Rapport : Méthode directe et stockage bande

SY Aboubacar

17 décembre 2025

Exercice 5 : DGBTRF, DGBTRS, DGBSV

DGBTRF : `dgbtrf` - Factorisation LU

Objectif : Effectue la factorisation LU d'une matrice bande.
Opération :

$$A = L \cdot U$$

où L est une matrice triangulaire inférieure et U une matrice triangulaire supérieure.
Format : Stocke les matrices L et U dans le tableau d'entrée `AB`.
Utilisation :

```
dgbtrf_(&la, &la, &kl, &ku, AB, &lab, ipiv, &info);
```

Complexité :
— arithmétique : $\mathcal{O}(n \cdot (kl * ku))$ pour une matrice bande de diagonale n. Car pour chaque colonne on va : calculer le/les pivot (kl division), le soustraire à (ku) éléments à droite.
— mémoire : étant donner qu'ont réécrit dans la même matrice, la mémoire n'augmente pas donc : $\mathcal{O}(n * (kl + ku))$

DGBTRS : `dgbtrs` - Résolution du système

Objectif : Résout un système linéaire $A\mathbf{x} = \mathbf{b}$ après factorisation.
Prérequis : Doit être appelé après une factorisation LU, soit DGBTRF ou notre implémentation de la factorisation LU tridiagonale.
Opération :

$$\begin{cases} L\mathbf{y} = P\mathbf{b} & \text{(descente)} \\ U\mathbf{x} = \mathbf{y} & \text{(remonter)} \end{cases}$$

Utilisation :

```
dgbtrs_("N", &la, &kl, &ku, &NRHS, AB, &lab, ipiv, RHS, &la, &info);
// AB -> matrice LU
// RHS -> b

```

Complexité :
— arithmétique : $\mathcal{O}(descente + remonte)$ -> $\mathcal{O}(n * kl + n * ku)$ car pour chaque ligne il peut y avoir au maximum kl/ku éléments. Pour une matrice bande de diagonale n, kl nombre de sous-diagonale et ku nombre de sur-diagonale.
— mémoire : étant donner qu'ont écrit dans la même matrice, la mémoire n'augmente pas donc : $\mathcal{O}(n * (kl + ku))$

DGBSV : `dgbsv` - Routine complète

Objectif : Résout un système linéaire en faisant une factorisation LU.
Opération :

$$\text{DGBSV} = \text{DGBTRF} + \text{DGBTRS}$$

$$AX = B$$

Utilisation :

```
dgbsv_(&n, &kl, &ku, &nrhs, AB, &ldab, ipiv, RHS, &ldb, &info);
// AB -> matrice bande d'ordre n, avec ku sur-diagonales et kl sous-diagonales
// RHS -> b

```

Complexité : Même que DGBTRF + DGBTRS.

— arithmétique : $\mathcal{O}(DGBTRF + DGBTRF) = \mathcal{O}(n \cdot (kl + ku) + n * kl + n * ku) = \mathcal{O}(n \cdot (2kl + 2ku)$ pour une matrice bande de diagonale n.

— mémoire : étant donner qu'ont réécrit dans la même matrice, la mémoire n'augmente pas donc : $\mathcal{O}(n * (kl + ku))$

2. Évaluation des performances et complexité

Taille n	Temps DGBTRF (s)	Temps DGBTRS (s)	Temps DGBSV (s)	Erreur résiduelle
10	0.000079	0.000003	0.000098	5.249507×10^{-8}
100	0.000122	0.000010	0.000077	1.207707×10^{-6}
10000	0.000698	0.000569	0.002051	1.052995×10^{-4}
100000	0.004275	0.003620	0.007898	1.673769×10^{-3}

TABLE 1 – Performances des fonctions LAPACK

Analyse de complexité :

On observe une évolution de temps linéaire pour DGBTRF et DGBTRS le temps d'exécution augmente proportionnellement à la taille n du problème, confirmant les complexité théorique. Pour DGBSV on remarque que pour les matrices de petite dimension elle est plus rapide de DGBTRF + DGBTRS, mais devient moins performante pour les matrices de grande dimension. L'erreur résiduelle croît nécessairement à cause de l'accumulation des erreurs d'arrondi lors des opérations en précision flottante

