

Documentation: Interpolation polynomiale et Approximation

Généré par Doxygen 1.9.1

1 Index des espaces de nommage	1
1.1 Liste des espaces de nommage	1
2 Index des fichiers	3
2.1 Liste des fichiers	3
3 Documentation des espaces de nommage	5
3.1 Référence de l'espace de nommage trace	5
3.1.1 Documentation des fonctions	5
3.1.1.1 trace()	5
3.2 Référence de l'espace de nommage trace_approx	5
3.2.1 Documentation des fonctions	5
3.2.1.1 trace_approx()	6
4 Documentation des fichiers	7
4.1 Référence du fichier approximation.c	7
4.1.1 Description détaillée	8
4.1.2 Documentation des fonctions	8
4.1.2.1 approximation()	8
4.1.2.2 approximation_exp()	9
4.1.2.3 carre()	10
4.1.2.4 copy_tab()	11
4.1.2.5 libere_tab_1d()	11
4.1.2.6 moyenne()	12
4.1.2.7 produit_tab()	13
4.1.2.8 tab_ln()	14
4.2 Référence du fichier approximation.h	14
4.2.1 Description détaillée	15
4.2.2 Documentation des fonctions	15
4.2.2.1 approximation()	15
4.2.2.2 approximation_exp()	16
4.2.2.3 carre()	17
4.2.2.4 libere_tab_1d()	18
4.2.2.5 moyenne()	18
4.2.2.6 produit_tab()	20
4.2.2.7 tab_ln()	21
4.3 Référence du fichier gestionFichier.c	21
4.3.1 Documentation des fonctions	22
4.3.1.1 ecriture_points()	22
4.3.1.2 ecriture_points_approx()	23
4.3.1.3 fermetureFichier()	24
4.3.1.4 freeMat()	25
4.3.1.5 lecture_colonne()	25

4.3.1.6 lecture_points()	26
4.3.1.7 liberation_points()	27
4.3.1.8 matrice()	28
4.3.1.9 ouvertureFichier()	28
4.4 Référence du fichier gestionFichier.h	29
4.4.1 Description détaillée	30
4.4.2 Documentation des macros	31
4.4.2.1 NOMBRES_POINTS_1	31
4.4.2.2 NOMBRES_POINTS_2	31
4.4.2.3 NOMBRES_POINTS_3	31
4.4.2.4 NOMBRES_POINTS_4	31
4.4.3 Documentation des fonctions	31
4.4.3.1 ecriture_points()	31
4.4.3.2 ecriture_points_approx()	32
4.4.3.3 fermetureFichier()	33
4.4.3.4 freeMat()	34
4.4.3.5 lecture_colonne()	34
4.4.3.6 lecture_points()	35
4.4.3.7 matrice()	36
4.4.3.8 ouvertureFichier()	37
4.5 Référence du fichier interpolation.c	38
4.5.1 Description détaillée	39
4.5.2 Documentation des fonctions	39
4.5.2.1 ecriture_polynome_L()	39
4.5.2.2 ecriture_polynome_N()	40
4.5.2.3 methode_lagrange()	41
4.5.2.4 methode_neville()	42
4.6 Référence du fichier interpolation.h	42
4.6.1 Description détaillée	43
4.6.2 Documentation des fonctions	44
4.6.2.1 affiche_polynome()	44
4.6.2.2 cree_polynome()	44
4.6.2.3 ecriture_polynome_L()	44
4.6.2.4 ecriture_polynome_N()	45
4.6.2.5 methode_lagrange()	46
4.6.2.6 methode_neville()	47
4.6.2.7 produit_polynomes()	47
4.7 Référence du fichier main.c	48
4.7.1 Description détaillée	48
4.7.2 Documentation des fonctions	48
4.7.2.1 main()	49
4.8 Référence du fichier TRACE PYTHON/trace.py	49

