoir masquer le coût des accès aux données distantes. Les solutions algorithmiques proposent en général une vision par blocs des calculs et une structuration en phases des programmes. L'optimisation est fonction des performances de l'architecture et du nombre de processeurs. Ces restructurations entrainent quelquefois le choix de nouvelles méthodes numériques démontrées plus efficaces en parallèle.

Actuellement, les bibliothèques utilisent un modèle de programmation parallèle de type SPMD[2] (rares sont celles exploitant les capacités MIMD[3] des machines parallèles). La parallélisation d'une bibliothèque de procédures de calcul pour une architecture à mémoire distribuée revient donc à paralléliser chacune des fonctions de cette bibliothèque. Les problèmes qui se posent alors sont la détermination de l'algorithmique parallèle et de la répartition des données qui optimiseront la localité des accès aux données par les processeurs.

Les problèmes numériques denses ont été très largement étudiés dans la littérature comme cibles de départ de l'algorithmique parallèle. Le benchmark LINPACK de résolution de systèmes 100x100 et 1000x1000 est d'ailleur une référence de comparaison des ordinateurs.

La plus populaire des bibliothèques parallèles est ScaLAPACK (Scalable LAPACK) développée par l'équipe de J. Dongarra à ORNL et UTK. Elle est issue d'un projet de la DARPA de réalisation d'une bibliothèque *scalable* pour l'algèbre linéaire dense ou creuse pour les machines massivement parallèles qui regroupe aussi PARPAK, PARPRE et CAPSS et des "drivers" pour la résolution de problèmes plus complexes. Les BLAS parallèles sont basées pour les transferts de données sur les BLACS et des interfaces de communication standard telles PVM ou MPI. De nombreux constructeurs ont ensuite optimisé les implémentations de ScaLAPACK en utilisant leur propres bibliothèques de communication de messages (CMMD pour Thinking Machine, NX pour Intel).

La bibliothèque CMSSL est une des rares bibliothèques data parallèle. Elle est due à Thinking Machine et on y trouve une méthode d'Arnoldi implicite ainsi que des calculs sur matrices creuses utilisant le format COO data parallèle. Il est à noter que ce format permet d'exprimer très simplement la matrice mais ne fournit pas les meilleures performances pures malgré l'optimisation des communications à la compilation. Cette bibliothèque est la base de TMSSL projet de bibliothèque lié au projet GlobalWork, repris depuis peu par SUN.

La bibliothèque PETSc est actuellement développée à Argone National Laboratory par l'équipe de Gropp et Smith. C'est plus qu'une bibliothèque dans le sens usuel du terme: elle propose des outils de visualisation et une approche orientée objet très poussée a été utilisée lors de la conception. Elle est basée sur BLAS, LAPACK et MPI et Elle fournit des méthodes pour les équations

2. Simple Program Multiple Data
3. Multiple Instruction Multiple Data

aux dérivées partielles, un *profiler* qui permet d'avoir un profil très précis des calculs qui ont été faits. Des *drivers*[4] sont aussi fournis.

La compagnie Scientific Computing Associate (SCA), basée à New Haven aux USA, propose un package nommé PCGPACK de gradient conjugué préconditionné parallèle. Elle est représentative de nombreuses autres sociétés qui développent des packages spécialisés et qui proposent des services de suivi et de maintenance.

La bibliothèque P-SPARLIB est la nouvelle bibliothèque en cours de développement à l'université de Minneapolis par l'équipe de SAAD. C'est une des bibliothèques les plus complètes actuellement pour les calculs manipulant des matrices creuses . Citons aussi le projet TEMPLATE de l'équipe ScaLAPACK et la bibliothèque ITPACK de KINCAID ET ALS..

