Implémentation d'une matrice CSR et utilisation de cette matrice pour la résolution de système linéaire

Généré par Doxygen 1.8.13

Table des matières

Chapitre 1

Utilisation de matrice CSR pour la résolution de AX = B

1.1 Présentation

Le module CSR implémente un stockage efficient, en terme de mémoire, pour des matrices creuses. Dans un premier temps, on utilise la décomposition LU pour résoudre le problème. Dans un second temps, on utilise la méthode du gradient conjugué pour résoudre le problème.

1.2 CSR

```
TYPE csr_matrix
        INTEGER , DIMENSION(:), ALLOCATABLE     :: m_rowidx
        INTEGER , DIMENSION(:), ALLOCATABLE     :: m_col
        REAL(KIND=selected_real_kind(15, 307)) , DIMENSION(:), ALLOCATABLE      :: m_val

        INTEGER                                 :: m_m ! Number of row
        INTEGER                                 :: m_n ! Number of column


END TYPE csr_matrix
```

L'implémentation du mode de stockage CSR est fait de tel sorte que rowIdx commence par un 0 et que les colonnes sont indéxés à partir de 1. Pour utiliser des matrices avec des matrices ayant des colonnes indéxés à partir de 0, il suffit juste de rajouter 1 à m_col. Ceci peut être mis en place très facilement grâce à une subfunction.

1.3 LU

On utilise la décomposition LU de la matrice A pour obtenir la solution. Cela revient, alors, à résoudre deux système triangulaire.

1.4 Gradient Conjugué

La méthode du gradient conjugué peut être utilisé de deux façon. Soit de manière complète, dans ce cas, l'intégralité des calculs sont fait pour obtenir la solution, soit de manière à rechercher une solution avec une précision donnée, dans ce cas, quand la précision de la solution est suffisante la fonction la retourne directement.

1.5 Solver

Ce module regroupe les appelles pour résoudre un système $AX = B$. Les deux méthodes proposés sont LU et gradient conjugué.

1.6 TEST

Ce module implémente quelque test pour vérifier le fonctionnement basique de l'ensemble des modules. Ceci permet d'évaluer rapidement si l'ajout ou modification de fonction/subroutine aux différents modules modifie le comportement attendu.

1.7 Compilation

Pour compiler le code executer le script *compile.sh*.

1.8 Exemple

On importe, dans un premier temps les modules nécéssaires.

```
USE csr
USE lu
USE conjugate_gradient
USE solver
USE unit_test
```

Dans cet exemple, on va créer une matrice **A** et un vecteur-colonne **B** charger depuis un fichier. On crée aussi un vecteur-colonne pour chaque solution X et Y.

```
TYPE(csr_matrix) a
TYPE(csr_matrix) b
TYPE(csr_matrix) x
TYPE(csr_matrix) y
TYPE(csr_matrix) r
```

On peut alors charger les matrices depuis les fichiers.

```
a = create_csr_matrix_from_file("A.csr")
b = create_csr_matrix_from_file("B.csr")
```

On va ensuite calculer la solution du sytème $AX = B$ en utilisant les deux méthodes implémentés. X est la solution utilisant la décomposition LU tandis que Y est la solution utilisant la méthode du gradient conjugué.

```
x = solver_lu(a, b)
y = solver_gc(a, b)
```

On peut alors, afficher le résultat des deux méthodes.

```
WRITE (*,*) "Solution"
WRITE(*,*) "LU :"
CALL x%DISPLAY
WRITE(*,*) "GC :"
CALL y%DISPLAY
```

On peut alors, afficher l'erreur des deux méthodes.

```
WRITE (*,*) "Erreur"
r = csr_matrix_vector_sub(csr_matrix_prod(a,x), b)
WRITE(*,*) "LU :", sqrt(csr_matrix_inner_product(r,r))
r = csr_matrix_vector_sub(csr_matrix_prod(a,y), b)
WRITE(*,*) "GC :", sqrt(csr_matrix_inner_product(r,r))
```

On sauvegarde alors la matrice A dans le fichier **svg1**.

```
CALL a%CSR_MATRIX_SAVE_TO_FILE("A.csr")
CALL b%CSR_MATRIX_SAVE_TO_FILE("B.csr")
```

Il est possible d'initialiser à la main les matrices **A** et **B**. On doit alors créer une matrice vide de dimension voulue puis initialiser les valeurs non nulles

```
a = create_csr_matrix_square(3)
b = create_csr_matrix(3, 1)

CALL set(a, 1, 1, 5.d0)
CALL set(a, 2, 1, 1.d0)
CALL set(a, 2, 2, 1.d0)
CALL set(a, 2, 3, 8.d0)
CALL set(a, 3, 1, 7.d0)
CALL set(a, 3, 2, 1.d0)
CALL set(a, 3, 3, 2.d0)
CALL set(a, 3, 2, 10.d0)

CALL set(b, 1, 1, 1.d0)
CALL set(b, 2, 1, 1.d0)
CALL set(b, 3, 1, 1.d0)
```

Il est aussi possible de tester le fonctionnement basique des modules.

```
CALL test
```

A la fin, on doit désallouer les matrices utilisés.

```
CALL free_csr_matrix(a)
CALL free_csr_matrix(b)
CALL free_csr_matrix(x)
CALL free_csr_matrix(y)
CALL free_csr_matrix(r)
```

Chapitre 2

Index des modules

2.1 Liste des modules

Liste de tous les modules avec une brève description :

Chapitre 3

Index du type de données

3.1 Liste des types de données

Liste des types de données avec une brève description :

Chapitre 4

Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 5

Documentation des modules

5.1 Référence du module conjugate_gradient

Ce module permet d'utiliser l'algorithme du gradient conjugué. Deux implémentations sont fournies, la première effectue le nombre maximal d'étape, la seconde s'arrête à une précision donnée.

Fonctions/Subroutines

— type(csr_matrix) function conjugate_gradient_direct (A, B, eps)
Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué Source https://www.ljll.↩ math.upmc.fr/hecht/ftp/InfoSci/doc-pdf/Master_book_GC.pdf.

5.1.1 Description détaillée

Ce module permet d'utiliser l'algorithme du gradient conjugué. Deux implémentations sont fournies, la première effectue le nombre maximal d'étape, la seconde s'arrête à une précision donnée.

Auteur

Mechineau Alexandre

5.1.2 Documentation de la fonction/subroutine

5.1.2.1 conjugate_gradient_direct()

```
type(csr_matrix) function conjugate_gradient::conjugate_gradient_direct (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B,
            real, intent(in), optional eps )
```

Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué Source https://www.ljll.↩ math.upmc.fr/hecht/ftp/InfoSci/doc-pdf/Master_book_GC.pdf.

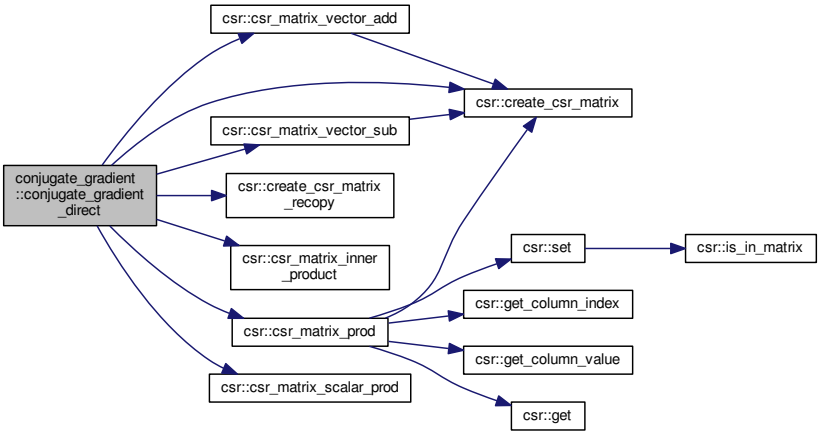
Paramètres

A	Matrice CSR
B	Matrice CSR représentant un vecteur colonne
eps	Précison requise (Optionel)

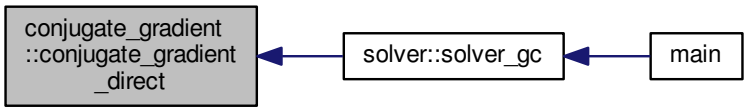
Renvoie

Approximation, dans le pire des cas, de la solution du système

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2 Référence du module csr

Ce module permet de créer,stocker et manipuler une matrice sous la forme CSR. Plusieurs opérations courantes sont implémentés, avec certains rafinement pour le cas de vecteur.

Les types de données

— type csr_matrix
Type décrivant une matrice CSR Attention m_col stocke l'index des colonnes en commencant par 1. [CSR].

Fonctions/Subroutines

- type(csr_matrix) function create_csr_matrix_from_file (f)
  Cette fonction permet de créer une matrice depuis un fichier. La matrice dans le fichier doit utilisé le style C-CSR modifié pour que les colonnes commence par être indexé par 1. ***On admet que le fichier passé est valide***
- subroutine csr_matrix_save_to_file (M, f)
  Cette subroutine permet de sauvegarder une matrice CSR dans un fichier.
- type(csr_matrix) function create_csr_matrix_square (n)
  Cette fonction permet de créer une matrice carrée vide.
- type(csr_matrix) function create_csr_matrix_recopy (M)
  Cette fonction permet de dupliquer une matrice.
- type(csr_matrix) function create_csr_matrix (m, n)
  Cette fonction crée une matrice de taille quelconque.
- subroutine free_csr_matrix (M)
  Cette subroutine permet de détruire une matrice CSR.
- logical function is_in_matrix (M, row, col)
  Cette fonction permet de vérifier que des indices sont bien valide pour une matrice donnée.
- logical function is_column_vector (V)
  Cette fonction permet de vérifier si une matrice est un vecteur colonne.
- real(kind=selected_real_kind(15, 307)) function get (M, row, col)
  Cette fonction permet d'obtnir la valeur d'un élément d'une matrice.
- subroutine set (M, row, col, val)
  Cette subroutine permet de modifier une valeur dans une matrice CSR.
- subroutine display_debug (M)
  Cette subroutine affiche une matrice CSR avec des informations avancées.
- subroutine display (M)
  Cette subroutine affiche une matrice CSR.
- integer function, dimension(:), allocatable get_column_index (M, row)
  Cette fonction permet d'obtenir les colonnes non-nulles d'une ligne donnée.
- real(kind=selected_real_kind(15, 307)) function, dimension(:), allocatable get_column_value (M, row)
  Cette fonction permet d'obtenir les valeurs non-nulles d'une ligne donnée.
- type(csr_matrix) function csr_matrix_prod (A, B)
  Cette fonction calcule le produit de deux matrices.
- logical function csr_matrix_equal (A, B)
  Cette fonction permet de vérifier si deux matrices sont égales.
- type(csr_matrix) function tr (M)
  Cette fonction calcule la transposée d'une matrice.
- real(kind=selected_real_kind(15, 307)) function csr_matrix_inner_product (A, B)
  Cette fonction calcule le produit scalaire de deux vecteurs.
- type(csr_matrix) function csr_matrix_vector_add (A, B)
  Cette fonction calcule la somme de deux vecteurs colonnes.
- type(csr_matrix) function csr_matrix_vector_sub (A, B)
  Cette fonction calcule la différence de deux vecteurs colonnes.
- type(csr_matrix) function csr_matrix_additive_inverse (A)
  Cette fonction calcule l'inverse associé à la loi additive d'une matrice.
- type(csr_matrix) function csr_matrix_scalar_prod (M, K)
  Cette fonction calcule le produit entre un réel et une matrice.