4.9 Référence du fichier TRACE PYTHON/trace_approx.py	49
Index	51

Chapitre 1

Index des espaces de nommage

1.1 Liste des espaces de nommage

Liste de tous les espaces de nommage avec une brève description:

trace	5
trace_approx	5

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

approximation.c	Contients les fonctions pour l'approximation via la méthode de regression	7
approximation.h	Contients les fonctions pour l'approximation via la méthode de regression	14
gestionFichier.c	21
gestionFichier.h	Contients les fonctions de gestions des fichiers	29
interpolation.c	Contients les portotypes desfocntions de gestions des fichiers	38
interpolation.h	Contients les portotypes des fonctions pour l'interpolation	42
main.c	Contients l'appel au différente focntion d'interpolation et de regression	48
TRACE PYTHON/ trace.py	49
TRACE PYTHON/ trace_approx.py	49

Chapitre 3

Documentation des espaces de nommage

3.1 Référence de l'espace de nommage trace

Fonctions

— def `trace` (fichier, fichierP, choix)

3.1.1 Documentation des fonctions

3.1.1.1 `trace()`

```
def trace.trace (
    fichier,
    fichierP,
    choix )
```

Fonction pour tracer les polynomes, fichier correspond au fichier des x, f(x) et fichierP contient les poitns

Définition à la ligne 6 du fichier trace.py.

3.2 Référence de l'espace de nommage trace_approx

Fonctions

— def `trace_approx` (fichier, fichierP, choix)

3.2.1 Documentation des fonctions

3.2.1.1 `trace_approx()`

```
def trace_approx.trace_approx (
    fichier,
    fichierP,
    choix )
```

Fonction pour tracer les polynomes, `fichier` correspond au fichier des `x`, `f(x)` et `fichierP` contient les poitns

Définition à la ligne 5 du fichier `trace_approx.py`.

Chapitre 4

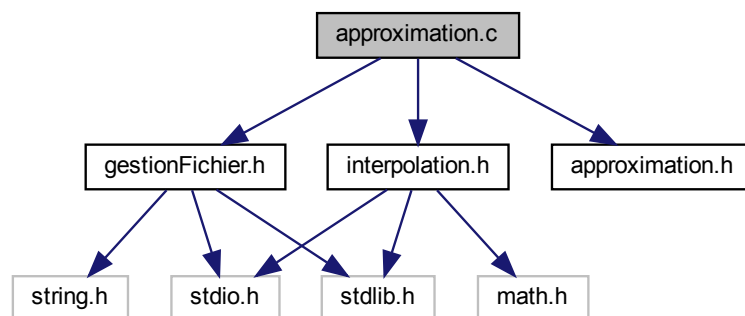
Documentation des fichiers

4.1 Référence du fichier approximation.c

Contient les fonctions pour l'approximation via la méthode de regression.

```
#include "interpolation.h"
#include "gestionFichier.h"
#include "approximation.h"
```

Graphe des dépendances par inclusion de approximation.c:



Fonctions

- float **carre** (float x)
Permet de calculer un flottant au carre.
- void **libere_tab_1d** (float *tab)
Procédure permettant de liberer tableau 1D.
- void **tab_in** (float *tab, int n)
Procédure permettant de remplacer chaque case du tableau par sa composé avec ln.
- float **moyenne** (float *points, int n)
Permet de calculer la moyenne de chaque case du tableau.
- void **produit_tab** (float *points0, float *points1, int n)
Procédure permettant de remplacer chaque case du tableau par son produit avec elle meme.
- float * **copy_tab** (float *tab, int taille)
Permet de faire la copie d'un tableau.
- float * **approximation** (float *pointsx, float *pointsy, int nb_points)
Permet de faire l'approximation par la methode de regression linéaire.
- float * **approximation_exp** (float *pointsx, float *pointsy, int nb_points)
Permet de faire l'approximation par la methode de regression pour une equation ax^b .

4.1.1 Description détaillée

Contients les fonctions pour l'approximation via la méthode de regression.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

08/11/2022

4.1.2 Documentation des fonctions

4.1.2.1 approximation()

```
float* approximation (
    float * pointsx,
    float * pointsy,
    int nb_points )
```

Permet de faire l'approximation par la methode de regression linéaire.

