Bibliothèques mathématiques

Informatique pour le calcul scientifique : bases et outils

Jérémie Gaidamour
gaidamour@idris.fr

24 septembre 2014

Plan

Motivation

Une bibliothèque (ou *library* en anglais) logicielle est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

Pourquoi utiliser une bibliothèque logicielle ?

- Productivité :
  - Réduire le temps de développement.
  - Faciliter la maintenance, la lisibilité et la portabilité du code.
- Performances et fonctionnalités :
  - Bénéficier de l'expertise des auteurs de la bibliothèque.
  - De nouvelles fonctionnalités lors des mises à jour.
- Communauté utilisateurs et support :
  - Mutualisation des moyens, stabilité, correction des bugs.

Motivation

Risques liés à l'utilisation de bibliothèques :

- Dépendances à des projets tiers :
  - Pas l'expertise pour corriger des problèmes.
  - Régressions, interfaces ou fonctionnalités changeantes.
  - Disponibilité sur les plateformes cibles.
  - Abandon possible du projet.
- Complexité de l'installation.
- Problématique des licences logicielles.

Les bibliothèques mathématiques

Besoins récurrents en simulation numérique, modélisation ou analyse numérique :

- Algèbre Linéaire
- Résolution de systèmes linéaires
- Calcul de valeurs et vecteurs propres
- Transformée de Fourier rapide (FFT)
- Résolution d'équations différentielles
- Système d'équations non linéaires
- Discrétisation et maillages
- Renumérotation et partitionnement de graphes
- Problèmes d'optimisation
- Statistiques
- Calcul arithmétique en précision arbitraire
- ...

Autres bibliothèques logicielles utilisées en calcul scientifique :

- Parallélisme et communication (MPI, threads)
- Gestion des Entrées/Sorties
- Visualisation de résultats
- Structures de données
- Mesure de temps,
- Générateurs de nombres aléatoires
- ...

Ressources

Quelques sites recensant les bibliothèques mathématiques :

- Projet PLUME `www.projet-plume.org/logiciels_maths`
- The Guide To Avail. Mathematical Software `http://gams.nist.gov`
- Wikipedia `http://en.wikipedia.org/wiki/Mathematical_software`
- Arnold Neumaier `www.mat.univie.ac.at/~neum/software.html`
- Antoine Le Hyaric `www.ann.jussieu.fr/~lehyaric/freesoft/`
- Florian De Vuyst `www.twiki.org/cgi-bin/view/Main/FlorianDeVuyst`

Objectifs de ce cours

Impossible d'être exhaustif ou de couvrir tous les domaines . . .

Objectifs

- Faire un tour d'horizon des principales bibliothèques scientifiques.
- Comprendre leurs intérêts : fonctionnalités, performances, . . .
- Savoir choisir, installer et utiliser quelques bibliothèques.

Ma présentation s'inspire de celle donnée par Violaine Louvet :

http://calcul.math.cnrs.fr/spip.php?article225

Plan

Algèbre linéaire dense

BLAS : **B**asic **L**inear **A**lgebra **S**ubroutines (1979)

Ensemble de routines *standardisées* réalisant des opérations de base de l'algèbre linéaire comme des additions de vecteurs ou des multiplications de matrices.

Routines classées en fonction du ratio flops/nb. d'op. mémoire

- Niv. 1 : opérations sur des vecteurs.
  $$y = y + \alpha x, \text{ ratio en : } 2n/3n \Rightarrow 2/3$$
- Niv. 2 : opérations matrices/vecteurs.
  $$y = y + Ax, \text{ ratio en } 2n^2/n^2 \Rightarrow 2$$
- Niv. 3 : opérations matrices/matrices.
  $$C = C + AB, \text{ ratio en } 2n^3/4n^2 \Rightarrow n/2$$

Privilégier l'écriture des algorithmes avec des BLAS de niveau 3.

Support des réels et des complexes, en simple et double précision.

Documentation : `www.netlib.org/blas/`

Algèbre linéaire dense

Interface Fortran native : www.netlib.org/blas/blasqr.pdf

Interface C standard via CBLAS (<cblas.h>).

Exemple : $y = \alpha Ax + \beta y$ ou $y = \alpha A^T x + \beta * y$

```
SUBROUTINE DGEMV(TRANS, M, N,
                 ALPHA, A, LDA,
                        X, INCX,
                 BETA,  Y, INCY)
```

Implémentations disponibles

- Implémentation de *référence* (= **non optimisée**) en Fortran 77.
- Implémentations libres (BSD) : ATLAS, OpenBLAS (= GotoBlas)
- Aussi disponible dans les bibliothèques scientifiques constructeurs : MKL (Intel), ACML (AMD), ESSL (IBM), ...

Parallélisme

- L'implémentation de référence des BLAS est *séquentielle*.
- Les implémentations ATLAS, OpenBLAS, MKL (entre autres) sont multithreadées.
- Il faut compiler les bibliothèques avec le support des threads et faire l'édition de liens avec les bonnes bibliothèques (ex : `https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor`).
- Attention au nombre de threads lorsque les applications sont déjà multithreadées !

Variables d'environnements

- ATLAS : définition du nombre de threads max. à la compilation.
- OpenBlas : OPENBLAS_NUM_THREADS.
- MKL (Intel) : MKL_NUM_THREADS.