5.2.1 Description détaillée

Ce module permet de créer,stocker et manipuler une matrice sous la forme CSR. Plusieurs opérations courantes sont implémentés, avec certains rafinement pour le cas de vecteur.

Auteur

Mechineau Alexandre

5.2.2 Documentation de la fonction/subroutine

5.2.2.1 create_csr_matrix()

```
type (csr_matrix) function csr::create_csr_matrix (
             integer, intent(in) m,
             integer, intent(in) n )
```

Cette fonction crée une matrice de taille quelconque.

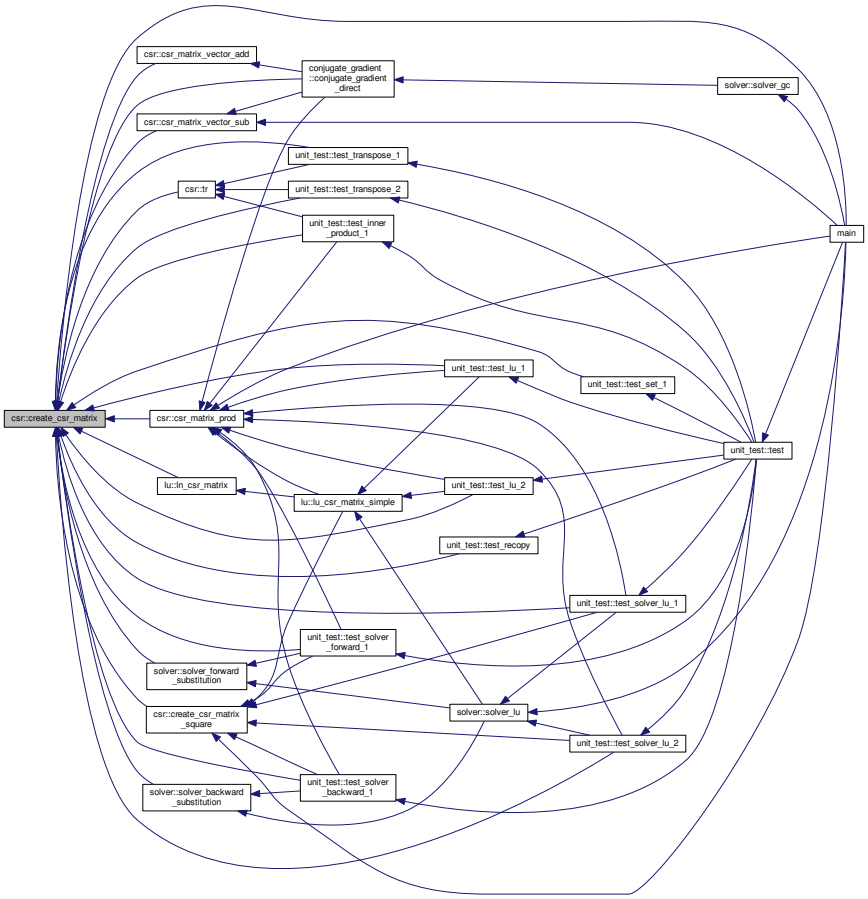
Paramètres

m	Nombre de ligne de la matrice
n	Nombre de colonne de la matrice

Renvoie

Matrice CSR de taille $m * n$

Voici le graphe des appelants de cette fonction :



5.2.2.2 create_csr_matrix_from_file()

```
type (csr_matrix) function csr::create_csr_matrix_from_file (
             character(len=*), intent(in) f )
```

Cette fonction permet de créer une matrice depuis un fichier. La matrice dans le fichier doit utilisé le style C-CSR modifié pour que les colonnes commence par être indexé par 1. **On admet que le fichier passé est valide**

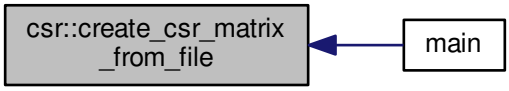
Paramètres

f	Fichier stockant une matrice CSR

Renvoie

Matrice CSR créé depuis le fichier

Voici le graphe des appelants de cette fonction :



5.2.2.3 create_csr_matrix_recopy()

```
type (csr_matrix) function csr::create_csr_matrix_recopy (
             type (csr_matrix), intent(in) M )
```

Cette fonction permet de dupliquer une matrice.

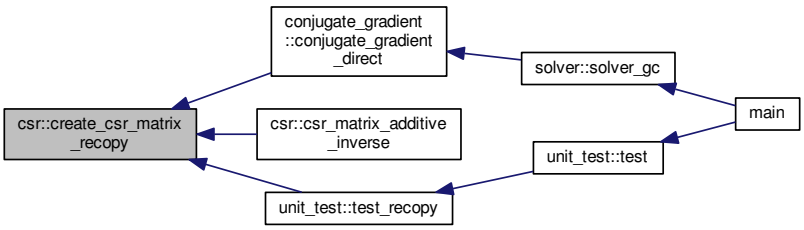
Paramètres

M	Une matrice CSR que l'on souhaite recopier

Renvoie

Une nouvelle matrice CSR avec les mêmes valeurs que la matrice passée en argument

Voici le graphe des appelants de cette fonction :



5.2.2.4 create_csr_matrix_square()

```
type (csr_matrix) function csr::create_csr_matrix_square (
            integer n )
```

Cette fonction permet de créer une matrice carrée vide.

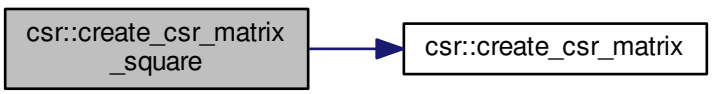
Paramètres

n	Taille d'un coté d'une matrice carrée

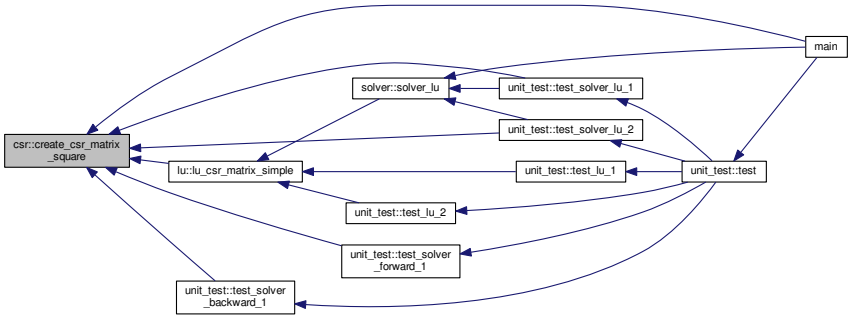
Renvoie

Matrice carrée de taille $n * n$

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.5 csr_matrix_additive_inverse()

```
type(csr_matrix) function csr::csr_matrix_additive_inverse (
            class(csr_matrix), intent(in) A )
```

Cette fonction calcule l'inverse associé à la loi additive d'une matrice.

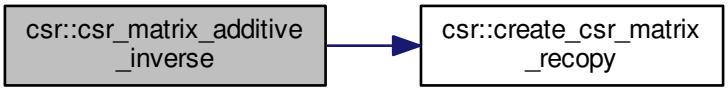
Paramètres

A	Matrice CSR

Renvoie

$-A$

Voici le graphe d'appel pour cette fonction :



5.2.2.6 csr_matrix_equal()

```
logical function csr::csr_matrix_equal (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Cette fonction permet de vérifier si deux matrices sont égales.

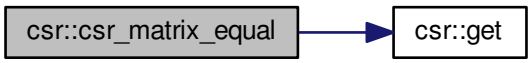
Paramètres

A	Matrice CSR
B	Matrice CSR

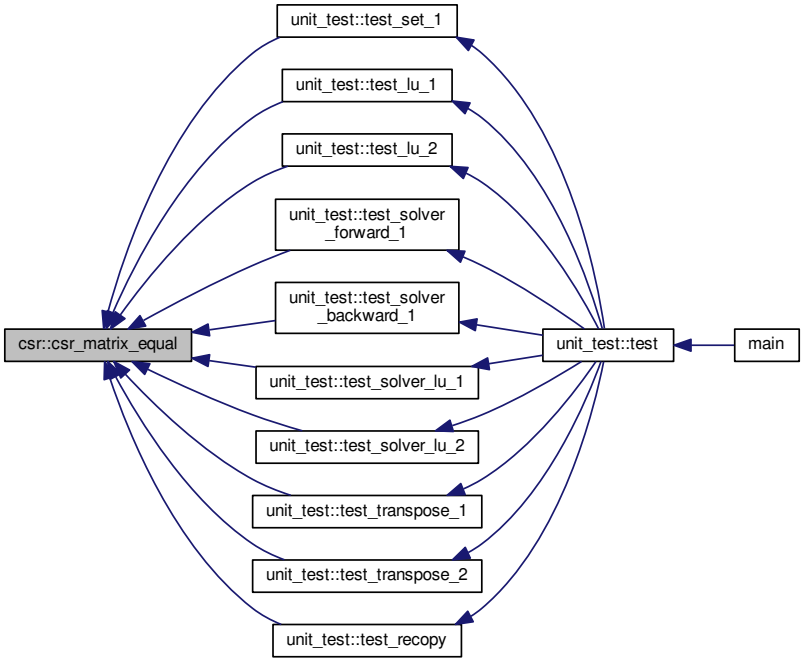
Renvoie

$A = B$

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.7 csr_matrix_inner_product()

```
real(kind=selected_real_kind(15, 307)) function csr::csr_matrix_inner_product (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Cette fonction calcule le produit scalaire de deux vecteurs.