Paramètres

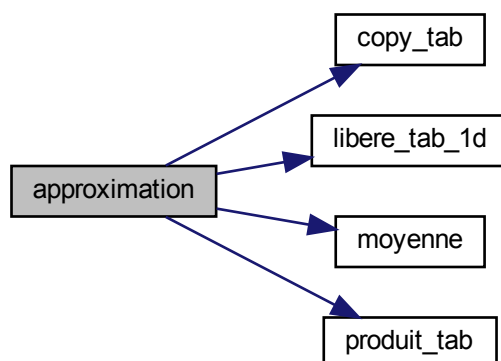
in	<i>pointsx</i>	Pointeur sur la première case du tableau des abscisse.
in	<i>pointsy</i>	Pointeur sur la première case du tableau des ordonnées.
in	<i>nb_points</i>	Nombre de point du tableau.

Renvoie

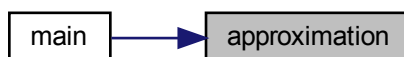
Renvoie un pointeur sur ce nouveau tableau qui contient deux cases, une pour **a** et l'autre pour **b** dans l'équation $ax + b$.

Définition à la ligne 132 du fichier approximation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**4.1.2.2 approximation_exp()**

```

float* approximation_exp (
    float * pointsx,
    float * pointsy,
    int nb_points )
  
```

Permet de faire l'approximation par la méthode de régression pour une équation ax^b .

Paramètres

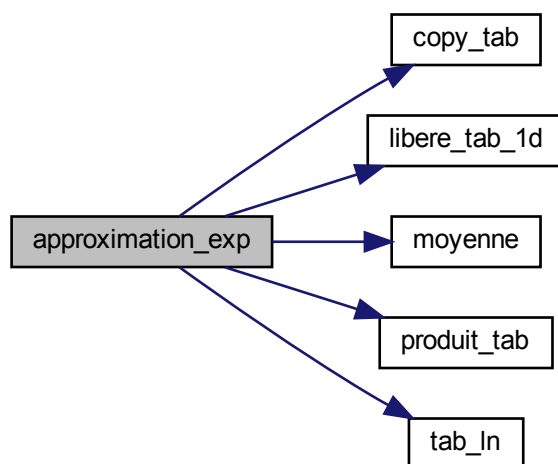
in	<i>pointsx</i>	Pointeur sur la première case du tableau des abscisse.
in	<i>pointsy</i>	Pointeur sur la première case du tableau des ordonnées.
in	<i>nb_points</i>	Nombre de point du tableau.

Renvoie

Renvoie un pointeur sur ce nouveau tableau qui contient deux cases, une pour **a** et l'autre pour **b** dans l'equation ax^b .

Définition à la ligne 193 du fichier approximation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.1.2.3 carre()

```
float carre (
    float x )
```

Permet de calculer un flottant au carre.

Paramètres

in	<i>x</i>	Nombre a élever au carrée.
----	----------	----------------------------

Renvoie

Renvoie la $*x**2$.

Définition à la ligne 20 du fichier approximation.c.

4.1.2.4 copy_tab()

```
float* copy_tab (
    float * tab,
    int taille )
```

Permet de faire la copie d'un tableau.

Paramètres

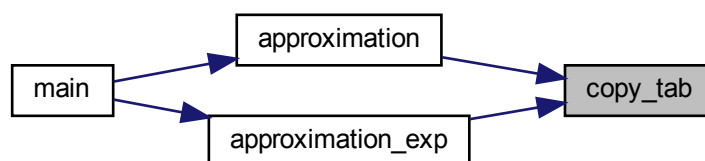
in	<i>tab</i>	Pointeur sur la première case du tabeau.
in	<i>taille</i>	Taille du tableau.

Renvoie

Renvoie un poitneur sur ce nouveau tableau.

Définition à la ligne 105 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :



4.1.2.5 libere_tab_1d()

```
void libere_tab_1d (
    float * tab )
```

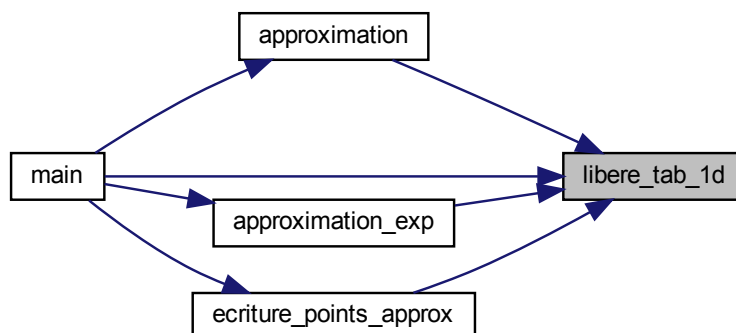
Procédure permettant de libérer tableau 1D.