Algèbre linéaire dense

LAPACK : **L**inear **A**lgebra **PACK**age

Résolution de systèmes d'**équations linéaires**, **moindres carrés** linéraires, problèmes de **valeurs propres**, décomposition en **valeurs singulières**.

Fournit aussi les opérations de **factorisation de matrices** sous-jacentes à ces problèmes : LU, Cholesky, QR, SVD, Schur.

Support des matrices pleines et bandes ($\mathbb{R}$, $\mathbb{C}$, simple et double précision).

- LAPACK repose sur les BLAS (algorithmes écrits par blocs).
- LAPACK est utilisé par de nombreux logiciels de calcul scientifique : MATLAB, Scilab, SciPy, ...

Documentation

www.netlib.org/lapack

www.netlib.org/lapack/lapackqref.ps

www.phys.ens.fr/~hare/MP025/guidelapack-2012-03-06.pdf

Interface de LAPACK

- Implémentation de référence écrite en Fortran 90. Interface Fortran 95 aussi disponible (lapack95).
- L'interface C s'appelle LAPACKE (en cours de standardisation). Attention, CLAPACK n'est que la conversion automatique (f2c) du code Fortran de référence en C.

  http://www.netlib.org/lapack/lapacke.html

Convention de nommages des routines BLAS/LAPACK

Nom des routines en 5 ou 6 caractères : X*YY*ZZZ

- X = **S** (réel), **D** (double), **C** (complexe), **Z** (complexe double).
- YY = type de matrices (ex : **DI** : Diagonal ; **GE** : Générale, ...)
- Les caractères restants décrivent le type d'opération.

Routines LAPACK

SV(X)	Équations linéaires
LS, LSY , LSS, LSD	Moindres carrés
LSE, GLM	Moindres carrés généralisés
EV(X), EVD, EVR	Éléments propres, cas symétrique
ES(X), EV(X)	Éléments propres, cas non-symétrique (YY=GE)
SVD, SDD	Décomposition en valeurs singulières (YY=GE)
GV(X), GVD	Éléments propres, cas symétrique défini, généralisé
ES(X), EV(X)	Éléments propres, cas non-symétrique, généralisé (YY=GG)
GV(X), GVD, ES(X), EV(X), SVD	Décomposition en valeurs singulières, généralisé

Exemple :

```
SUBROUTINE DGEEV(JOBVL, JOBVR, N, A, LDA, WR, WI, VL, LDVL, VR,
                 LDVR, WORK, LWORK, INFO)
```

En Fortran, les éléments d'une matrices 2D sont stockés en mémoire colonne par colonne (Column-Major Order).

Interfaces C BLAS/LAPACK

- En C, les matrices peuvent être stockées dans des tableaux 1D selon la convention Fortran.
- L'interface C permet aussi d'utiliser la convention Row-Major (cf. argument supplémentaire de type `enum CBLAS_ORDER`).
- Les fonctions de l'interface C sont préfixées par `cblas_` et `LAPACKE_`.

Note : Il est aussi possible d'appeler directement les routines Fortran en C, mais cela n'est pas recommandé.

Structures matricielles BLAS/LAPACK :

- Pour les matrices non symétriques : tableau Fortran 2D.
- Pour les matrices symétriques, hermitiennes ou triangulaires :
  - Format complet : tableau Fortran 2D dont on ignore la moitié des éléments.

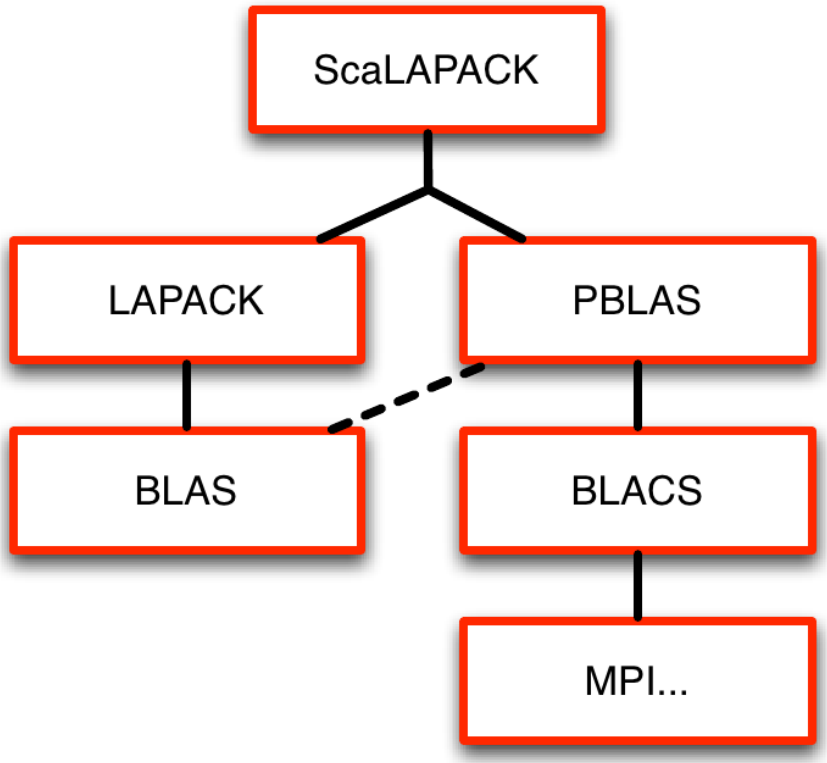
$$\texttt{UPLO = 'U'} : A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{pmatrix}$$

  - Format compact (*packed*) dans un tableau 1D de taille $n(n+1)/2$: $A = (a_{11}, a_{12}, a_{22}, a_{13}, a_{23}, a_{33}, a_{14}, a_{24}, a_{34}, a_{44})$.
- Pour les matrices bandes, stockage diagonale par diagonale.