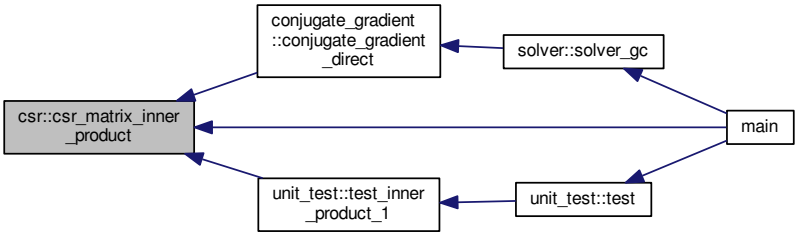
Paramètres

A	Matrice CSR définissant un vecteur colonne
B	Matrice CSR définissant un vecteur colonne

Renvoie

$\langle A|B\rangle$

Voici le graphe des appelants de cette fonction :



5.2.2.8 csr_matrix_prod()

```
type(csr_matrix) function csr::csr_matrix_prod (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Cette fonction calcule le produit de deux matrices.

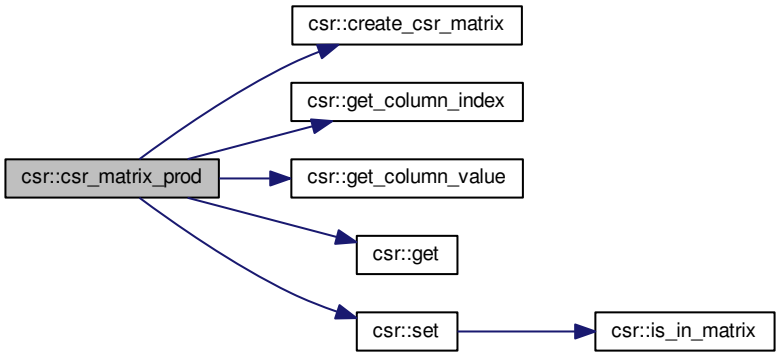
Paramètres

A	Matrice CSR
B	Matrice CSR

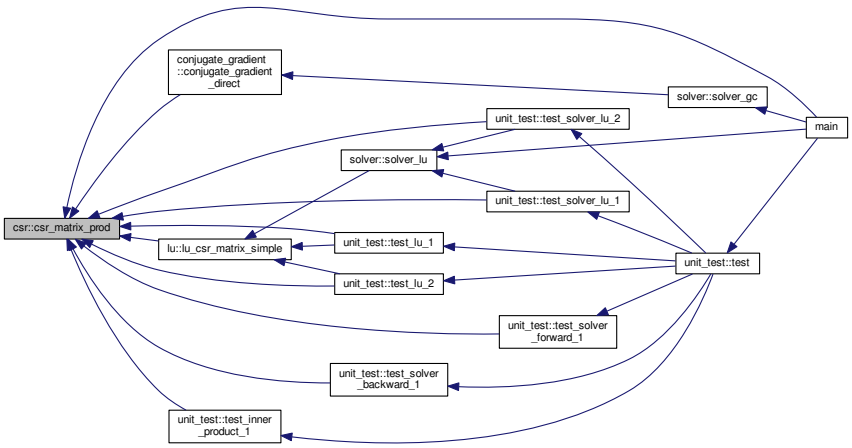
Renvoie

$A * B$

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.9 csr_matrix_save_to_file()

```
subroutine csr::csr_matrix_save_to_file (
            class (csr_matrix) M,
            character(len=*), intent(in) f )
```

Cette subroutine permet de sauvegarder une matrice CSR dans un fichier.

Paramètres

M	Matrice CSR à sauver dans un fichier
f	Fichier de sortier

5.2.2.10 csr_matrix_scalar_prod()

```
type(csr_matrix) function csr::csr_matrix_scalar_prod (
            class(csr_matrix), intent(in) M,
            real(kind=selected_real_kind(15, 307)), intent(in) K )
```

Cette fonction calcule le produit entre un réel et une matrice.

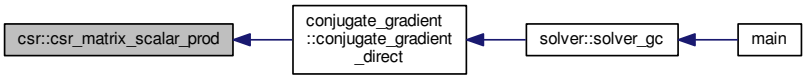
Paramètres

M	Matrice CSR
K	un réel

Renvoie

$k * M$

Voici le graphe des appelants de cette fonction :



5.2.2.11 csr_matrix_vector_add()

```
type(csr_matrix) function csr::csr_matrix_vector_add (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Cette fonction calcule la somme de deux vecteurs colonnes.

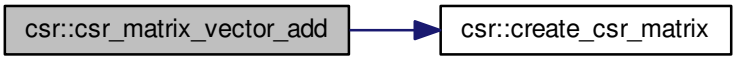
Paramètres

A	Matrice CSR définissant un vecteur colonne
B	Matrice CSR définissant un vecteur colonne

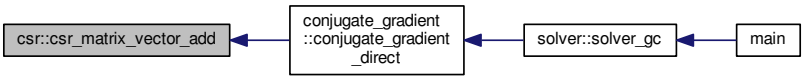
Renvoie

$A + B$

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.12 csr_matrix_vector_sub()

```
type(csr_matrix) function csr::csr_matrix_vector_sub (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Cette fonction calcule la différence de deux vecteurs colonnes.

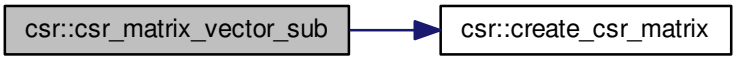
Paramètres

A	Matrice CSR définissant un vecteur colonne
B	Matrice CSR définissant un vecteur colonne

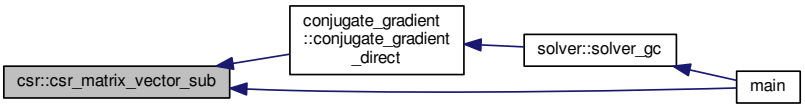
Renvoie

$A - B$

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.13 display()

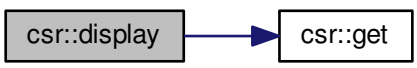
```
subroutine csr::display (
            class(csr_matrix), intent(in) M )
```

Cette subroutine affiche une matrice CSR.

Paramètres

M	Matrice à afficher

Voici le graphe d'appel pour cette fonction :



5.2.2.14 display_debug()

```
subroutine csr::display_debug (
            class(csr_matrix), intent(in) M )
```

Cette subroutine affiche une matrice CSR avec des informations avancées.

Paramètres

M	Matrice à afficher

5.2.2.15 free_csr_matrix()

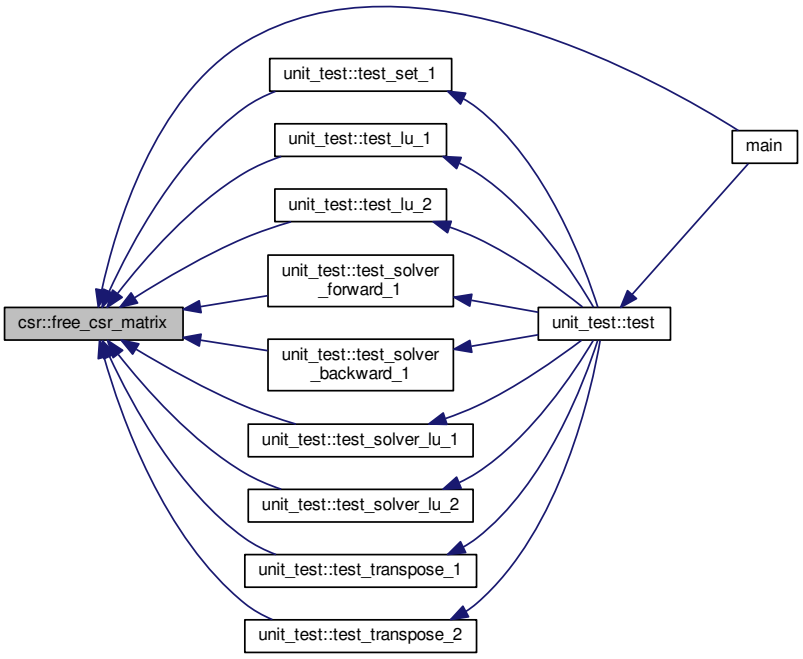
```
subroutine csr::free_csr_matrix (
            type(csr_matrix) M )
```

Cette subroutine permet de détruire une matrice CSR.

Paramètres

M	Matrice à désallouer

Voici le graphe des appelants de cette fonction :



5.2.2.16 get()

```
real(kind=selected_real_kind(15, 307)) function csr::get (
            class(csr_matrix), intent(in) M,
            integer, intent(in) row,
            integer, intent(in) col )
```

Cette fonction permet d'obtnir la valeur d'un élément d'une matrice.

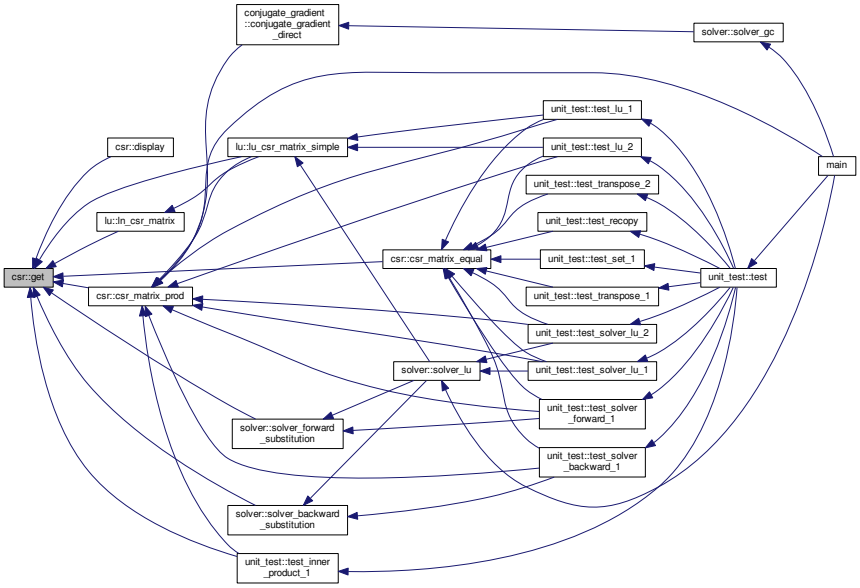
Paramètres

M	Une matrice CSR
row	Ligne de la valeur souhaitée
col	Colonne de la valeur souhaitée

Renvoie

Valeur contenu à la position (row, col). Si La position est invalide retourne 0.

Voici le graphe des appelants de cette fonction :



5.2.2.17 get_column_index()