Paramètres

in	<i>tab</i>	Pointeur sur la première case du tableau.
----	------------	---

Définition à la ligne 31 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :

**4.1.2.6 moyenne()**

```
float moyenne (
    float * points,
    int n )
```

Permet de calculer la moyenne de chaque case du tableau.

Paramètres

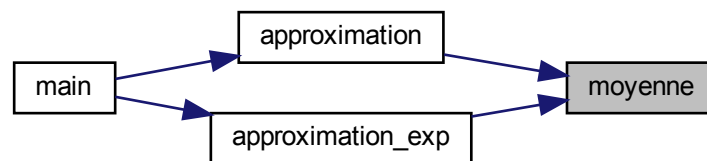
in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Renvoie

Renvoie la $\sum x_i^2$.

Définition à la ligne 57 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :



4.1.2.7 produit_tab()

```
void produit_tab (  
    float * points0,  
    float * points1,  
    int n )
```

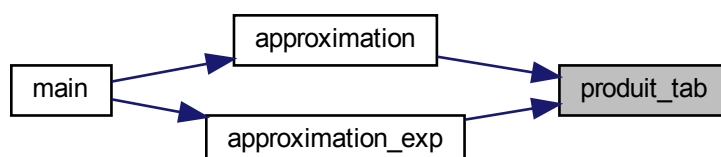
Procédure permettant de remplacer chaque case du tableau par son produit avec elle meme.

Paramètres

in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Définition à la ligne 86 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :



4.1.2.8 tab_ln()

```
void tab_ln (
    float * tab,
    int n )
```

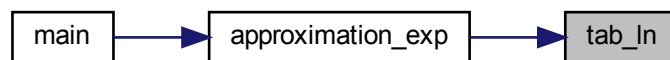
Procédure permettant de remplacer chaque case du tableau par sa composé avec ln.

Paramètres

in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Définition à la ligne 43 du fichier approximation.c.

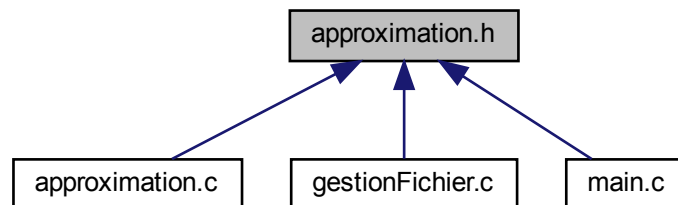
Voici le graphe des appelants de cette fonction :



4.2 Référence du fichier approximation.h

Contient les fonctions pour l'approximation via la méthode de regression.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Fonctions

- float `carre` (float)
Permet de calculer un flottant au carre.
- void `libere_tab_1d` (float *)
Procédure permettant de liberer tableau 1D.
- void `tab_ln` (float *, int)
Procédure permettant de remplacer chaque case du tableau par sa composé avec ln.
- float `moyenne` (float *, int)
Permet de calculer la moyenne de chaque case du tableau.
- void `produit_tab` (float *, float *, int)
Procédure permettant de remplacer chaque case du tableau par son produit avec elle meme.
- float * `approximation` (float *, float *, int)
Permet de faire l'approximation par la methode de regression linéaire.
- float * `approximation_exp` (float *, float *, int)
Permet de faire l'approximation par la methode de regression pour une equation ax^b .

4.2.1 Description détaillée

Contients les fonctions pour l'approximation via la méthode de regression.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

08/11/2022

4.2.2 Documentation des fonctions

4.2.2.1 approximation()

```
float* approximation (
    float * pointsx,
    float * pointsy,
    int nb_points )
```

Permet de faire l'approximation par la methode de regression linéaire.