ScaLAPACK

ScaLAPACK : **Sca**lable **L**inear **A**lgebra **PACK**age

C'est une version distribuée de LAPACK : www.netlib.org/scalapack/.



- BLACS : **B**asic **L**inear **A**lgebra **C**ommunication **S**ubprograms.
- PBLAS : **P**arallel **BLAS**.

Distribution **bloc-cyclique 2D** des matrices :

Matrix 9x9 partitionnée en blocs 2x2 et distribuée sur une grille de 2x3 processeurs :

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

(0,0)

1,1	1,2	1,7	1,8
2,1	2,2	2,7	2,8
5,1	5,2	5,7	5,8
6,1	6,2	6,7	6,8
9,1	9,2	9,7	9,8

(0,1)

1,3	1,4	1,9
2,3	2,4	2,9
5,3	5,4	5,9
6,3	6,4	6,9
9,3	9,4	9,9

(0,2)

1,5	1,6
2,5	2,6
5,5	5,6
6,5	6,6
9,5	9,6

(1,0)

3,1	3,2	3,7	3,8
4,1	4,2	4,7	4,8
7,1	7,2	7,7	7,8
8,1	8,2	8,7	8,8

(1,1)

3,3	3,4	3,9
4,3	4,4	4,9
7,3	7,4	7,9
8,3	8,4	8,9

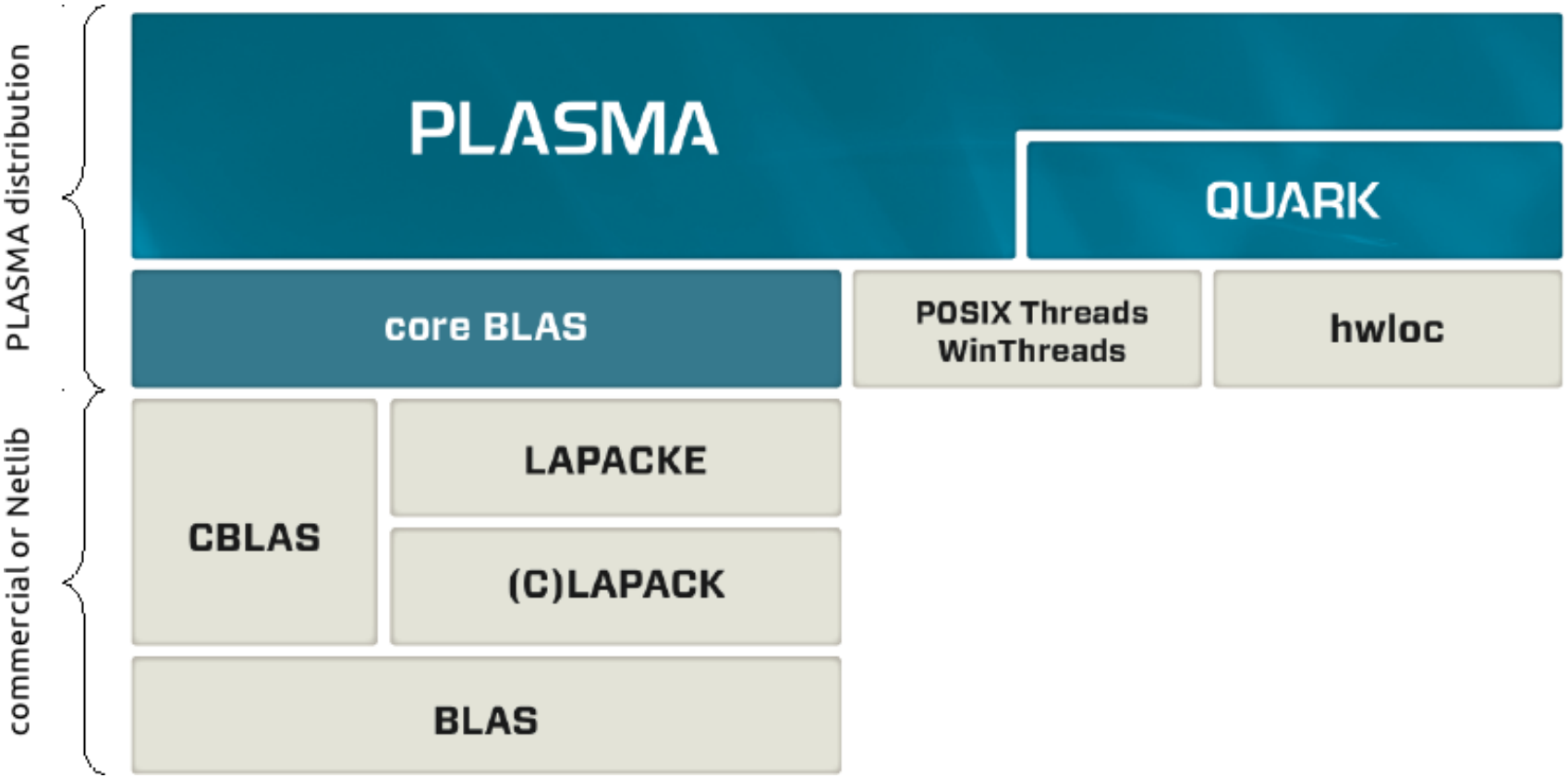
(1,2)

3,5	3,6
4,5	4,6
7,5	7,6
8,5	8,6

Voir aussi : `http://acts.nersc.gov/scalapack/hands-on/datadist.html`

PLASMA (et MAGMA)

PLASMA : **P**arallel **L**inear **A**lgebra **S**oftware for **M**ulticore **A**rchitecture.