```
integer function, dimension(:), allocatable csr::get_column_index (
            class(csr_matrix), intent(in) M,
            integer, intent(in) row )
```

Cette fonction permet d'obtenir les colonnes non-nulles d'une ligne donnée.

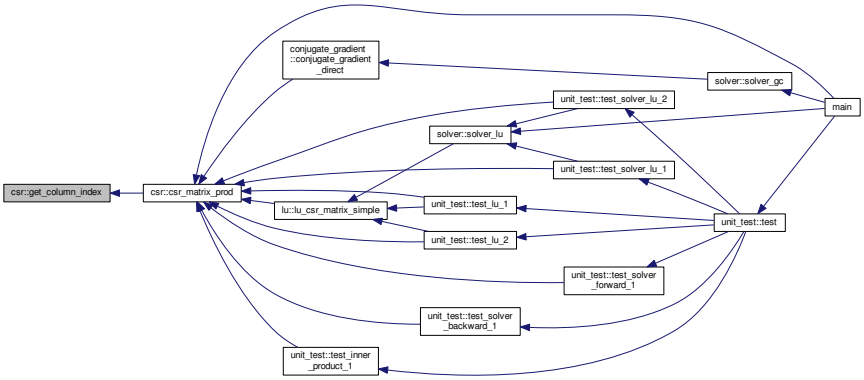
Paramètres

M	Matrice CSR
row	index d'une ligne de la matrice

Renvoie

Ensemble des colonnes à valeur non nulle

Voici le graphe des appelants de cette fonction :



5.2.2.18 get_column_value()

```
real(kind=selected_real_kind(15, 307)) function, dimension(:), allocatable csr::get_column_↩
value (
            class(csr_matrix), intent(in) M,
            integer, intent(in) row )
```

Cette fonction permet d'obtenir les valeurs non-nulles d'une ligne donnée.

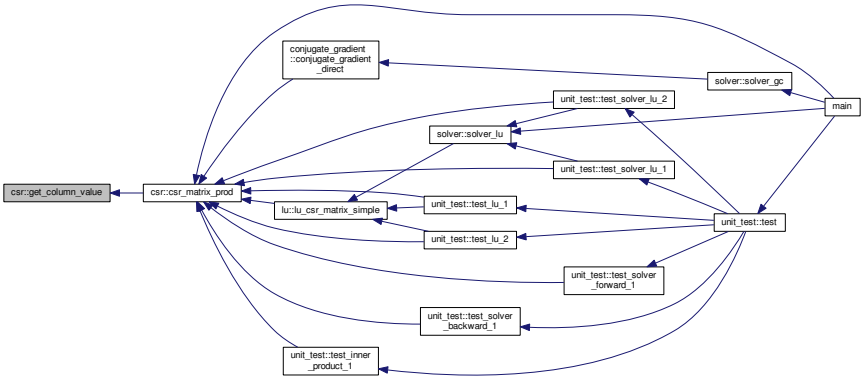
Paramètres

M	Matrice CSR
row	index d'une ligne de la matrice

Renvoie

Ensemble des valeurs non nulle

Voici le graphe des appelants de cette fonction :



5.2.2.19 is_column_vector()

```
logical function csr::is_column_vector (
            class(csr_matrix), intent(in) V )
```

Cette fonction permet de vérifier si une matrice est un vecteur colonne.

Paramètres

V	Une matrice CSR

Renvoie

Retourne VRAI si la matrice est un vecteur colonne

5.2.2.20 is_in_matrix()

```
logical function csr::is_in_matrix (
            class(csr_matrix), intent(in) M,
            integer, intent(in) row,
            integer, intent(in) col )
```

Cette fonction permet de vérifier que des indices sont bien valide pour une matrice donnée.

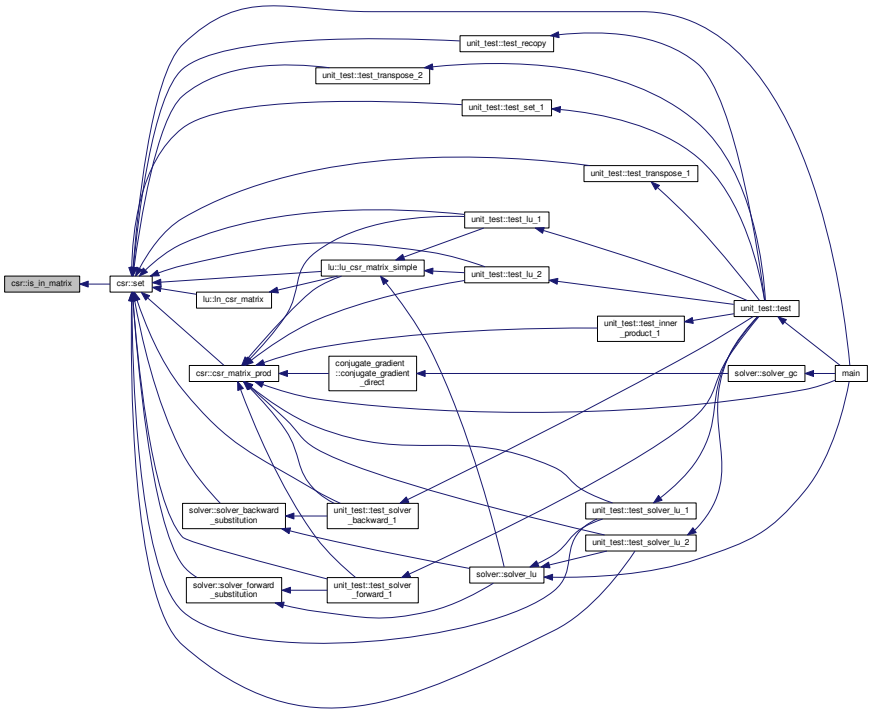
Paramètres

M	Matrice CSR
row	Ligne
col	Colonne

Renvoie

Retourne VRAI si les indices (row, col) sont valides pour la matrice donnée

Voici le graphe des appelants de cette fonction :



5.2.2.21 set()

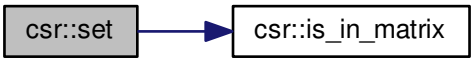
```
subroutine csr::set (
            class(csr_matrix) M,
            integer, intent(in) row,
            integer, intent(in) col,
            real(kind=selected_real_kind(15, 307)), intent(in) val )
```

Cette subroutine permet de modifier une valeur dans une matrice CSR.

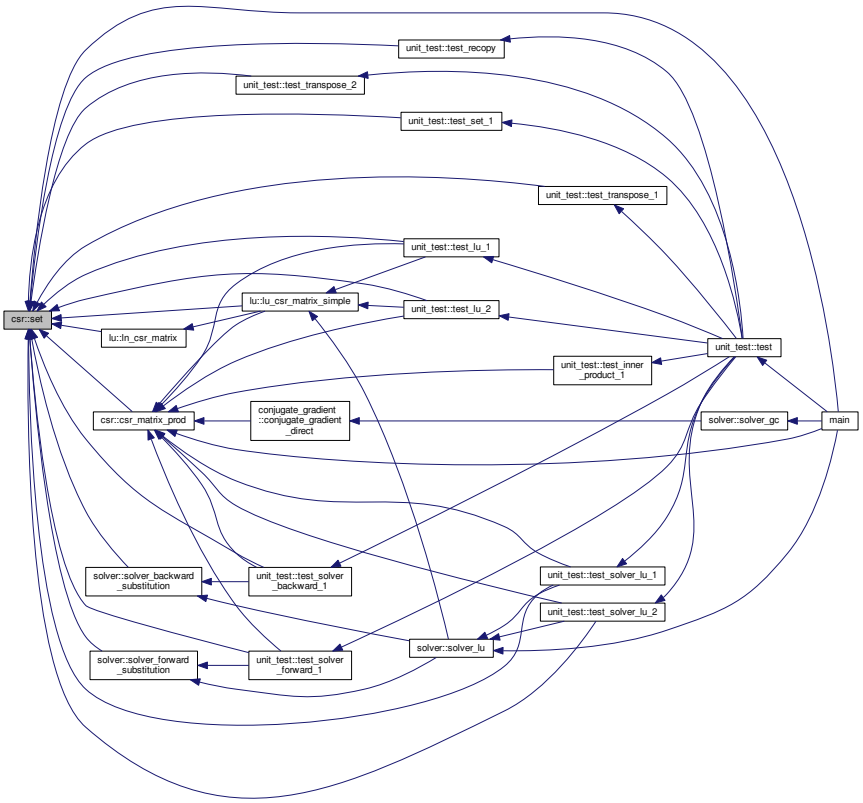
Paramètres

M	Une matrice CSR
row	Ligne de la valeur souhaitée
col	Colonne de la valeur souhaitée
val	Valeur souhaitée

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.2.2.22 tr()

```
type(csr_matrix) function csr::tr (
            class(csr_matrix), intent(in) M )
```

Cette fonction calcule la transposée d'une matrice.

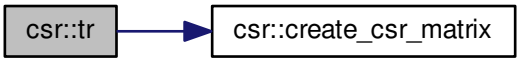
Paramètres

M	Matrice CSR

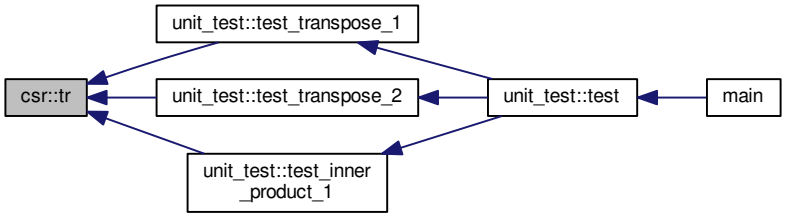
Renvoie

M^T

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.3 Référence du module lu

Ce module permet d'obtenir la décomposition LU d'une matrice.

Fonctions/Subroutines

- type(csr_matrix) function, dimension(2) lu_csr_matrix_simple (M)
  Retourne la décomposition LU d'une matrice selon l'algorithme Doolittle sans permutation.
- type(csr_matrix) function ln_csr_matrix (A, n)
  Fonction intérmédiare dans le calcul de LU.

5.3.1 Description détaillée

Ce module permet d'obtenir la décomposition LU d'une matrice.

Auteur

Mechineau Alexandre

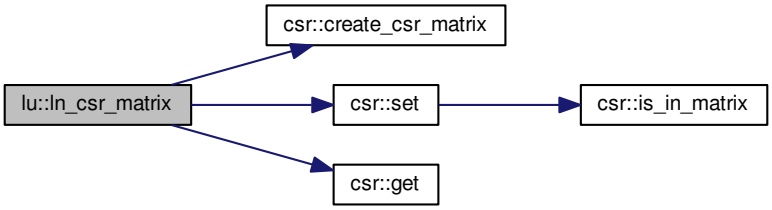
5.3.2 Documentation de la fonction/subroutine

5.3.2.1 ln_csr_matrix()

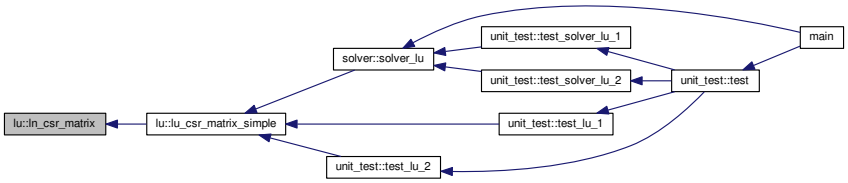
```
type(csr_matrix) function lu::ln_csr_matrix (
            type(csr_matrix), intent(in) A,
            integer, intent(in) n )
```

Fonction intérmédiare dans le calcul de LU.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.3.2.2 lu_csr_matrix_simple()

