Utilisation d'Intel MKL en Calcul Scientifique

Application aux Vibrations de Membranes Non Uniformes

AHOTONHOUN Aimé Césaire
BONOU Justus
HANDJEMEDJI Ezéchiel

ENSGMM-ABOMEY
Université Nationale des Sciences Technologies, Ingénierie et Mathématiques (UNSTIM)
Sous la supervision de : Dr. Carlos AGOSSOU

Année Académique 2025-2026

Plan de la Présentation

Contexte du Projet

Les vibrations de membranes

Phénomène physique omniprésent dans :

- Instruments de musique
- Capteurs mécaniques
- Isolation vibratoire
- Biologie cellulaire

Objectifs principaux

1. Comprendre la physique des vibrations
2. Simulation numérique avancée
3. Maîtriser **Intel MKL**
4. Analyser et visualiser

Parallélisme

LAPACK

BLAS

Membrane Uniforme : Solution Analytique

Équation des ondes

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u \text{ avec } c = \sqrt{T/\rho}$$

Modes propres de vibrations

$$\phi_{n,m}(x,y) = \sin\left(\frac{n\pi x}{L_x}\right) \sin\left(\frac{m\pi y}{L_y}\right)$$

Conditions aux limites

$$u(0,y,t) = u(Lx,y,t) = 0$$

$$u(x,0,t) = u(x,Ly,t) = 0$$

Fréquences propres

$$\omega_{n,m} = c\sqrt{\left(\frac{n\pi}{L_x}\right)^2 + \left(\frac{m\pi}{L_y}\right)^2}$$

Solution par séparation

$$u(x,y,t) = X(x)Y(y)G(t)$$

Interprétation physique

Exemples de modes

- Mode (1,1) : Toute la membrane dans le même sens
- Mode (1,2) : Une moitié monte, l'autre descend
- Mode (2,1) : Même chose mais selon l'autre direction
- Mode (2,2) : Quatre régions alternent haut/bas

Membrane Non Uniforme : Modèle Réaliste

Équation générale

$$-\nabla \cdot \big(p(x,y)\nabla u(x,y)\big) + q(x,y)u(x,y) = \lambda w(x,y)u(x,y)$$

Conditions aux limites

$$u(x,y) = 0 \text{ sur } \partial\Omega$$

Coefficients variables

$$p(x,y) = 1 + 0.5\sin(2\pi x)\cos(2\pi y)$$
$$w(x,y) = 1 + 0.3xy$$
$$q(x,y) = 50e^{-50[(x-0.5)^2+(y-0.5)^2]}$$

ENSGMM
«Les mathématiques au centre de toutes les disciplines»

Maillage uniforme

$$h = \frac{1}{N+1}, \quad x_i = ih, \quad y_j = jh$$

Discrétisation du Laplacien

$$\nabla \cdot (p\nabla u) \approx \frac{1}{h^2}\left[p_{i+\frac{1}{2},j}(u_{i+1,j} - u_{i,j}) - p_{i-\frac{1}{2},j}(u_{i,j} - u_{i-1,j})\right]$$

$$+\frac{1}{h^2}\left[p_{i,j+\frac{1}{2}}(u_{i,j+1} - u_{i,j}) - p_{i,j-\frac{1}{2}}(u_{i,j} - u_{i,j-1})\right]$$

Discrétisation complète

$$-\frac{1}{h^2}\left[p_{i+\frac{1}{2},j}(u_{i+1,j} - u_{i,j}) - p_{i-\frac{1}{2},j}(u_{i,j} - u_{i-1,j})\right]$$
$$-\frac{1}{h^2}\left[p_{i,j+\frac{1}{2}}(u_{i,j+1} - u_{i,j}) - p_{i,j-\frac{1}{2}}(u_{i,j} - u_{i,j-1})\right] + q_{i,j}u_{i,j} = \lambda w_{i,j}u_{i,j}$$

ENSGMM
«Les mathématiques au centre de toutes les disciplines»

Discrétisation

Problème discrétisé

$$A\mathrm{u} = \lambda B\mathrm{u}$$

- A : Matrice de rigidité (creuse)
- B : Matrice de masse (diagonale)
- $\lambda = \omega^2$: Valeurs propres
- u : Vecteur des déplacements aux nœuds

«Les mathématiques au centre de toutes les disciplines»

Solveurs de Valeurs Propres dans MKL

Solveur	Type,Problème	Application
dsyev	Dense, standard	Petites matrices
dsygv	Dense, généralisé	Notre choix
FEAST	Creux, sélectif	Grands systèmes
PARDISO+	Creux, standard	Problèmes standards

Table – Solveurs disponibles dans Intel MKL

Pourquoi dsygv ?