Voir aussi :

www.training.prace-ri.eu/uploads/tx_pracetmo/plasma_magma.pdf

Algèbre linéaire creuse

Une matrice creuse est une matrice contenant beaucoup de zéros (ex : équations aux dérivées partielles . . .).

$$A = \begin{pmatrix} 1 & * & 2 & * \\ * & * & * & 3 \\ 4 & 5 & 6 & * \\ * & * & * & 7 \end{pmatrix}$$

Exemple de format de stockage :

IJV (ou COO - Coordinate list) :

```
I = [0, 0, 1, 3, 2, 2, 2]
J = [0, 2, 3, 3, 2, 1, 0]
V = [1, 2, 3, 7, 6, 5, 4]
```

CSR (Compressed Sparse Row) :

```
col_ind = [0, 2, 3, 0, 1, 2, 3]
row_ptr = [0, 2, 3, 6, 7]
val     = [1, 2, 3, 4, 5, 6, 7]
```

SPARSEKIT permet de convertir des matrices creuses en différents formats : http://people.sc.fsu.edu/~burkardt/f_src/sparsekit/sparsekit.html (Fortran 90).

Équation aux dérivées partielles

Deux gros framework parallèle dans le domaine :

- PETSc (*P*ortable, *E*xtensible *T*oolkit for *S*cientific *C*omputation) : `www.mcs.anl.gov/petsc/`
  - Développé à Argonne National Laboratory.
  - En C "objet". Très bon support Fortran. Interface Python.
  - Conçu comme une bibliothèque cohérente.
- Trilinos : `http://trilinos.org/`
  - Développé principalement à Sandia National Laboratories.
  - En C++. Interfaces Fortran 2003 et Python.
  - Organisé comme une collection de "packages".
  - Objectif général plus large (intégration verticale).

Liste de logiciels pour la résolution de problèmes d'algèbre linéaire : `www.netlib.org/utk/people/JackDongarra/la-sw.html`

PETSc n'est pas une boîte noire mais permet de tester facilement de nombreux algorithmes.

Gestion des vecteurs parallèles (classe `Vec`) :

- Chaque processus MPI est responsable d'une partie contiguë du vecteur global.

```
VecCreate(MPI_Comm, Vec*), VecDestroy(Vec *)
VecSetSizes(Vec, PetscInt n, PetscInt N), VecDuplicate
VecDot(Vec, Vec, PetscScalar *), VecNorm, VecScale, VecAXPY
VecGetArray(Vec, PetscScalar **), VecGetArrayF90
```

- Assemblage parallèle des vecteurs :
  - Il n'est pas nécessaire de créer les éléments localement.
  - Possibilité de recouvrir les communications de la phase d'assemblage par du calcul.

```
VecSetValues, VecAssemblyBegin(), VecAssemblyEnd()
```

Matrices parallèles (classe `Mat`) :

- Différents formats supportés : Denses, AIJ (= CSR), AIJ par blocs, diagonales, symétriques, matrix free ...
- Interface similaire pour tous les formats de matrices (polymorphisme).
- Penser à *préallouer* les matrices. La phase d'assemblage est similaire à celle des vecteurs.
- Matrices partitionnées entre les processus MPI par blocs de lignes. Chaque processus redécoupe sa matrice locale par bloc-colonnes pour faciliter le produit matrice-vecteur parallèle. www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/Mat/MatCreateAIJ.html
- Possibilité de contrôler le partitionnement des matrices (`MatPartitioning`, `MatOrdering`).

```
MatCreate(MPI_Comm, Mat*)
MatSetSizes, MatSetType, MatSetFromOptions
```

Résolution de systèmes linéaires :

- Méthodes de Krylov (`KSP`) et préconditionneurs (`PC`) :
  - Méthode du gradient conjugué, GMRES, ...
  - Jacobi, Méthodes de Schwartz (`PCASM`), ILU, ...
- Accès à de nombreuses bibliothèques externes via l'interface de PETSc : solveurs directs, multigrilles ...

Résolution de systèmes non linéaires : voir la classe SNES.

Tutoriels PETSc :

`www.mcs.anl.gov/research/projects/petsc/documentation/tutorials/`

Trilinos

Notion de paquets logiciels :

- Trilinos est une collection de bibliothèques logiciels (C/C++) de calcul scientifique visant les plateformes parallèles.
- Les paquets de Trilinos interagissent entre eux et partagent la même infrastructure logicielle mais ils sont sont développés de manière indépendante.
- Actuellement, co-existence de deux générations de paquets.

Fonctionnalités :

- Discrétisations de problèmes,
- Géométrie, Maillage, équilibrage de charge,
- Algèbre linéaire, solveur linéaire,
- Solveurs non-linéaire,
- Entrées-sorties parallèles,
- Interfaces, ...