La bibliothèque NAG parallèle est basée sur ScaLAPACK et propose des extensions pour le calcul sur matrices creuses. D'autres bibliothèques sont disponibles de part le monde, telle PUL développée à Edinburgh, PCG++ à Austin ou encore PIM à Rio do sud et l'université du Kent et la bibliothèque du Sandia National Laboratory AZTEC qui propose plusieurs méthodes itératives performantes. Des sociétés comme IBM et CRAY proposent leur propres implémentations de ScaLAPACK basées sur leurs bibliothèques de bas niveaux. Pour finir, notons l'existence de la bibliothèque de partitionnement de graphes et de matrices creuses METIS développée à Minneapolis par l'équipe de KARYPIS.

3 Les bibliothèques BLAS, BLACS, PBLAS, LAPACK et ScaLAPACK pour le calcul avec des matrices denses

3.1 La structure interne de la bibliothèque ScaLAPACK pour matrices denses

Le plus grand succès de bibliothèque séquentielle est celui des bibliothèques BLAS, pour les noyaux de calculs, et LAPACK, pour les méthodes directes de résolution de systèmes linéaires, le calcul de valeurs/vecteurs propres et la décomposition en valeurs singulières.

Leur succès provient de la disponibilité du code source (domaine public) et donc de leur portabilité (FORTRAN 77) ainsi que de leur structuration hiérarchique (LAPACK utilise au maximum les BLAS). Leur évolution continue a suivi les meilleures méthodes algorithmiques. De plus, leur efficacité sur la plupart des machines du marché peut être obtenue par la programmation assembleur des fonctions les plus utilisées des BLAS qui a immédiatement rejaillit sur LAPACK.

Une structure hiérarchique du même type a été définie pour la version parallèle avec l'introduction d'une bibliothèque de fonctions de communications (BLACS). La hiérarchie logicielle globale obtenue est décrite dans la Figure 1.

4. Une procédure comprenant une successions d'appels à des routines d'une bibliothèque qui permettent de résoudre un problème précis.

Le mnémonique des fonctions (issu des conventions utilisées en FORTRAN),

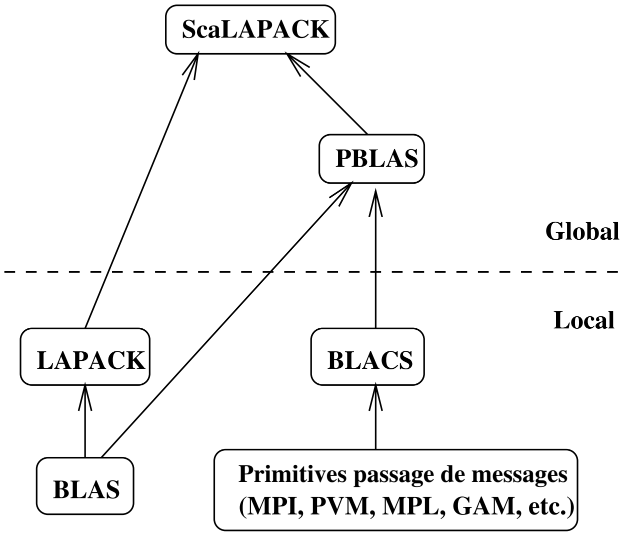

FIG. 1 - *Hiérarchie des bibliothèque d'algèbre linéaire BLAS et LAPACK en séquentiel, PBLAS et ScaLAPACK en parallèle.*

permet de déterminer, le type et la structure des données (précision S,D,I,C,Z; variables A,B,C pour les scalaires, X,Y pour les vecteurs) et l'opération réalisée. Les bibliothèque BLAS et LAPACK fonctionnent avec des matrices denses, trapézoidales, symétriques (quelquefois des matrices bandes), transposées ou non.

3.2 Les BLAS

Les BLAS (Basic Linear Algebra Subroutines) implémentent les noyaux de base de l'algèbre linéaire selon trois niveaux correspondant aux structures de données passées en paramètre. Les procédures ont été développées à l'origine en FORTRAN 77 et peuvent être appelées depuis un programme C ou FORTRAN.