Paramètres

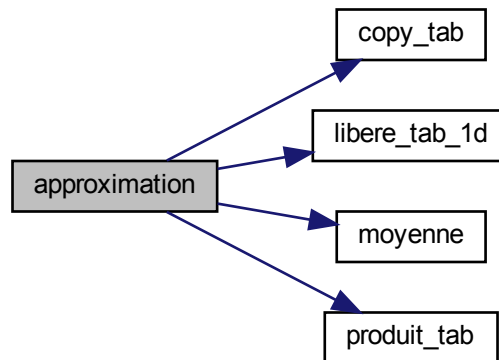
in	<code>pointsx</code>	Pointeur sur la première case du tabeau des abscisse.
in	<code>pointsy</code>	Pointeur sur la première case du tabeau des ordonnées.
in	<code>nb_points</code>	Nombredelapoint du tablea.

Renvoie

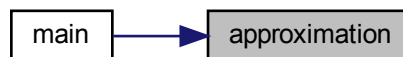
Renvoie un pointeur sur ce nouveau tableau qui contient deux cases, une pour **a** et l'autre pour **b** dans l'équation $ax + b$.

Définition à la ligne 132 du fichier `approximation.c`.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**4.2.2.2 approximation_exp()**

```
float* approximation_exp (
    float * pointsx,
    float * pointsy,
    int nb_points )
```

Permet de faire l'approximation par la méthode de regression pour une équation ax^b .

Paramètres

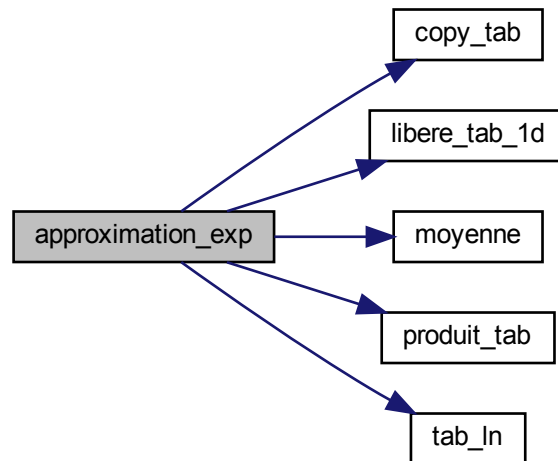
in	<i>pointsx</i>	Pointeur sur la première case du tableau des abscisse.
in	<i>pointsy</i>	Pointeur sur la première case du tableau des ordonnées.
in	<i>nb_points</i>	Nombre de point du tableau.

Renvoie

Renvoie un pointeur sur ce nouveau tableau qui contient deux cases, une pour **a** et l'autre pour **b** dans l'équation ax^b .

Définition à la ligne 193 du fichier approximation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**4.2.2.3 carre()**

```
float carre (
    float x )
```

Permet de calculer un flottant au carre.

Paramètres

in	x	Nombre a élever au carrée.
----	---	----------------------------

Renvoie

Renvoie la $x**2$.

Définition à la ligne 20 du fichier approximation.c.

4.2.2.4 libere_tab_1d()

```
void libere_tab_1d (  
    float * tab )
```

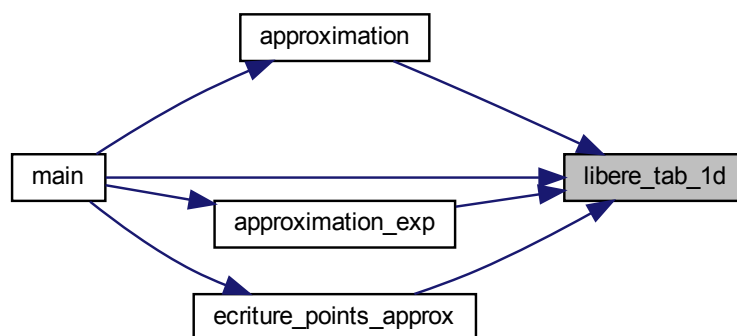
Procédure permettant de liberer tableau 1D.

Paramètres

in	tab	Pointeur sur la première case du tableau.
----	-----	---

Définition à la ligne 31 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :

**4.2.2.5 moyenne()**

```
float moyenne (  
    float * points,  
    int n )
```


Permet de calculer la moyenne de chaque case du tableau.