Trilinos - Liste des paquets

	Objective	Package(s)
Discretizations	Meshing & Discretizations	STKMesh, Intrepid, Pamgen, Sundance, Mesquite
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Tpetra
	Interfaces	Xpetra, Thyra, Stratimikos, Piro, ...
	Load Balancing	Zoltan, Isorropia, Zoltan2
	"Skins"	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	Utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Phalanx, Trios, ...
Solvers	Iterative linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos, Amesos2, ShyLU
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi
	Incomplete factorizations	AztecOO, Ifpack, Ifpack2
	Multilevel preconditioners	ML, CLAPS, MueLu
	Block preconditioners	Meros, Teko
	Nonlinear solvers	NOX, LOCA
	Optimization	MOOCHO, Aristos, TriKota, Globipack, Optipack
	Stochastic PDEs	Stokhos

Tutoriel : www.vecpar.org/slides/TrilinosTutorialVecpar2014.pptx

Matrices et Vecteurs distribués :

- Disponible dans les paquets : **Eptra**, **Tpetra** et **Kokkos**. Voir : `Tpetra::CrsMatrix, Tpetra::MultiVector`
  `http://trilinos.org/docs/dev/packages/tpetra/doc/html/`
- La description de la distribution parallèle des données utilise le concept de `Map` (= correspondance entre indices locaux et indices globaux). Les objets peuvent être redistribués.

Les objets 2D creux et parallèles nécessitent au moins 2 `Maps` :

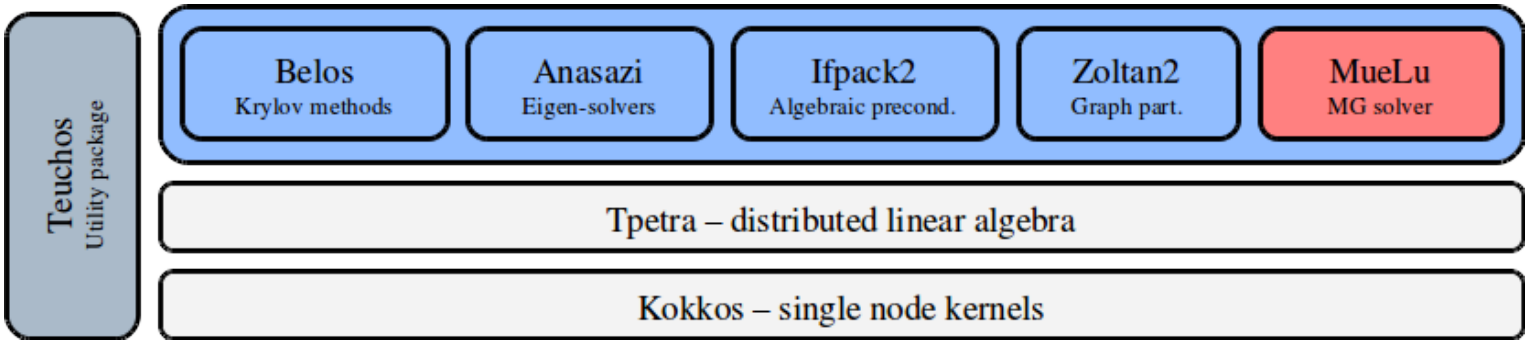
$$\begin{pmatrix} 2 & -1 & * \\ -1 & 2 & -1 \\ * & -1 & 2 \end{pmatrix}$$

$$P_0 : \begin{cases} RowMap = \{0, 1\} \\ ColMap = \{0, 1, 2\} \end{cases}$$

$$P_1 : \begin{cases} RowMap = \{2\} \\ ColMap = \{1, 2\} \end{cases}$$

Résolution de systèmes linéaires :

- Paquets : **AztecOO**, **Ifpack** (préconditionneurs) et **Belos** (Krylov), **Amesos** (méthodes directes), **ML** (multigrilles).
- Configuration : `Teuchos::ParameterList`.
- Passages d'objets entre le code utilisateur et les bibliothèques : *Smart Pointers* (gestion du transfert de propriété des objets et libération automatique des ressources).



Bibliothèques C++ d'algèbre linéaire

En C++ , de nombreuses bibliothèques proposent des interfaces pour l'algèbre linéaire :

- Eigen http://eigen.tuxfamily.org/
  - Bibliothèque C++ utilisant les templates et composée de fichiers d'entetes.
  - Classes pour les vecteurs, les matrices denses et creuses.
  - Factorisation de matrices (LU, QR, ...), solveurs itératifs, calcul de valeurs propres et accès à des bibliothèques externes.
  - Support de la vectorisation.

```
#include <Eigen/Dense>
using namespace Eigen;
int main() {
  Matrix2f A, b;
  A << 2, -1, -1, 3;  b << 1, 2, 3, 1;
  cout << "A:\n" << A << "\nb:\n" << b << endl;

  Matrix2f x = A.ldlt().solve(b);
  cout << "The solution is:\n" << x << endl;
}
```

Bibliothèques C++ d'algèbre linéaire

Voir aussi :

- Blitz++ http://blitz.sourceforge.net/
  - Bibliothèque générique de tableaux et vecteurs en C++.
  - Utilise intensivement les « Expression Templates ».
- Armadillo http://arma.sourceforge.net/
  - Bibliothèque d'algèbre linéaire en C++.