<u>Les BLAS de Niveau 1</u>

Ces procédures[5] implémentent les opérations de base sur les vecteurs.

Par exemple: la mise à jour d'un vecteur par un autre vecteur (_AXPY): $y \leftarrow y + \alpha x$, et le produit scalaire (_DOT): $s \leftarrow x^T y$, (où x et y sont des vecteurs et α un scalaire).

Le nombre d'opérations flottantes effectuées est $O(2n)$ pour $O(3n)$ accès mémoires aux éléments.

<u>Les BLAS de Niveau 2</u>

5. définies à l'origine par Lawson, Hanson, Kincaid et Krogh

Elles ont été définies[6] pour implémenter des opérations entre matrices et vecteurs.

Par exemple: le produit matrice-vecteur (_GEMV) : $y \leftarrow \beta y + \alpha Ax$, (où A est une matrice, β un scalaires).

Le nombre d'opérations flottantes effectuées est $O(2n^2)$ pour $O(n^2)$ accès mémoires aux éléments. Le rapport $\frac{nombred'oprations}{accsmmoires}$ passe de 2/3 à 2 ce qui permet en augmentant le nombre d'opérations par accès mémoire d'augmenter les performances en tirant meilleur parti des architectures avec hiérarchie de mémoires et unités pipelines.

<u>Les BLAS de Niveau 3</u>

Elles ont été définies[7] pour les opérations de type matrice-matrice.

Par exemple: le produit de matrices (_GEMM) : $C \leftarrow \beta C + \alpha AB$, (où B et C sont des matrices); la mise à jour de rang k (_SYRK) $C \leftarrow \alpha AA^T + \beta C$; la résolution de systèmes triangulaires (_TRSM) $C \leftarrow \alpha T^{-1}B$.

Le rapport $\frac{oprations}{accs}$ de $n/2$ permet une très bonne réutilisation des données présentes dans les niveaux de caches du processeur.

3.3 LAPACK

LAPACK (Linear Algebra PACKage) est une bibliothèque contenant l'implémentation des problèmes principaux d'algèbre linéaire : résolution de systèmes linéaires, solution des moindres carrés linéaires, des problèmes de valeurs propres ainsi que des problèmes de valeurs singulières. Des factorisations de matrices de type : LU, Cholesky, QR, SVD, Schur et Schur généralisée sont également fournies ainsi que des calculs de réordonnancement de factorisations de Schur et d'estimations du conditionnement.

LAPACK utilise de manière intensive les algorithmes par blocs pour utiliser au mieux les hiérarchies de mémoires grâce aux opérations BLAS de niveau 3.

3.4 Distribution de Données cyclique par blocs

Le choix de la distribution des données dans les mémoires locales des processeurs agit sur deux points cruciaux pour la performance d'une application parallèle:

- Elle agit sur l' équilibrage de la charge des divers processeurs travaillant de concert. La distribution des données induit le nombre d'opérations flottantes à effectuer et les communications nécessaires entre chaque processeur.

- Elle agit sur le surcoût dû aux communications et synchronisations de l'algorithme utilisé. La répartition des données induit les schémas de communications et les attentes/synchronisations lors de l'exécution d'opérations liées par une dépendance temporelle.

6. par Dongarra, Du Croz, Hammarling et Hanson
7. par Dongarra, Du Croz, Duff et Hammarling

Pour déterminer une répartition des données performante et simple d'utilisation, le choix de l'architecture virtuelle de la machine cible est important. La "vision" en anneau ou en grille rebouclée (anneaux dans les deux dimensions), très étudiée dans la littérature, permet une bonne adéquation entre les données matricielles et un découpage intuitif des calculs.