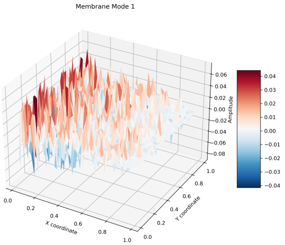
- Matrices symétriques réelles
- Problème généralisé
- Stabilité numérique
- Tous les modes calculés
- Simple à implémenter
- Bon pour taille modérée

ENSGMM «Les mathématiques au centre de toutes les disciplines»

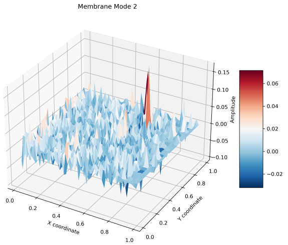
Code : Utilisation de dsygv

```c
#include <mkl.h>

int main() {
    int n = 2500;        // 50x50 points
    int itype = 1;       // A*x = lambda*B*x
    char jobz = 'V';     // Valeurs et vecteurs propres
    char uplo = 'U';     // Stockage triangle sup rieur
    double *A = (double*)mkl_malloc(n*n*sizeof(double), 64);
    double *B = (double*)mkl_malloc(n*n*sizeof(double), 64);
    double *w = (double*)mkl_malloc(n*sizeof(double), 64);
    int lwork = -1;
    double work_query;
    dsygv(&itype, &jobz, &uplo, &n, A, &n, B, &n,
          w, &work_query, &lwork, &info);
    lwork = (int)work_query;
    double *work = (double*)mkl_malloc(lwork*sizeof(double), 64);
    dsygv(&itype, &jobz, &uplo, &n, A, &n, B, &n,
          w, work, &lwork, &info);

    mkl_free(A); mkl_free(B); mkl_free(w); mkl_free(work);
    return 0;
}
```

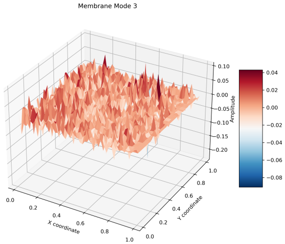
Visualisation des Modes Propres



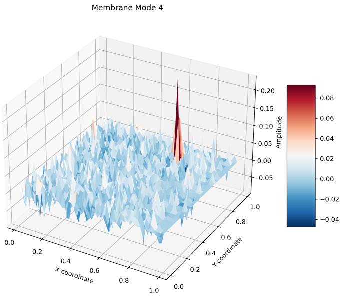
Mode 1



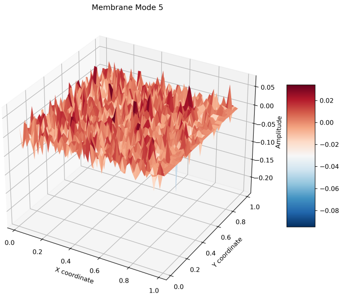
Mode 2



Mode 3



Mode 4



Mode 5

Figure – Les cinq premiers modes de vibration de la membrane non uniforme

Analyse des Performances

Résultats pour N=50

- **Taille** : 2500×2500
- **Mémoire** : $\sim$ 50 Mo
- **Temps** : 8.877 s
- **Threads** : 4
- **Précision** : Double

Limitations

- Stockage dense d'une matrice creuse
- 99.8% de zéros stockés
- Non scalable pour $N > 100$

Pour les grands systèmes

1. Format CSR : Stocker seulement les éléments non nuls
2. Solveur FEAST : Extraire sélectivement les valeurs propres désirées
3. Solveur itératif : Méthode de Lanczos ou Arnoldi pour les très grands

Récapitulatif

Accomplissements

- Modélisation physique complète
- Discrétisation par différences finies
- Implémentation avec **Intel MKL**
- Résolution numérique efficace
- Visualisation des résultats

Physique

Mathématiques

Informatique

Compétences développées

- Calcul scientifique haute performance
- Utilisation de BLAS/LAPACK
- Analyse numérique
- Visualisation scientifique

Code Source Complet

github.com/AyomHead/projet_calcul_scientifique

Contenu

- Code C avec MKL
- Scripts Python
- Données numériques
- Visualisations
- Document LaTeX

Références

- Intel MKL Documentation
- LAPACK User's Guide
- Numerical Linear Algebra
- Finite Element Procedures

MERCI POUR VOTRE ATTENTION

Vos questions ou approches sont les bienvenues.