Solveurs directs

Solveur direct parallèle de systèmes linéaires creux (factorisation de matrices : LU, LDL^T, LL^T) :

- MUMPS : http://mumps.enseeiht.fr/
  - Code F90 + MPI.
  - Grande stabilité numérique. Très bon support utilisateurs.
- PaStiX : http://pastix.gforge.inria.fr/
  - En C, MPI + threads, GPU.
  - Optimisation de la consommation mémoire. Support des bibliothèques d'ordonnancement pour le multicœur.
- SuperLU : http://crd.lbl.gov/~xiaoye/SuperLU/
  - En C. 3 versions distinctes : séquentielle, MPI, pthreads.
  - Matrices non symétriques.

Voir aussi : www.cise.ufl.edu/research/sparse/codes/

Solveurs multigrilles parallèles

- ML (Trilinos)
  - en C. Interfacé avec Trilinos.
  - Algorithmes pour Poisson, Elasticité, Petrov-Galerkin, H(curl), H(div) SA-AMG, Petrov-Galerkin, ...
  - Permet d'utiliser les lisseurs et solveurs de Trilinos.
- Hypre (BoomerAMG) http://acts.nersc.gov/hypre/
  - Préconditionneurs performants dont multigrille sur grille structurée et non structurée.

Voir aussi :

https://computation-rnd.llnl.gov/linear_solvers/pubs/yang1.pdf

Partitionnement de graphes

- Scotch et PT-Scotch http://www.labri.fr/~pelegrin/scotch/
  - Partitionneur séquentiel et parallèle de graphes, de maillages et renumérotation de matrices creuses.
  - Interface de compatibilité METIS.
  - Utilisé notamment par les solveurs MUMPS et PaStiX.
- METIS et ParMETIS

  http://glaros.dtc.umn.edu/gkhome/views/metis
  - Partitionneur séquentiel et parallèle de graphes, de maillages et renumérotation de matrices creuses.
  - Aussi utilisé par de nombreux logiciels.
- Zoltan et Isorropia (Trilinos)

  trilinos.org/capability-areas/meshes-geometry-and-load-balancing/
  - Répartition de charge, partitionnement, coloration de graphes et renumérotation.
  - Méthodes géométriques, graphes et d'hypergraphes.
  - Interfaces pour utiliser les partitionneurs Scotch, PT-Scotch, METIS et ParMETIS.

Valeurs et vecteurs propres

- LAPACK, ScaLAPACK, PLASMA, MAGMA pour les matrices denses.
- ELPA http://elpa.rzg.mpg.de/
  - Solveur parallèle distribué pour les matrices denses symétriques.
  - Format de données compatible ScaLAPACK.
- ARPACK (**AR**noldi **PACK**age) : www.caam.rice.edu/software/ARPACK/
  - Calcul des valeurs propres et des vecteurs propres de grosses matrices creuses.
  - Basée sur les algorithmes itératifs de Lanczos/Arnoldi.
  - Ecrit en Fortran 77.
- SLEPc www.grycap.upv.es/slepc/
  - Implémentation parallèle de recherche de valeurs propres.
  - Décomposition en valeurs singulières.
  - Construit au dessus de PETSc.
- Anasazi (Trilinos) www.sandia.gov/~rblehou/anasazi-toms.pdf

Transformées de Fourier

- FFTW (**F**astest **F**ourier **T**ransform in the **W**est) : `www.fftw.org`
  - Permet d'effectuer des calculs de transformées de Fourier discrètes en une ou plusieurs dimensions.
  - Adapte le choix de l'algorithme aux détails du matériel sous-jacent de manière à réaliser les meilleures performances.
  - Peut utiliser des threads (pthread ou OpenMP) et MPI.
  - Ecrit en C. Interface Fortran. GPL.
- FFTPACK - FFT en Fortran 77 : `www.netlib.org/fftpack/`
- FFT aussi disponible dans la MKL, ESSL, ACML, ... mais attention à la portabilité.

Exécution simultanée par plusieurs processus légers

Attention aux problèmes de **thread safety**. Par exemple, dans FFTW, seule la fonction `fftw_execute` est thread-safe.

Voir aussi : `www.fftw.org/benchfft/ffts.html` (liste de bibliothèques et comparaisons de performances).

Equations différentielles ordinaires (ODE)

Résolution d'ODE :

- ODEPACK : https://computation.llnl.gov/casc/odepack
  - Collection de solveurs (Fortran 77) pour la résolution de systèmes différentiels ordinaires.
- GSL (**G**nu **S**cientific **L**ibrary) : www.gnu.org/software/gsl/
  - Large choix de routines mathématiques écrites en C/C++ dont des solveurs d'ODE.
- SUNDIALS (**SU**ite of **N**onlinear and **DI**fferential/**AL**gebraic equation **S**olvers) https://computation.llnl.gov/casc/sundials/
  - Résolution de systèmes d'ODE et d'ADE. Ecrit en C.