Les deux distributions évidentes de matrices sur une grille de processeurs sont la décomposition par blocs et la décomposition cyclique (voir Figure 2 (a) et (b)). Malheureusement, ces deux cas simples ne donnent pas de bon résultats en ce qui concerne la répartition de la charge pour le cas par blocs (par exemple lors d'un parcourt diagonale de la matrice, certains processeurs ont terminé très rapidement) ou en ce qui concernent les surcoûts de communication pour le cas cyclique (beaucoup de communications nécessaires). La distribution par blocs cycliques permet de trouver le meilleur compromis entre les deux cas extrêmes ci-dessus tout en conservant la lisibilité et la facilite algorithmique du placement d'une matrice de données sur une grille de processeurs [5, 8, 9].

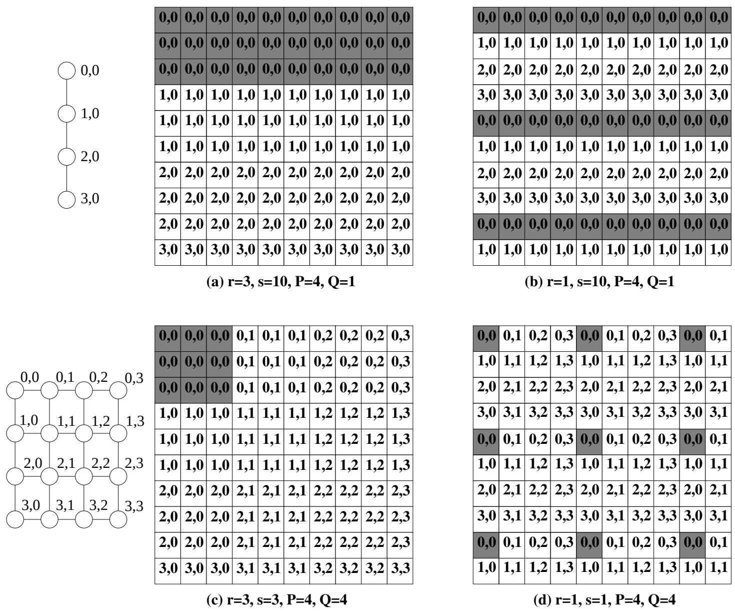

FIG. 2 - *Différentes allocations cycliques (par blocs de taille $r \times s$) d'une matrice 10×10 sur une grille de $P \times Q$ processeurs (dessinée sur la gauche). Les numéros dans la matrice représentent les indices des processeurs où sont rangés les éléments.*

Cette distribution considère un bloc rectangulaire de données comme la base sur laquelle on applique la répartition cyclique (voir figure 2 (c) et (d)). Le nombre d'éléments par processeurs est donc équilibré et la charge en calcul dans le temps ainsi que les communications peuvent être réglés en jouant sur la taille des blocs de données ou la forme de la grille de processeurs [4].

Pour les PBLAS ou ScaLAPACK, les informations concernant la distribution bloc cyclique d'une matrice sont contenues dans un descripteur [8]. Avec les langages data-parallèles de type High Performance Fortran ou Fortran-D, la distribution bloc cyclique des données est spécifiée par l'utilisateur au moment de la déclaration des tableaux.

3.5 (Re)distribution des données

La distribution initiale d'une matrice stockée sur un processeur en morceaux stockés sur P processeurs au format bloc cyclique est un problème non trivial, de même que le passage d'une distribution bloc cyclique à une autre ou le retour des résultats sur un processeur. De plus, le changement du nombre de processeurs ou du format des grilles virtuelles, qui peut permettre des gains significatifs en temps d'exécution est aussi un problème algorithmique si l'on veut faire ces transferts de manière performante.

Une collection de fonctions a donc été mise au point pour simplifier le travail du programmeur (_PGEMR2D) [7].

3.6 BLACS

Les BLACS (Basic Linear Algebra Communication Subroutines) sont des procédures pour les opérations de communication de données matricielles dans un réseau représenté sous forme d'une grille virtuelle rectangulaire [6].

Les procédures principales sont `SD` (envoyer un message), `RV` (recevoir), `BS` et `BR` (diffuser) et des opérateurs globaux comme `MAX` (maximum), `MIN` (minimum) et `SUM` (sommation).