```
type(csr_matrix) function, dimension(2) lu::lu_csr_matrix_simple (
            type(csr_matrix), intent(in) M )
```

Retourne la décomposition LU d'une matrice selon l'algorithme Doolittle sans permutation.

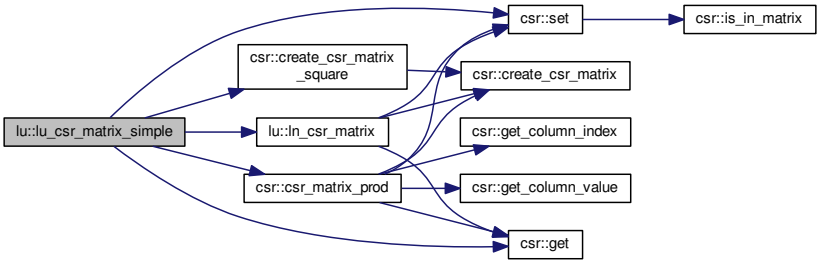
Paramètres

M	Matrice CSR

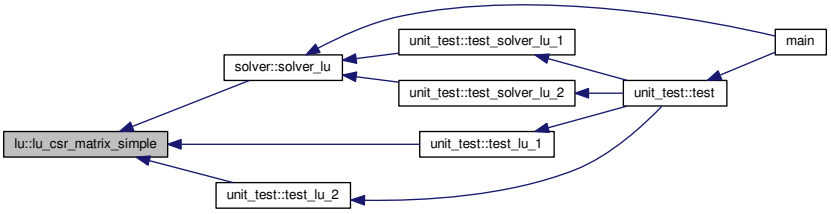
Renvoie

Tableau de 2 matrice CSR (L, U)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.4 Référence du module solver

Ce module permet de résoudre des systèmes linéaires en utilisant soit la décomposition LU, soit la méthode du gradient conjugué.

Fonctions/Subroutines

- type(csr_matrix) function solver_lu (A, B)
  Résouds le système $AX = B$ en utilisant la décomposition LU de A.
- type(csr_matrix) function solver_gc (A, B, eps)
  Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué
- type(csr_matrix) function solver_forward_substitution (L, B)
  Résouds le système $LX = B$ un utilisant la méthode substitution.
- type(csr_matrix) function solver_backward_substitution (L, B)
  Résouds le système $LX = B$ un utilisant la méthode substitution inversée.

5.4.1 Description détaillée

Ce module permet de résoudre des systèmes linéaires en utilisant soit la décomposition LU, soit la méthode du gradient conjugué.

Auteur

Mechineau Alexandre

5.4.2 Documentation de la fonction/subroutine

5.4.2.1 solver_backward_substitution()

```
type(csr_matrix) function solver::solver_backward_substitution (
            type(csr_matrix), intent(in) L,
            type(csr_matrix), intent(in) B )
```

Résouds le système $LX = B$ un utilisant la méthode substitution inversée.

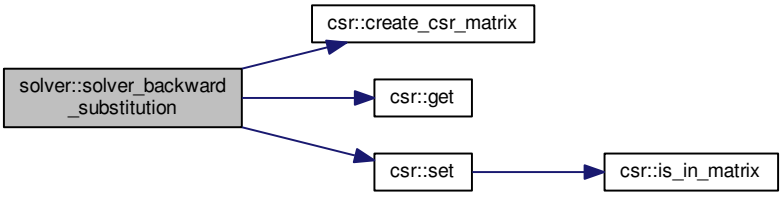
Paramètres

L	Matrice CSR triangulaire supérieur
B	Matrice CSR représentant un vecteur colonne

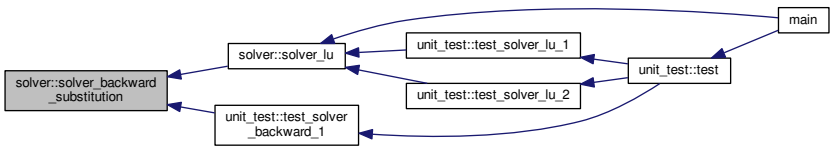
Renvoie

Solution du système

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.4.2.2 solver_forward_substitution()

```
type(csr_matrix) function solver::solver_forward_substitution (
            type(csr_matrix), intent(in) L,
            type(csr_matrix), intent(in) B )
```

Résouds le système $LX = B$ un utilisant la méthode substitution.

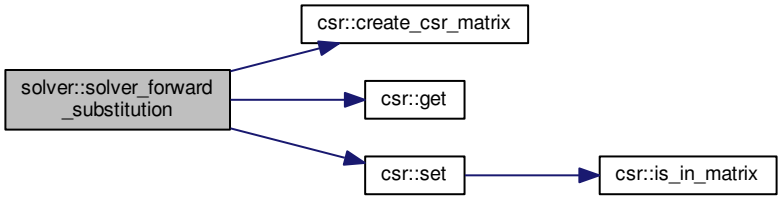
Paramètres

L	Matrice CSR triangulaire inférieur
B	Matrice CSR représentant un vecteur colonne

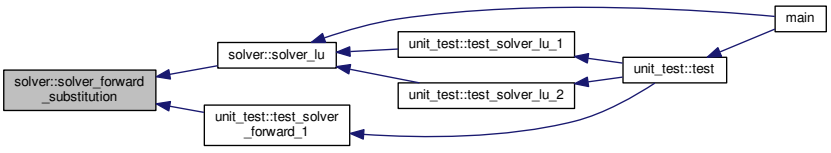
Renvoie

Solution du système

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.4.2.3 solver_gc()

```
type(csr_matrix) function solver::solver_gc (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B,
            real, intent(in), optional eps )
```

Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué

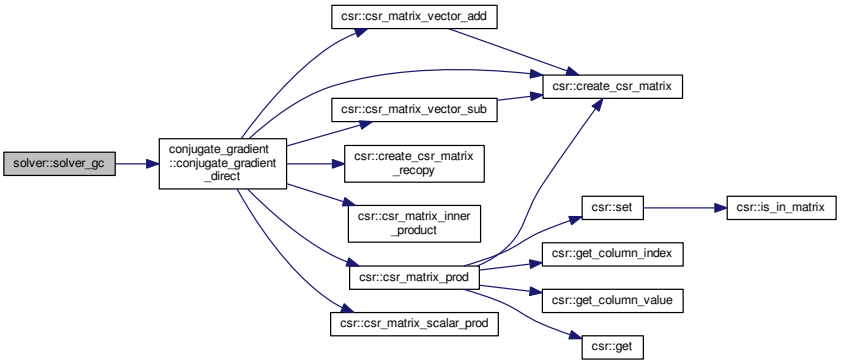
Paramètres

A	Matrice CSR
B	Matrice CSR représentant un vecteur colonne
eps	Précison requise (Optionel)

Renvoie

Approximation, dans le pire des cas, de la solution du système

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

5.4.2.4 solver_lu()

