Université Versailles-Saint-Quentin

Paris-Saclay

Spécialité :
Calcul Haute Performance, Simulation

Module :
Calcul numerique

January 7, 2026

Rapport :
Rapport CN TDP 5.

Présenté par:
- OUNNAS Masten

Encadré par:
- Jerome Gurhem.
- Thomas Dufaud.

Contents

1 Introduction

L'objectif de ce TP est d'appliquer des algorithmes de résolution de systèmes linéaires, étudiés en cours, au problème physique de l'équation de la chaleur stationnaire 1D. Nous nous intéressons particulièrement à la résolution numérique par la méthode des différences finies, en exploitant les bibliothèques de calcul haute performance BLAS et LAPACK. Ce rapport se concentre sur les méthodes directes (Factorisation LU) et compare l'efficacité des routines génériques de LAPACK face à une implémentation spécifique optimisée pour la structure tridiagonale du problème.

2 Exercice 1 : Modélisation et Discrétisation

Nous considérons l'équation de la chaleur dans un milieu homogène isotrope :

$$\begin{cases} -k\frac{\partial^2 T}{\partial x^2} = g, \quad x \in]0,1[\\ T(0) = T_0 \\ T(1) = T_1 \end{cases} \tag{1}$$

Où $k > 0$ est la conductivité thermique et g un terme source (ici supposé nul $g = 0$).

2.1 Approximation de la dérivée seconde

Nous discrétisons le domaine $[0, 1]$ en $n+2$ points x_i espacés d'un pas constant $h = \frac{1}{n+1}$. En utilisant un développement de Taylor-Young à l'ordre 2 pour $T(x+h)$ et $T(x-h)$, nous obtenons l'approximation centrée de la dérivée seconde :

$$T''(x_i) \approx \frac{T(x_{i-1}) - 2T(x_i) + T(x_{i+1})}{h^2} \tag{2}$$

2.2 Système Linéaire

L'équation discrétisée s'écrit en chaque nœud interne :

$$-k\frac{T_{i-1} - 2T_i + T_{i+1}}{h^2} = g_i \tag{3}$$

Cela conduit à la résolution d'un système linéaire $Ax = b$ où $A \in R^{n\times n}$ est une matrice tridiagonale symétrique définie positive de la forme :

$$A = \frac{k}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \tag{4}$$

3 Exercice 2 : Environnement de Développement

Afin de garantir la portabilité du code et la reproductibilité des résultats (critère essentiel en HPC), l'environnement de développement a été conteneurisé via **Docker**.

- **Langage :** C (Norme C99).
- **Bibliothèques :** BLAS (Basic Linear Algebra Subprograms) et LAPACK (Linear Algebra PACKage).

- **Compilation :** Automatisée via un `Makefile` liant les librairies statiques ou dynamiques selon l'architecture.
- **Gestion de version :** Le projet est versionné sous Git.

4 Exercices 3 et 4 : Stockage Bande et Validation

4.1 Stockage General Band (GB)

La matrice A étant creuse, le stockage dense (n^2 éléments) est inefficace en mémoire cache. Nous utilisons le format *General Band* de LAPACK en mode *Column-Major*. Pour une matrice de dimension N avec kl sous-diagonales et ku sur-diagonales, ce format requiert un tableau de dimension $(2kl + ku + 1) \times N$ pour permettre le pivotage lors de la factorisation LU.

Listing 1: Implémentation du stockage GB pour Poisson 1D

```
void set_GB_operator_colMajor_poisson1D(double* AB, int *lab, int *la, int *kv){
  int m = *la;
  int n = *lab;
  int v = *kv;

  for(int i = 0; i < m; ++i){
    int k = i * n;
    // Fill-in initialization
    if(v >= 0){
      for(int j = 0; j < v; ++j) AB[k+j] = 0.0;
    }
    AB[k+v] = -1.0;    // Sur-diagonale
    AB[k+v+1] = 2.0;   // Diagonale
    AB[k+v+2] = -1.0; // Sous-diagonale
  }
  // Conditions aux bords
  AB[0] = 0.0;
  AB[n*m - 1] = 0.0;
}
```

4.2 Validation numérique

La validation a été effectuée en calculant l'erreur relative $\frac{||Ax_{exact} - b||}{||b||}$ via la routine BLAS `cblas_dgbmv`. Nous obtenons une erreur de l'ordre de 10^{-16}, ce qui correspond à la précision machine (double précision), validant ainsi notre construction matricielle.

5 Exercices 5 et 6 : Résolution Directe et Performance

Nous comparons ici deux approches pour la résolution du système $Ax = b$.

5.1 Méthode 1 : LAPACK Standard (Exercice 5)

Nous utilisons la routine `dgbtrf` pour la factorisation LU suivie de `dgbtrs` pour la résolution. Cette méthode est robuste mais généraliste :

- Elle gère des largeurs de bande arbitraires.
- Elle effectue un pivotage partiel pour la stabilité numérique, ce qui implique des tests conditionnels et des échanges de lignes.