La notion de contexte permet d'isoler les communications de la bibliothèque des communications extérieures. Les structures de données communiquées sont des matrices correspondant aux BLAS.

Ces procédures sont disponibles pour toutes les machines parallèles à mémoire distribuée et au dessus du système PVM.

3.7 PBLAS

Les PBLAS sont la version parallèle des BLAS. Elles sont bâties sur les BLAS séquentielles pour les calculs et sur les BLACS pour les communications. De plus, les PBLAS contiennent une procédure explicite de transposition de matrice ($C \leftarrow \beta C + \alpha A^T$).

Les interfaces des procédures sont très proches de celles des BLAS séquentielles avec l'ajout des descripteurs de l'allocation des matrices.Par exemple pour le produit de matrices:

---

8. Le descripteur contient la taille de la matrice, la taille d'un bloc élémentaire, les coordonnées du processeur possédant dans sa mémoire l'élément du coin supérieur gauche de la matrice, le contexte BLACS de définition et la dimension principale locale.

```
CALL   DGEMM('No transpose', 'No transpose', M, N, K,
       ALPHA, A, LDA, BETA, B, LDB, C, LDC)

CALL   PDGEMM('No transpose', 'No transpose', M, N, K,
       ALPHA, A, IA, JA, DESCA, BETA, B , IB, JB, DESCB,
       C, IC, JC, DESCC)
```

3.8 ScaLAPACK

ScaLAPACK (Scalable LAPACK) est la bibliothèque LAPACK pour machines parallèles à mémoire distribuée. Celle-ci est basée sur les BLAS, les PBLAS, les BLAS et LAPACK.

Les buts principaux lors la création de ScaLAPACK pour les machines parallèles ont été:

- la scalabilité, pour garantir des performances accrues lorsque on augmente le nombre de processeurs pour l'exécution des procédures,
- une stratégie identique à celle de LAPACK (performance, portabilité, interface quasi-identique, ...)

Actuellement, ScaLAPACK contient un sous-ensemble des procédures de LAPACK: factorisations LU, LL^t, QR, moindres carrés, calculs de valeurs propres.

4 Des Bibliothèques vers les *boites à outils (toolkits)* pour le calcul scientifique parallèle

4.1 La bibliothèque PARPACK (Parallel ARnoldi PACKage)

Cette sous partie de ScaLAPACK correspond à la version parallèle de ARPACK vue précédemment. Elle implémente la méthode d'Arnoldi implicite introduite par SORENSEN, en Fortran77 et en utilisant MPI pour les communications. Cette méthode (et de nombreuses autres, avec l'apparition des méthodes de KRYLOV dans les bibliothèques) comporte plusieurs phases de calcul, dont la plus coûteuse est la phase dite de projection. Les calculs de Ritz, qui sont une autre phase de la méthode, sont faits de manière redondante sur les processeurs en utilisant LAPACK.

4.2 La bibliothèque P-SPARSLIB

Quand on a des matrices creuses en algèbre linéaire on a souvent recourt à des méthodes itératives et donc le produit matrice-vecteur est très utilisé. Dans la bibliothèque P-SPARSLIB, la représentation de matrices creuses utilisée est optimisée pour les produits matrice-vecteur. Tous les algorithmes utilisés reposent sur des méthodes de KRYLOV itératives. Après distribution sur plusieurs processeurs de parties d'une matrice creuse, localement les formats utilisés sont de type CSR, Block Sparse Row ou dense qui permettent aux phases principales d'être réalisées en parallèles.

De nombreuses méthodes itératives préconditionnées basées sur les sous-espaces de Krylov, telle la méthode FGMRES ou des gradients conjugués, sont proposées dans cette bibliothèque.