```
type(csr_matrix) function solver::solver_lu (
            type(csr_matrix), intent(in) A,
            type(csr_matrix), intent(in) B )
```

Résouds le système $AX = B$ en utilisant la décomposition LU de A.

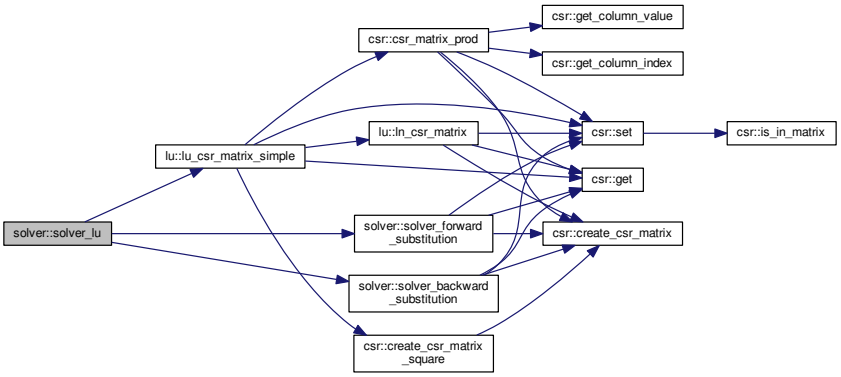
Paramètres

A	Matrice CSR
B	Matrice CSR représentant un vecteur colonne

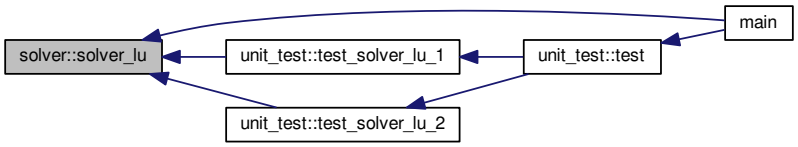
Renvoie

Approximation, dans le pire des cas de la solution du système

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



5.5 Référence du module unit_test

Ce module contient plusieurs test permettant d'assurer le bon fonctionnement des modules CSR, LU, CONJUG↩ ATE_GRADIENT et SOLVER.

Fonctions/Subroutines

- subroutine test ()
  Cette subroutine appelle l'ensemble des tests implémentés et affiche chaqu'un des résultats.
- logical function test_set_1 ()
- logical function test_lu_1 ()
- logical function test_lu_2 ()
- logical function test_solver_forward_1 ()

- logical function test_solver_backward_1 ()
- logical function test_solver_lu_1 ()
- logical function test_solver_lu_2 ()
- logical function test_transpose_1 ()
- logical function test_transpose_2 ()
- logical function test_inner_product_1 ()
- logical function test_recopy ()

5.5.1 Description détaillée

Ce module contient plusieurs test permettant d'assurer le bon fonctionnement des modules CSR, LU, CONJUG↩ ATE_GRADIENT et SOLVER.

Auteur

Mechineau Alexandre

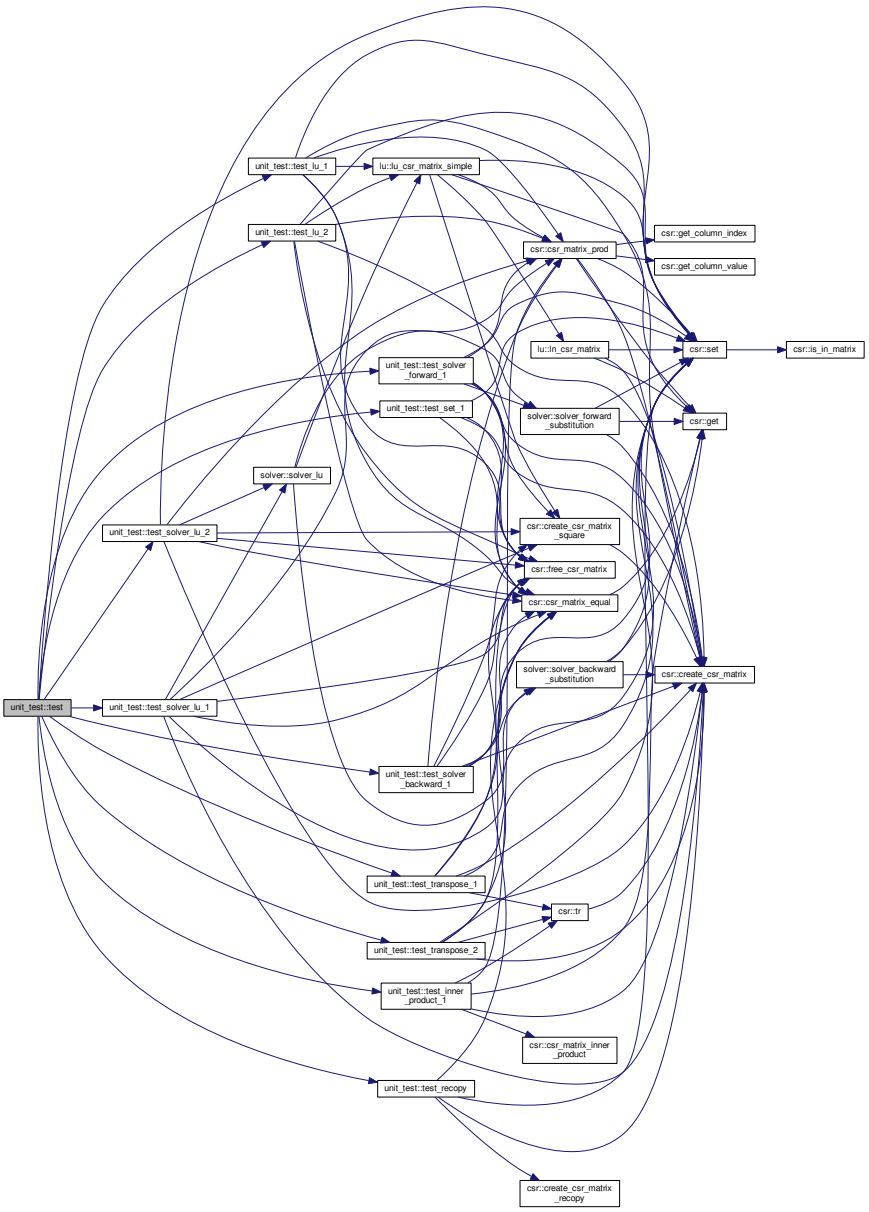
5.5.2 Documentation de la fonction/subroutine

5.5.2.1 test()

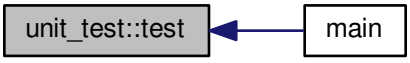
```
subroutine unit_test::test ( )
```

Cette subroutine appelle l'ensemble des tests implémentés et affiche chaqu'un des résultats.

Voici le graphe d'appel pour cette fonction :



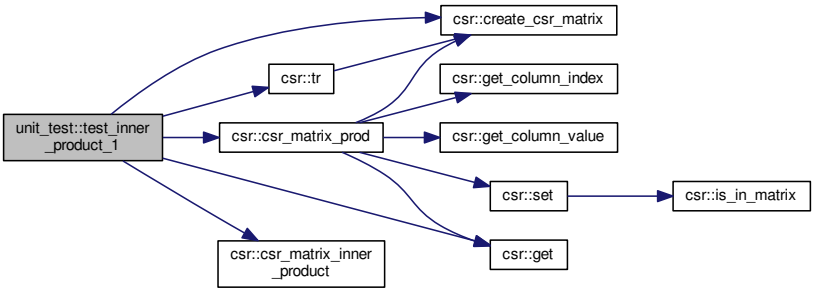
Voici le graphe des appelants de cette fonction :



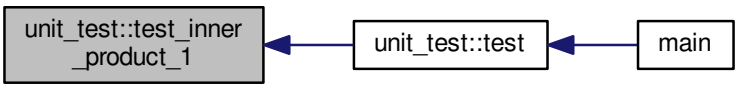
5.5.2.2 test_inner_product_1()

```
logical function unit_test::test_inner_product_1 ( )
```

Voici le graphe d'appel pour cette fonction :



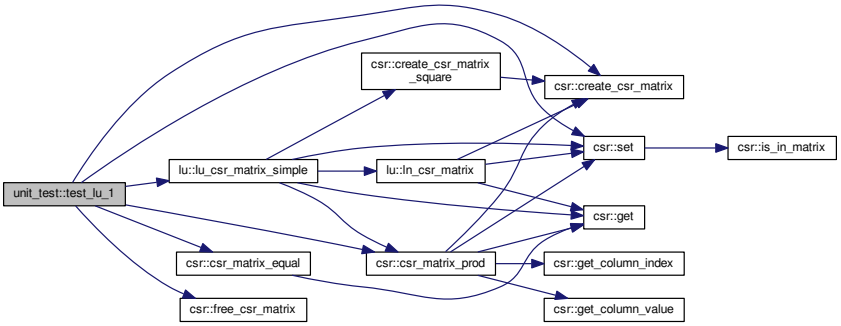
Voici le graphe des appelants de cette fonction :



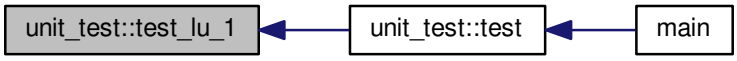
5.5.2.3 test_lu_1()

```
logical function unit_test::test_lu_1 ( )
```

Voici le graphe d'appel pour cette fonction :



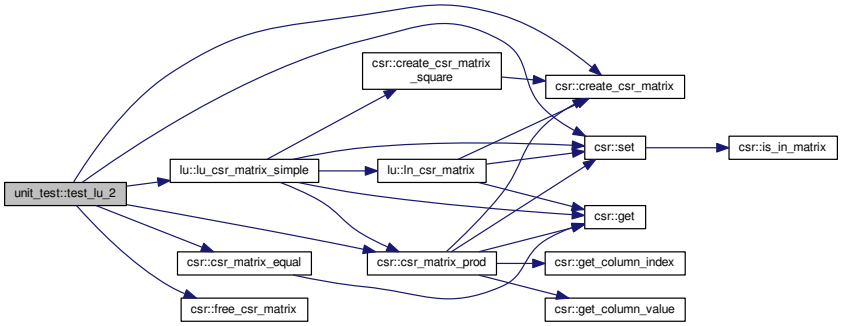
Voici le graphe des appelants de cette fonction :



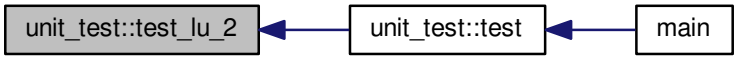
5.5.2.4 test_lu_2()

```
logical function unit_test::test_lu_2 ( )
```

Voici le graphe d'appel pour cette fonction :



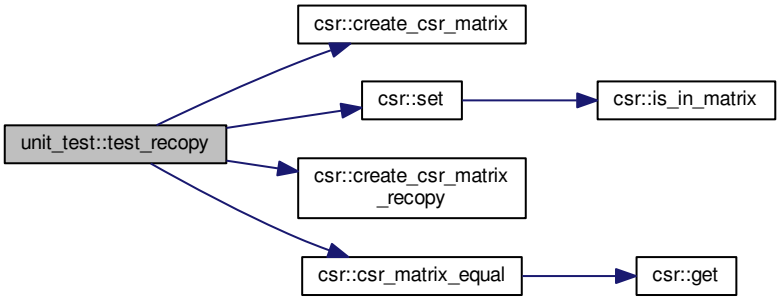
Voici le graphe des appelants de cette fonction :



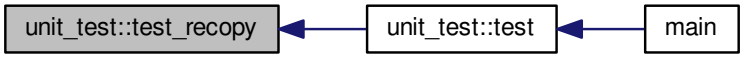
5.5.2.5 test_recopy()

```
logical function unit_test::test_recopy ( )
```

Voici le graphe d'appel pour cette fonction :



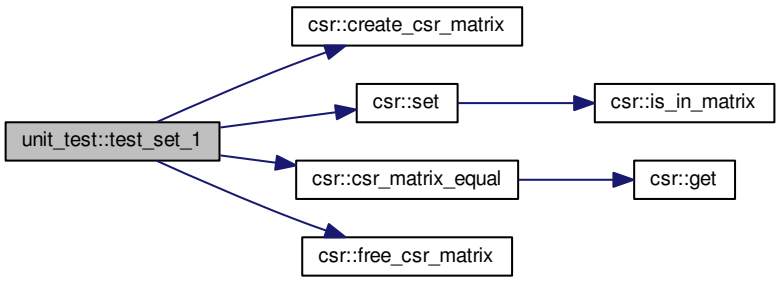
Voici le graphe des appelants de cette fonction :



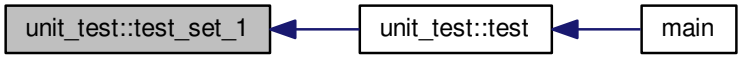
5.5.2.6 test_set_1()

```
logical function unit_test::test_set_1 ( )
```

Voici le graphe d'appel pour cette fonction :



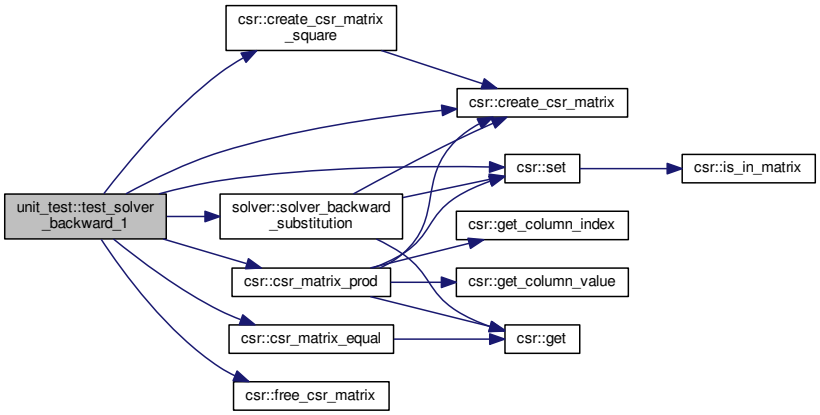
Voici le graphe des appelants de cette fonction :



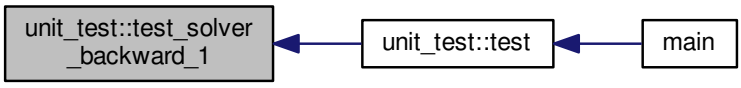
5.5.2.7 test_solver_backward_1()