5.2 Méthode 2 : Factorisation LU Optimisée (Exercice 6)

Nous avons implémenté une routine dédiée `dgbtrftridiag`. La matrice du Laplacien étant à diagonale dominante stricte ($|2| > |-1| + |-1|$), la factorisation LU est stable sans pivotage. Notre optimisation consiste à :

1. Supprimer le pivotage (réduction des accès mémoire indirects).
2. Restreindre les calculs strictement aux 3 diagonales non-nulles, évitant le traitement du "fill-in" potentiel géré par LAPACK.

5.3 Analyse des performances

Les temps d'exécution ont été mesurés pour des tailles de matrice N variant de 100 à 50 000.

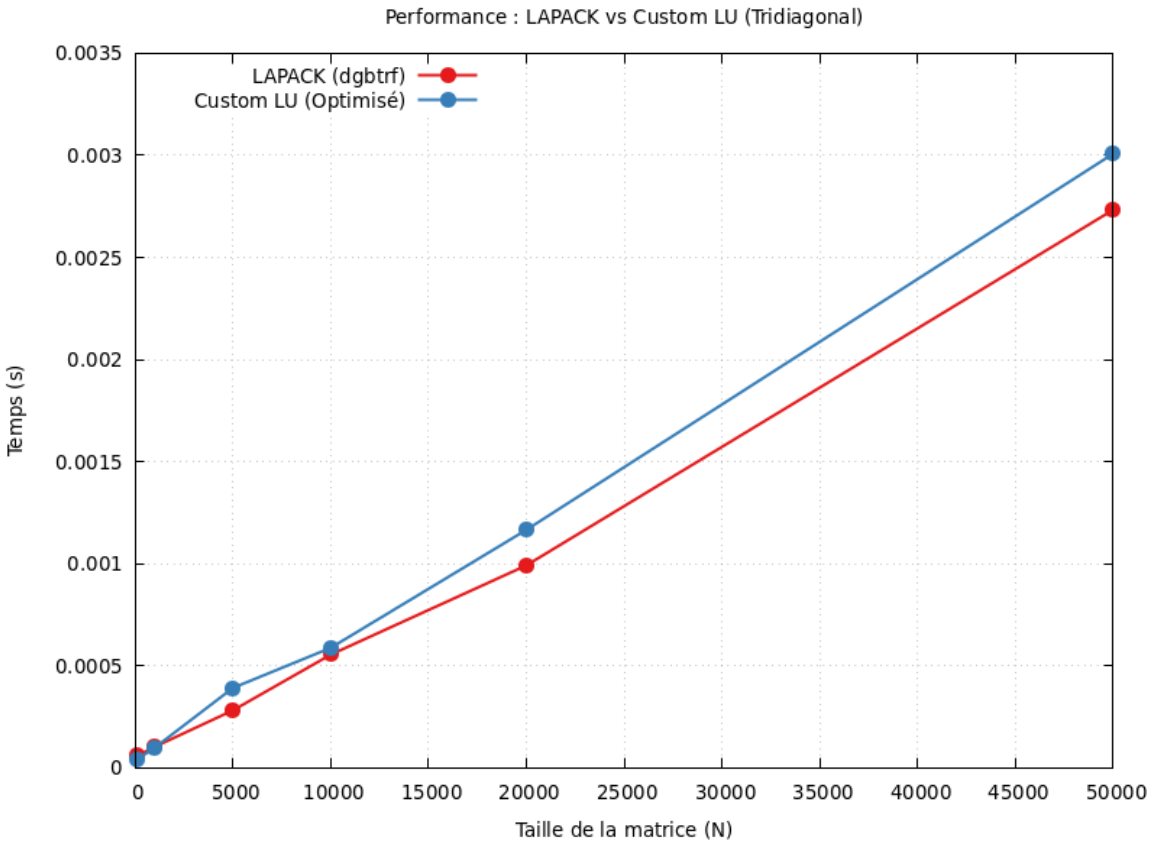

Figure 1: Comparaison des temps d'exécution : LAPACK (dgbtrf) vs Custom LU

Interprétation des résultats :

- **Complexité :** Les deux courbes confirment une complexité linéaire $O(N)$. Ceci est cohérent avec la théorie des matrices bandes, contrairement à la complexité cubique $O(N^3)$ des matrices denses.
- **Comparaison :** On observe que les performances de l'implémentation "maison" (Custom LU) sont très proches, voire légèrement inférieures sur cette architecture, à celles de LAPACK.
- **Analyse HPC :** Bien que notre algorithme effectue théoriquement moins d'opérations flottantes (FLOPs), la bibliothèque LAPACK bénéficie d'optimisations bas niveau extrêmement poussées (vectorisation AVX/SSE, gestion optimale du cache CPU, déroulage de boucles). Notre code C "naïf", bien qu'algorithmiquement optimisé, peine à battre ces routines assembleurs sur des calculs vectoriels séquentiels simples.

6 Conclusion

Ce TP a permis de valider la résolution de l'équation de la chaleur 1D par des méthodes directes. L'utilisation du format de stockage bande permet de passer d'une complexité quadratique en mémoire et cubique en temps à une complexité linéaire ($O(N)$). La comparaison entre une routine générique optimisée (LAPACK) et une routine spécifique montre que le gain algorithmique (suppression du pivotage) ne suffit pas toujours à surpasser les optimisations matérielles intégrées aux bibliothèques standards pour des opérations de complexité $O(N)$.