4.3 La boite à outils PETSc

PETSc est une bibliothèque orientée objet basée sur les BLAS, LAPACK et MPI (comme ScaLAPACK++). La Figure 3 illustre la hiérarchie des niveaux d'abstraction de la bibliothèque. Au centre de la bibliothèque se trouve un module dédié aux méthodes de sous espaces de Krylov. Ces méthodes sont de plus en plus utilisées car elles sont très bien adaptées aux systèmes linéaires et au calcul de valeurs propres non symétriques pour de grosses matrices. C'est une bibliothèque qui permet par exemple à un utilisateur d'encapsuler son propre solveur à l'intérieur de la bibliothèque. En contrepartie, c'est une boite à outils relativement lourde à installer.

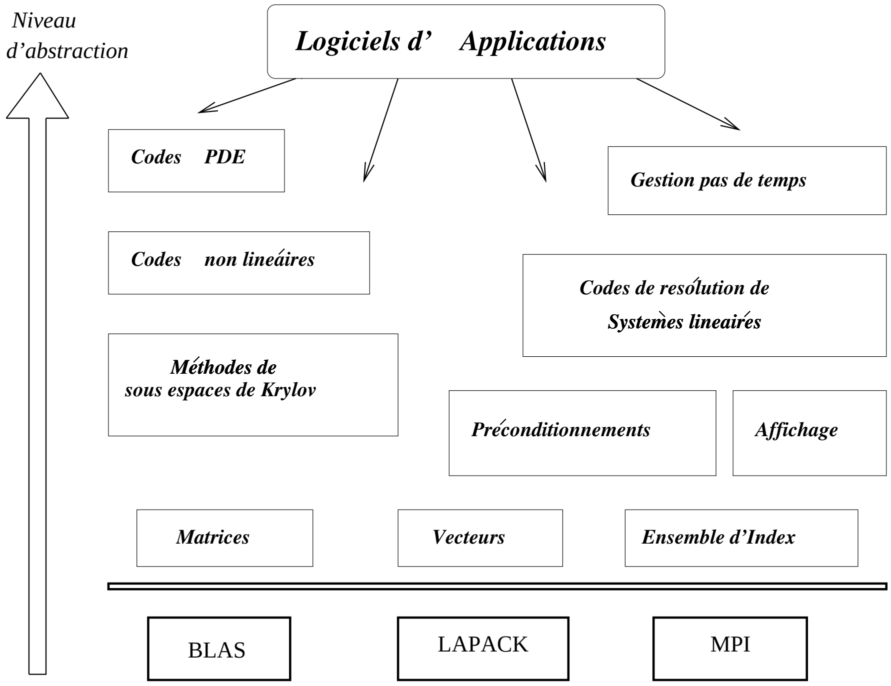

Fig. 3 - *Structure de la bibliothèque PETSc*

Notons que certains paramètres des méthodes numériques itératives sont très difficiles à ajuster car l'utilisateur doit très bien connaître les propriétés numériques et parallèles des méthodes présentes dans les bibliothèques. Par exemple, les méthodes de type Arnoldi et GMRES possèdent trois parties; deux sont parallèles (la projection et le redémarrage) et une est séquentielle (calcul de Ritz). La taille m du sous-espace de Krylov est très délicate à choisir : plus elle est grande et plus la convergence est rapide mais plus la partie séquentielle augmente [3]. Le choix des différents paramètres de ces méthodes demande donc une expertise qui n'est pas encore disponible dans ces bibliothèques.

5 Conclusion et perspectives

Il existe plusieurs bibliothèques parallèles pour les applications bien conditionnées et de tailles entrant entièrement dans la mémoire globale de la machine parallèle. Elles sont pour la plupart basées sur ScaLAPACK et les BLAS (optimisées par les constructeurs en fonction des processeurs) pour les calculs et les bibliothèques PVM ou MPI pour l'échange de données entre les processeurs.

Des bibliothèques considèrent le cas des matrices creuses en ayant recourt principalement aux méthodes de KRYLOV.