```
logical function unit_test::test_solver_backward_1 ( )
```

Voici le graphe d'appel pour cette fonction :



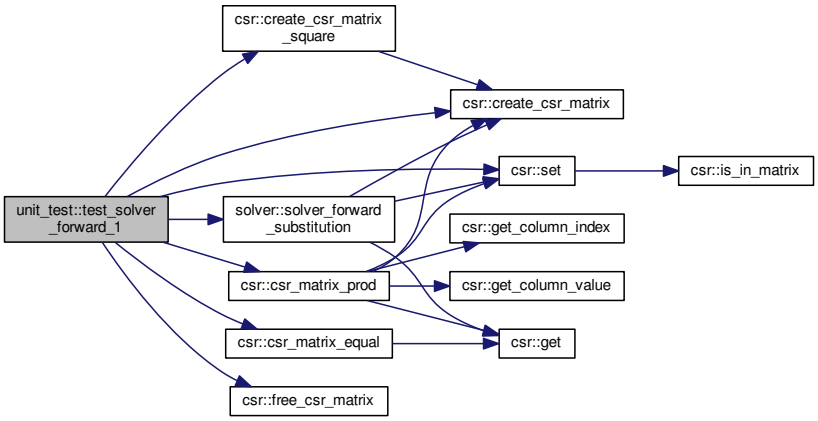
Voici le graphe des appelants de cette fonction :



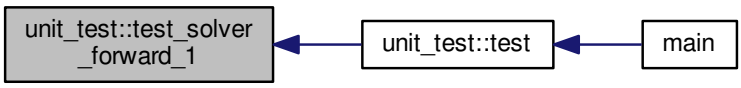
5.5.2.8 test_solver_forward_1()

```
logical function unit_test::test_solver_forward_1 ( )
```

Voici le graphe d'appel pour cette fonction :



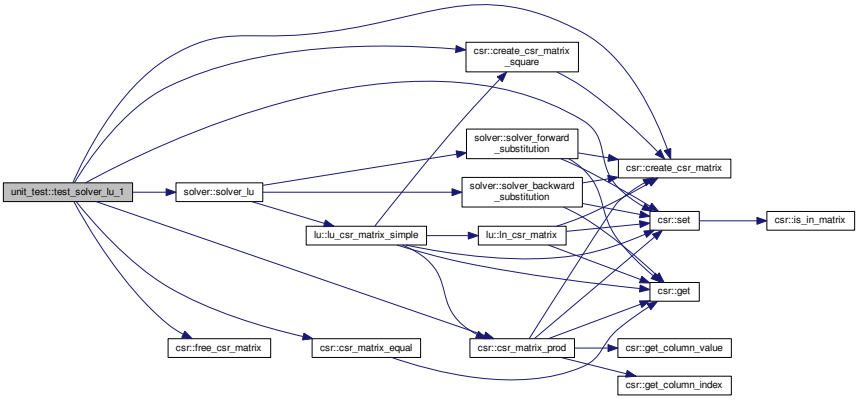
Voici le graphe des appelants de cette fonction :



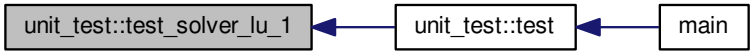
5.5.2.9 test_solver_lu_1()

```
logical function unit_test::test_solver_lu_1 ( )
```

Voici le graphe d'appel pour cette fonction :



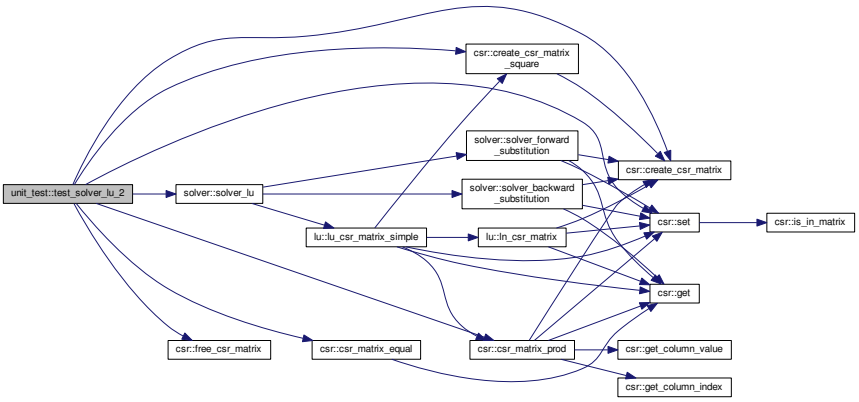
Voici le graphe des appelants de cette fonction :



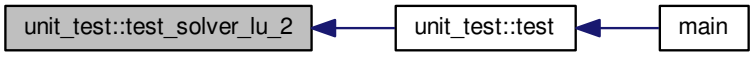
5.5.2.10 test_solver_lu_2()

```
logical function unit_test::test_solver_lu_2 ( )
```

Voici le graphe d'appel pour cette fonction :



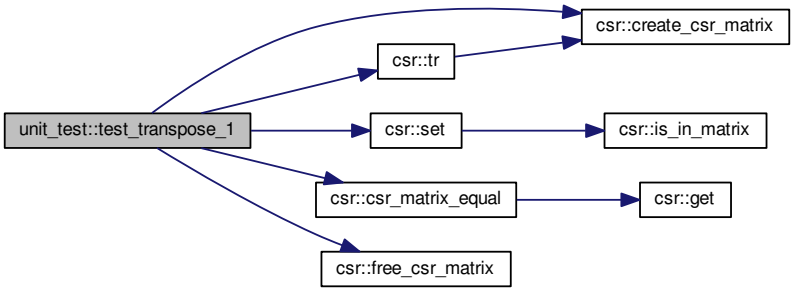
Voici le graphe des appelants de cette fonction :



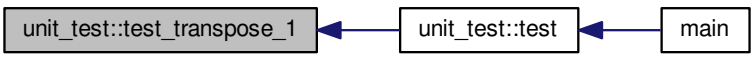
5.5.2.11 test_transpose_1()

```
logical function unit_test::test_transpose_1 ( )
```

Voici le graphe d'appel pour cette fonction :



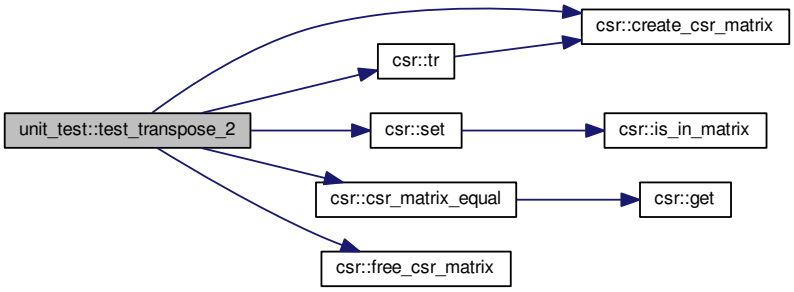
Voici le graphe des appelants de cette fonction :



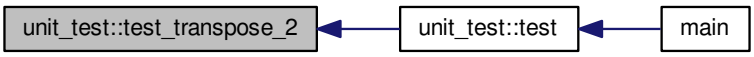
5.5.2.12 test_transpose_2()

```
logical function unit_test::test_transpose_2 ( )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



Chapitre 6

Documentation du type de données

6.1 Réference du type csr : :csr_matrix

Type décrivant une matrice CSR Attention m_col stocke l'index des colonnes en commencant par 1. [CSR].

Fonctions membres publiques

- PROCEDURE is_column_vector
- tr
- csr_matrix_scalar_prod
- is_in_matrix
- get
- get_column_index
- get_column_value
- set
- display
- display_debug
- csr_matrix_additive_inverse
- csr_matrix_save_to_file

Attributs publics

- integer, dimension(:), allocatable m_rowidx
  Tableau stockant l'index des lignes.
- integer, dimension(:), allocatable m_col
  Tableau stockant les colonnes non nulles.
- real(kind=selected_real_kind(15, 307)), dimension(:), allocatable m_val
  Tableau stockant les valeurs non nulles.
- integer m_m
  Nombre de colonnes de la matrice.
- integer m_n
  Nombre de lignes de la matrice.

6.1.1 Description détaillée

Type décrivant une matrice CSR Attention m_col stocke l'index des colonnes en commencant par 1. [CSR].

6.1.2 Documentation des fonctions/subroutines membres

6.1.2.1 csr_matrix_additive_inverse()

```
csr::csr_matrix::csr_matrix_additive_inverse ( )
```

6.1.2.2 csr_matrix_save_to_file()

```
csr::csr_matrix::csr_matrix_save_to_file ( )
```

6.1.2.3 csr_matrix_scalar_prod()

```
csr::csr_matrix::csr_matrix_scalar_prod ( )
```

6.1.2.4 display()

```
csr::csr_matrix::display ( )
```

6.1.2.5 display_debug()

```
csr::csr_matrix::display_debug ( )
```

6.1.2.6 get()

```
csr::csr_matrix::get ( )
```

6.1.2.7 get_column_index()

```
csr::csr_matrix::get_column_index ( )
```

6.1.2.8 get_column_value()

```
csr::csr_matrix::get_column_value ( )
```

6.1.2.9 is_column_vector()

```
PROCEDURE csr::csr_matrix::is_column_vector ( )
```

6.1.2.10 is_in_matrix()

```
csr::csr_matrix::is_in_matrix ( )
```

6.1.2.11 set()

```
csr::csr_matrix::set ( )
```

6.1.2.12 tr()

```
csr::csr_matrix::tr ( )
```

6.1.3 Documentation des données membres

6.1.3.1 m_col

```
integer, dimension(:), allocatable csr::csr_matrix::m_col
```

Tableau stockant les colonnes non nulles.

6.1.3.2 m_m

```
integer csr::csr_matrix::m_m
```

Nombre de colonnes de la matrice.

6.1.3.3 m_n

```
integer csr::csr_matrix::m_n
```

Nombre de lignes de la matrice.

6.1.3.4 m_rowidx

```
integer, dimension(:), allocatable csr::csr_matrix::m_rowidx
```

Tableau stockant l'index des lignes.

6.1.3.5 m_val

```
real(kind=selected_real_kind(15, 307)), dimension(:), allocatable csr::csr_matrix::m_val
```

Tableau stockant les valeurs non nulles.