Bibliothèques généralistes et autres outils

- Méthode des éléments finis :
  - GetFEM++, bibliothèque C++ d'éléments finis.
    `http://home.gna.org/getfem/`
  - MELINA, bibliothèque Fortran d'éléments finis.
    `http://anum-maths.univ-rennes1.fr/melina/`
  - OFELI, bibliothèque C++ d'éléments finis.
    `www.ofeli.net/`
  - FreeFem++, bibliothèque C++ d'éléments finis.
    `www.freefem.org/ff++/`
- Mécanique des fluides :
  - OpenFOAM, outils pour la CFD, `www.openfoam.com`

Plan

Entrées/Sorties

- HDF5 (**H**ierarchical **D**ata **F**ormat) : www.hdfgroup.org/HDF5/
  - Format standardisé d'entrées/sorties, séquentielles et parallèles.
  - Existence d'API en C, C++ et Fortran 90.
- NetCDF : www.unidata.ucar.edu/software/netcdf/
  - Bibliothèque et format de données portable.
  - Support du format HDF5 (NetCDF 4.0).
  - Parallel-NetCDF permet d'utiliser le format classique NetCDF en parallèle.
  - Beaucoup utilisé dans les applications de climatologie, de météorologie et d'océanographie.
- MPI-IO :
  - Interface d'entrée-sortie parallèle incluse dans MPI-2.
  - Attention à la portabilité des fichiers binaires.

Formats de données

Formats adaptés pour les fichiers de configurations, les fichiers de sorties etc. :

- XML (E**X**tensible **M**arkup **L**anguage) : `http://www.w3.org/XML/`
  - Langage de balisage extensible.
  - Permet de définir un format de donnée adapté à ses propres besoins
  - De nombreuses bibliothèques permettent la génération et la lecture de fichiers XML.
- JSON (**J**avaScript **O**bject **N**otation) : `http://json.org/`
  - Format d'échange de données textuelles utilisant une notation *objets* (sérialisation).
  - Facile à lire et à écrire pour un humain.
  - Il est possible de stocker des entrées JSON dans des bases de données.

Visualisation

- VTK (**V**isualization **T**oolKit) : `http://public.kitware.com/VTK/`
  - Bibliothèque d'algorithmes pour l'analyse et la visualisation de données. Permet de créer une chaîne de traitement pour produire des sorties 2D/3D.
  - Ecrit en C++. Interfaces pour les langages C++, Python, Java, ... Les données peuvent être visualisées directement ou en utilisant des outils de visualisation tels que ParaView, Visit ou Mayavi.
- OpenDX (**O**pen **D**ata **E**xplorer) : `www.opendx.org/`
  - Bibliothèque d'IBM pour la visualisation de données scientifiques.

Parallélisme en mémoire distribuée

Bibliothèques MPI (Message Passing Interface) :

- MPICH : www-unix.mcs.anl.gov/mpi/mpich/
  - Implémentation de référence des derniers standards MPI.
  - Sert de fondation à de nombreuses versions propriétaires optimisées : Intel MPI, Cray, IBM BG/Q, ...
  - Pas de support de l'InfiniBand directement dans MPICH, mais disponible dans MVAPICH.
- Open MPI : www.open-mpi.org
  - Issu de LAM/MPI. Utilisé par Bullx MPI.
  - *Historiquement* plus facile à utiliser : lancement, memchecker...
  - Attention au niveau de support des threads (`MPI_THREAD_MULTIPLE`).

Note : en C++, utiliser l'interface disponible dans Boost.

Autres modèles de programmation : PGAS (UPC, Coarray Fortran, Chapel, X10, Global Array), ...

Parallélisme en mémoire partagée :

- OpenMP : http://openmp.org
- Intel TBB, **T**hreading **B**uilding **B**locks : www.threadingbuildingblocks.org
- Interface pour les threads disponible dans C++11 ou Boost : http://en.highscore.de/cpp/boost/
- Threads POSIX (pthreads) : interface de programmation C pour gérer les threads.
- Ordonnanceurs de tâches pour les architectures multicoeurs : StarPU, QUARK, SMPSs ...

Voir aussi : https://software.intel.com/en-us/articles/choosing-the-right-threading-framework

Il existe aussi d'autres *formes* de parallélisme :

- MapReduce : pour traiter de grands jeux de données distribués.
  - Données distribuées, stockées sous la forme clé/valeur.
  - http://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/
- Bus logiciel (ORB) : intergiciel pour envoyer des requêtes entre des processus distribués :
  - CORBA (Common Object Request Broker Architecture) : architecture logicielle pour le développements de composants distribués.

Bibliothèques généralistes et autres outils

- BOOST : www.boost.org/
  - Bibliothèques C++ généralistes offrant de nombreuses fonctionnalités : algèbre linéaire, générateur de nombres aléatoires, outils systèmes, threads, sérialisation, interface C++/Python ...
- GSL (**G**nu **S**cientific **L**ibrary) : www.gnu.org/software/gsl/
  - Bibliothèque scientifique généraliste (ex : algèbre linéaire, générateur de nombres aléatoires, statistiques ...).
  - Réimplémentation ou réutilisation de bibliothèques existantes (FFTPACK, BLAS, ...).
  - Ecrit en C 'objet', interfaces pour de nombreux langages
- CMake : http://cmake.org, www.idris.fr/docs/cmake.html
  - Automatisation du processus de compilation et d'installation.

Plan

Appel croisé Fortran et C/C++

Différences Fortran/C

- Ordre des éléments en mémoire pour les tableaux multi-dimentionnels.
- Passage des paramètres de fonction (par adresse/par valeur).
- Indexation des tableaux (début à 0 ou 1).
- Types de données, chaînes de caractères (caractère char(0)).