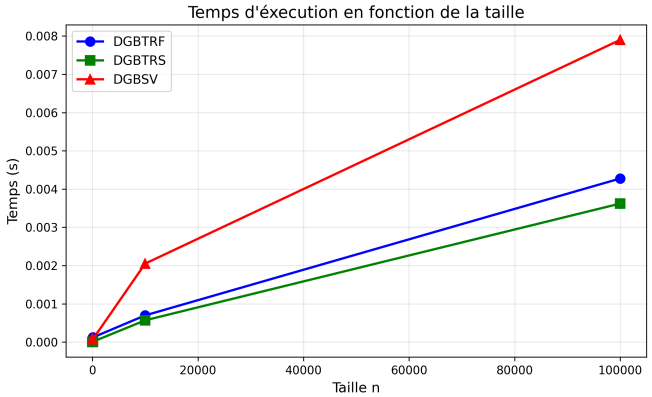

FIGURE 1 – Temps d'exécution en fonction de la taille n

Exercice 6 : LU pour matrices tridiagonales

1. Implémentation de la factorisation LU

Pour une matrice tridiagonale A (format GB avec $kl = ku = 1$), l'algorithme de factorisation LU spécialisé est :

```
int indexABCol(int i, int j, int *lab){
// TODO: Return the correct index formula for column-major band storage;
return (i-j)+(*lab)*j +2;
}

int dgbtrftridiag(int *la, int*n, int *kl, int *ku, double *AB, int *lab, int *
    ipiv, int *info){
// TODO: Implement specialized LU factorization for tridiagonal matrices

*ipiv = 1;
double e = 1e-12;
int _n = *n;
for(size_t i =1;i<_n;i++){

double mi = AB[indexABCol(i,i-1,lab)] / AB[indexABCol(i-1,i-1,lab)];
AB[indexABCol(i,i-1,lab)] = mi; // mi -> multiplicateur

AB[indexABCol(i,i,lab)] = AB[indexABCol(i,i,lab)] - AB[indexABCol(i-1,i,lab)]*mi
    ;

}

*info = 0;
return *info;
}

```

Factorisation LU pour matrices tridiagonales L'algorithme implémente une factorisation LU optimisée pour les matrices tridiagonales au format bande LAPACK. Pour chaque élément $a_{i,i-1}$ de la sous-diagonale L, il calcule le multiplicateur $m_i = \dfrac{a_{i,i-1}}{a_{i-1,i-1}}$ qu'il stocke à sa place. L'élément diagonal est ensuite mis à jour selon $u_{i,i} = a_{i,i} - a_{i-1,i} \times m_i$

2. Méthode de validation

Lorsque la solution analytique exacte $\mathbf{x}_{\text{exact}}$ est disponible, une validation peut être proposé en effectuant le calcul de l'erreur relative :

$$e_{\text{rel}} = \frac{\|\mathbf{x}_{\text{calc}} - \mathbf{x}_{\text{exact}}\|}{\|\mathbf{x}_{\text{exact}}\|}$$

Puisqu'on a nos disposition la fonction DGBSV, la solution obtenue avec DGBSV peut être utilisée comme $\mathbf{x}_{\text{exact}}$. L'algorithme est alors considéré comme validé lorsque l'erreur relative est inférieure à une tolérance fixée t :

$$\boxed{e_{\text{rel}} < \text{t} \quad \Rightarrow \quad \text{Algorithme validé}}$$

3. Évaluation des performances et comparaison

Taille n	LU triag (s)	DGBSV (s)	Erreur LU triag	Erreur DGBSV
1000	0.000048	0.000089	9.529314×10^{-1}	1.039657×10^{-6}
5000	0.000298	0.000311	9.808251×10^{-1}	5.490079×10^{-5}
10000	0.000325	0.000451	9.867498×10^{-1}	1.052995×10^{-4}
50000	0.001499	0.002159	9.942546×10^{-1}	8.163927×10^{-4}
100000	0.003116	0.004473	9.959745×10^{-1}	1.673769×10^{-3}

TABLE 2 – Comparaison des performances et précision entre notre implémentation maison et DGBSV

Le tableau révèle un problème majeur : l'implémentation maison, bien que légèrement plus rapide, produit des erreurs catastrophiques ($\sim$0.99) comparées à DGBSV (10^{-6} à 10^{-3}). Cette énorme différence suggère que l'algorithme maison contient une erreur fondamentale. Les erreurs de DGBSV augmentent avec la taille, alors que l'implémentation maison échoue complètement, produisant des solutions inutilisable malgré une factorisation LU apparemment correcte.