La documentation de ce type a été générée à partir du fichier suivant :
— Source/CSR.f90

Chapitre 7

Documentation des fichiers

7.1 Référence du fichier Source/CONJUGATE_GRADIENT.f90

Modules

— module conjugate_gradient
Ce module permet d'utiliser l'algorithme du gradient conjugué. Deux implémentations sont fournies, la première effectue le nombre maximal d'étape, la seconde s'arrête à une précision donnée.

Fonctions/Subroutines

— type(csr_matrix) function conjugate_gradient : :conjugate_gradient_direct (A, B, eps)
Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué Source https://www.ljll.↩ math.upmc.fr/hecht/ftp/InfoSci/doc-pdf/Master_book_GC.pdf.

7.2 Référence du fichier Source/CSR.f90

Les types de données

— type csr : :csr_matrix
Type décrivant une matrice CSR Attention m_col stocke l'index des colonnes en commencant par 1. [CSR].

Modules

— module csr
Ce module permet de créer,stocker et manipuler une matrice sous la forme CSR. Plusieurs opérations courantes sont implémentés, avec certains rafinement pour le cas de vecteur.

Fonctions/Subroutines

- type(csr_matrix) function csr : :create_csr_matrix_from_file (f)
  Cette fonction permet de créer une matrice depuis un fichier. La matrice dans le fichier doit utilisé le style C-CSR modifié pour que les colonnes commence par être indexé par 1. ***On admet que le fichier passé est valide***
- subroutine csr : :csr_matrix_save_to_file (M, f)
  Cette subroutine permet de sauvegarder une matrice CSR dans un fichier.
- type(csr_matrix) function csr : :create_csr_matrix_square (n)
  Cette fonction permet de créer une matrice carrée vide.
- type(csr_matrix) function csr : :create_csr_matrix_recopy (M)
  Cette fonction permet de dupliquer une matrice.
- type(csr_matrix) function csr : :create_csr_matrix (m, n)
  Cette fonction crée une matrice de taille quelconque.
- subroutine csr : :free_csr_matrix (M)
  Cette subroutine permet de détruire une matrice CSR.
- logical function csr : :is_in_matrix (M, row, col)
  Cette fonction permet de vérifier que des indices sont bien valide pour une matrice donnée.
- logical function csr : :is_column_vector (V)
  Cette fonction permet de vérifier si une matrice est un vecteur colonne.
- real(kind=selected_real_kind(15, 307)) function csr : :get (M, row, col)
  Cette fonction permet d'obtnir la valeur d'un élément d'une matrice.
- subroutine csr : :set (M, row, col, val)
  Cette subroutine permet de modifier une valeur dans une matrice CSR.
- subroutine csr : :display_debug (M)
  Cette subroutine affiche une matrice CSR avec des informations avancées.
- subroutine csr : :display (M)
  Cette subroutine affiche une matrice CSR.
- integer function, dimension(:), allocatable csr : :get_column_index (M, row)
  Cette fonction permet d'obtenir les colonnes non-nulles d'une ligne donnée.
- real(kind=selected_real_kind(15, 307)) function, dimension(:), allocatable csr : :get_column_value (M, row)
  Cette fonction permet d'obtenir les valeurs non-nulles d'une ligne donnée.
- type(csr_matrix) function csr : :csr_matrix_prod (A, B)
  Cette fonction calcule le produit de deux matrices.
- logical function csr : :csr_matrix_equal (A, B)
  Cette fonction permet de vérifier si deux matrices sont égales.
- type(csr_matrix) function csr : :tr (M)
  Cette fonction calcule la transposée d'une matrice.
- real(kind=selected_real_kind(15, 307)) function csr : :csr_matrix_inner_product (A, B)
  Cette fonction calcule le produit scalaire de deux vecteurs.
- type(csr_matrix) function csr : :csr_matrix_vector_add (A, B)
  Cette fonction calcule la somme de deux vecteurs colonnes.
- type(csr_matrix) function csr : :csr_matrix_vector_sub (A, B)
  Cette fonction calcule la différence de deux vecteurs colonnes.
- type(csr_matrix) function csr : :csr_matrix_additive_inverse (A)
  Cette fonction calcule l'inverse associé à la loi additive d'une matrice.
- type(csr_matrix) function csr : :csr_matrix_scalar_prod (M, K)
  Cette fonction calcule le produit entre un réel et une matrice.

7.3 Référence du fichier Source/LU.f90

Modules

- module lu
  Ce module permet d'obtenir la décomposition LU d'une matrice.

Fonctions/Subroutines

- type(csr_matrix) function, dimension(2) lu : :lu_csr_matrix_simple (M)
  Retourne la décomposition LU d'une matrice selon l'algorithme Doolittle sans permutation.
- type(csr_matrix) function lu : :ln_csr_matrix (A, n)
  Fonction intérmédiare dans le calcul de LU.

7.4 Référence du fichier Source/Projet.f90

Fonctions/Subroutines

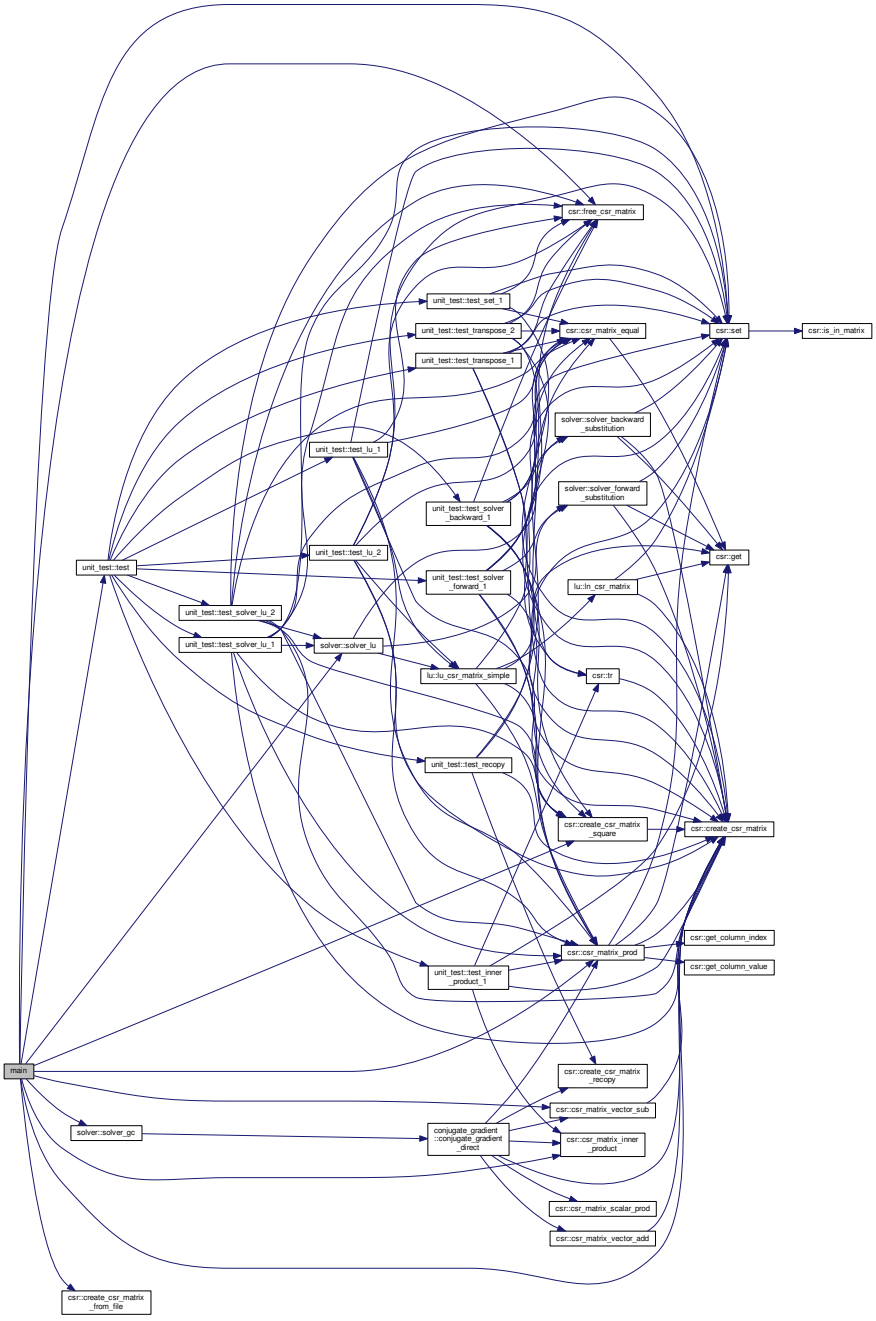
— program main

7.4.1 Documentation de la fonction/subroutine

7.4.1.1 main()

```
program main ( )
```

Voici le graphe d'appel pour cette fonction :



7.5 Référence du fichier Source/SOLVER.f90

Modules

— module solver
Ce module permet de résoudre des systèmes linéaires en utilisant soit la décomposition LU, soit la méthode du gradient conjugué.

Fonctions/Subroutines

- type(csr_matrix) function solver : :solver_lu (A, B)
  Résouds le système $AX = B$ en utilisant la décomposition LU de A.
- type(csr_matrix) function solver : :solver_gc (A, B, eps)
  Résouds le système $AX = B$ en utilisant la méthode du gradiant conjugué
- type(csr_matrix) function solver : :solver_forward_substitution (L, B)
  Résouds le système $LX = B$ un utilisant la méthode substitution.
- type(csr_matrix) function solver : :solver_backward_substitution (L, B)
  Résouds le système $LX = B$ un utilisant la méthode substitution inversée.

7.6 Référence du fichier Source/UNIT_TEST.f90

Modules

- module unit_test
  Ce module contient plusieurs test permettant d'assurer le bon fonctionnement des modules CSR, LU, CONJUGATE_GRADIENT et SOLVER.

Fonctions/Subroutines

- subroutine unit_test : :test ()
  Cette subroutine appelle l'ensemble des tests implémentés et affiche chaqu'un des résultats.
- logical function unit_test : :test_set_1 ()
- logical function unit_test : :test_lu_1 ()
- logical function unit_test : :test_lu_2 ()
- logical function unit_test : :test_solver_forward_1 ()
- logical function unit_test : :test_solver_backward_1 ()
- logical function unit_test : :test_solver_lu_1 ()
- logical function unit_test : :test_solver_lu_2 ()
- logical function unit_test : :test_transpose_1 ()
- logical function unit_test : :test_transpose_2 ()
- logical function unit_test : :test_inner_product_1 ()
- logical function unit_test : :test_recopy ()

Index