Différences à l'édition de liens

- Fortran ignore la casse dans le nom des fonctions. Les noms sont généralement transformés en minuscule.
- Les compilateurs Fortran ajoutent normalement un underscore '_' aux noms des fonctions.
- Les options par défaut à l'édition de lien des compilateurs Fortran et C diffèrent (ex : routine d'initialisation, bibliothèque Fortran `-lgfortran`).

Appel croisé Fortran et C/C++

Premières solutions :

- Créer une fonction *wrapper* pour gérer les différences d'interfaces (noms des fonctions et arguments).
- Les compilateurs Fortran proposent des options pour gérer les conventions de nommages (ex : GNU : `-fno-underscoring`, Intel : `-assume nounderscore`).
  https://gcc.gnu.org/onlinedocs/gfortran/Code-Gen-Options.html
- Utiliser le linker Fortran plutôt que celui du C pour les codes mixtes Fortran/C.

Voir aussi : http://docs.oracle.com/cd/E19205-01/819-5262/6n7bvdr18/

Appel croisé Fortran et C/C++

Méthode d'interfaçage recommandée

- La norme 2003 de Fortran définit comment interfacer des procédures Fortran avec des fonctions C.
- Cette solution est **portable** et permet de faire correspondre les **types** Fortran avec ceux du langage C.

```
module mywrapper
  use ISO_C_BINDING
  interface
    subroutine myfunction(array, size) bind(C, name='myfunction')
      import C_DOUBLE, C_INT
      real(kind=C_DOUBLE), dimension(*) :: array
      integer(kind=C_INT), value        :: size
    end subroutine myfunction
  end interface
end module mywrapper
```

Voir aussi : http://www.idris.fr/data/cours/lang/fortran/f2003/ (chapitre 9)
et https://gcc.gnu.org/onlinedocs/gfortran/Interoperability-with-C.html

Appel d'autres langages depuis Python

- Appels de code C++ depuis Python :
  - SWIG (**S**implified **W**rapper and **I**nterface **G**enerator) : appels de programmes C et C++ depuis des langages de programmation haut niveau (Python, PHP, Perl ...), www.swig.org/
  - BoostPython : interopérabilité C++/Python. www.boost.org/doc/libs/1_42_0/libs/python/
  - PyRex : écrire du Python avec les types de données du C. www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/
- Appels de code Fortran depuis Python :
  - f2py, **F**ortran **to** **Py**thon interface generator. http://cens.ioc.ee/projects/f2py2e/
  - PyFort : création d'extensions python à partir de routines Fortran. http://pyFortran.sourceforge.net/

Plan

TP 1 : produit de deux matrices denses

Objectif

Calculer efficacement le produit de deux matrices denses générées aléatoirement en utilisant une bibliothèque externe (en C ou en Fortran, au choix).

Compilation

- Les fichiers `blas_matmat.c` et `blas_matmat.f90` du répertoire `MatMatMult` implémentent une première version du produit matriciel.
- Un `Makefile` est fourni pour compiler ces exemples.

TP1 : produit de deux matrices denses (MatMatMult)

Produit matriciel

- Identifier l'appel BLAS permettant de réaliser un produit matriciel. http://www.netlib.org/blas/#_documentation
- Ecrire l'appel BLAS (ou CBLAS) et vérifier l'exactitude des résultats.
- Comparer les temps d'exécution à la version de référence pour différentes tailles de matrices.

Générateur de nombres aléatoires

- Remplacer le générateur de nombres aléatoires par des fonctions de la GSL (voir fichier `rng.c`). www.gnu.org/software/gsl/manual/gsl-ref.html#Random-Number-Generation
- En Fortran, il est nécessaire d'écrire un wrapper pour appeler les fonctions C (fichier `rng_wrapper.f90`).
- Utiliser la ligne de commande pour changer le générateur de nombres et la graine aléatoire.

TP 2 : résolution d'un système linéaire dense

Objectif

Résoudre le système linéaire $Ax = b$ en utilisant une décomposition LU.

Travail à réaliser

- Identifier l'appel LAPACK permettant de réaliser une factorisation LU.
- Implémenter la résolution du système.

TP 3 : résolution d'un système linéaire creux

Objectif

Utiliser PETSc pour résoudre l'équation de Poisson.

Travail à réaliser

- `ex1.c` et `ex1f.F` implémentent la résolution du système. Identifier la méthode de résolution utilisée.
- En ligne de commande, tester les options suivantes : `-ksp_monitor`, `-ksp_view`, `-log_summary`.
- En ligne de commande, remplacer le préconditionneur par un SOR (successive over-relaxation) symétrique.
- Changer le préconditionneur directement dans le code.
- Implémenter le calcul de la norme du résidu et comparer le résultat avec celui de l'option `-ksp_monitor_true_residual`.
- Etudier les différences entre `ex1.c` et la version parallèle disponible dans `ex23.c`.

TP 4 : résolution d'un système linéaire creux

Objectif

Utiliser Trilinos pour résoudre l'équation de Poisson.

Travail à réaliser

- `IfpackBelos.cpp` implémente la résolution du système. Quelle méthode de résolution est utilisée ?
- Identifier le role des paquets `Ifpack` et `Belos`. Comment ces paquets interagissent-t-ils entre eux ?
- Remplacer la méthode de résolution par une méthode de Jacobi.