Aucune bibliothèque ne propose encore la prise en compte des problèmes qui ne peuvent pas entrer entièrement dans la mémoire de la machine (*out of core*) même si des développements sont en cours. Or de plus en plus d'applications ont besoin de gérer des données énormes qui ne tiennent pas sur les mémoires centrales des ordinateurs parallèles les plus grands.

Certaines nouvelles machines parallèles présentent des processeurs qui sont eux-mêmes vectoriels ou parallèles. Les bibliothèques ne savent que rarement gérer cette possibilité et les systèmes d'exploitation n'en permettent pas une utilisation optimale.

Nous pensons que l'avenir est aux ateliers de programmation scientifique. Ils devront fournir à la fois des bibliothèques performantes, avec probablement une approche orientée objet, la possibilité de régler certains paramètres en fonction des données (maillages adaptatifs, optimisations), des fonctions de gestion de l'*out of core*, fournir des outils de visualisation [10], et des systèmes experts efficaces pour aider les utilisateurs dans le choix des paramètres et des méthodes. Les bibliothèques parallèles classiques seront alors incluses dans ces *toolkits* pour le calcul scientifique parallèle.

Principales pages Web sur le sujet:

- BLAS, LAPACK, BLACS, PBLAS et ScaLAPACK : `http://www.netlib.org/scalapack`
- Nombreuses informations et logiciels sur `http://www.netlib.org`
- PETCS : `http://www.mcs.anl.gov/petsc/petsc.html`
- P-SPARLIB : `http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html`
- Logiciels domaine public pour le calcul haute performance : `http://www.nhse.org`
- Site miroir : `http://www.hensa.ac.uk/parallel`

Remerciements

Nous remercions G. Alleon, F. Desprez, A. Petitet et Y. Saad pour leurs commentaires et leurs autorisations pour la reproduction de certain documents.

Références

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, et D. Sorensen, *LAPACK Users' Guide*, Second Edition, *SIAM, Philadelphie, PA.*, 1995.

[2] N. Emad, *Contribution à la résolution de grands problèmes de valeur propre*, thèse de l'université Pierre et Marie Curie, Paris 6, France, 1989.

[3] S.Petiton et G.Edjlali, *Krylov Subspace Methods on Multi-Connection Machines Environnement*, Science on the Connection Machine System, éditeurs: Jean-Michel Alimi, Arturo Serna et Hans Scholl, Octobre 1995.

[4] F. Desprez, J.J. Dongarra, et B. Tourancheau. Performance Complexity of LU Factorization with Efficient Pipelining and Overlap on a Multiprocessor. *Parallel Processing Letters*, 5-II, 1995.

[5] E. Anderson, A. Benzoni, J. Dongarra, S. Moulton, S. Ostrouchov, B. Tourancheau, et R. Van de geijn. LAPACK for distributed memory architecture. *Fifth SIAM Conference on Parallel Processing for Scientific Computing*, USA, 1991.

[6] E. Anderson, A. Benzoni, J. Dongarra, S. Moulton, S. Ostrouchov, B. Tourancheau, et R. Van de geijn. BLACS: Basic linear algebra communication subroutines. *Sixth Distributed Memory Computing Conference*, Portland, USA, 1991.

[7] L. Prylli, et B. Tourancheau. Efficient Block Cyclic Data Redistribution. *Journal of Parallel and Distributed Computing*, (à paraitre), 1997.

[8] F. Desprez. *Procédures de base pour le calcul scientifique sur machines parallèles à mémoire distribuée.* Thèse de l'INP-Grenoble, France, 1994.

[9] M. Pourzandi. *Étude de l'impact des recouvrements calcul-communication sur des algorithmes parallèles de calcul matriciel.* Thèse de l'UCB-Lyon, France, 1995.

[10] X. Vigouroux. *Analyse distribuée de traces d'exécution de programmes parallèles.* Thèse de l'Ecole Normale Supérieure de Lyon, France, 1996.