Paramètres

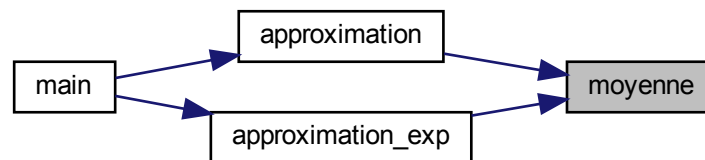
in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Renvoie

Renvoie la $*x**2$.

Définition à la ligne 57 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :

**4.2.2.6 produit_tab()**

```
void produit_tab (
    float * points0,
    float * points1,
    int n )
```

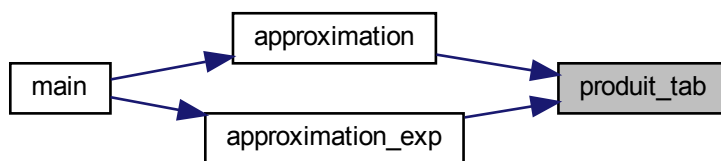
Procédure permettant de remplacer chaque case du tableau par son produit avec elle meme.

Paramètres

in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Définition à la ligne 86 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :



4.2.2.7 tab_ln()

```

void tab_ln (
    float * tab,
    int n )
  
```

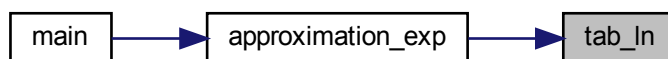
Procédure permettant de remplacer chaque case du tableau par sa composé avec ln.

Paramètres

in	<i>tab</i>	Pointeur sur la première case du tableau.
in	<i>n</i>	Taille du tableau.

Définition à la ligne 43 du fichier approximation.c.

Voici le graphe des appelants de cette fonction :



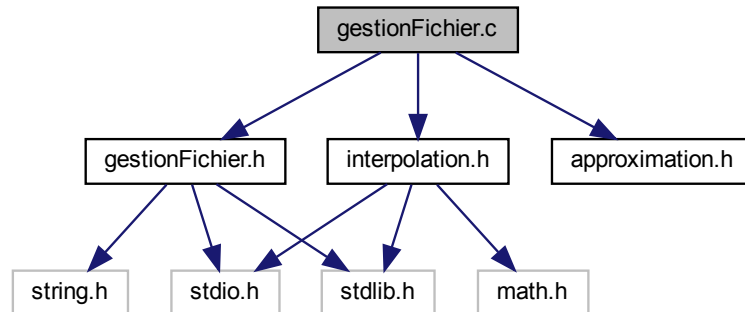
4.3 Référence du fichier gestionFichier.c

```

#include "interpolation.h"
#include "gestionFichier.h"
  
```

```
#include "approximation.h"
```

Graphe des dépendances par inclusion de gestionFichier.c:



Fonctions

- double ** **matrice** (unsigned int n, unsigned int m)
Permet de creer une matrice de double.
- void **freeMat** (double **tab, unsigned int n)
Procédure qui permet de libérer une matrice.
- FILE * **ouvertureFichier** (char *nom_fichier, char *mode)
Permet d'ouvrir un fichier.
- void **fermetureFichier** (FILE *descripteur)
Procédure permettant de fermer un fichier.
- double ** **lecture_points** (char *nom_fichier, unsigned int nb_points)
Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.
- float * **lecture_colonne** (char *nom_fichier, int num_col, int nb_points)
Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.
- void **ecriture_points** (char *nom_fichier, double x, double y)
Procédure qui permet d'ajouter deux points dans un fichier (à la fin).
- void **ecriture_points_approx** (char *nom_fichier, float *tab)
Procédure qui permet d'ajouter deux points dans un fichier (à la fin), ces deux points sont stockés dans un tableau.
- int **liberation_points** (double **points, unsigned int nb_points)
Fonction qui permet de libérer, un tableau de double.

4.3.1 Documentation des fonctions

4.3.1.1 **ecriture_points()**

```
void ecriture_points (
    char * nom_fichier,
    double x,
    double y )
```

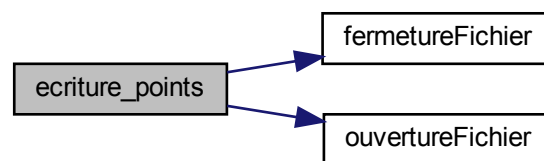
Procédure qui permet d'ajouter deux points dans un fichier (à la fin).

Paramètres

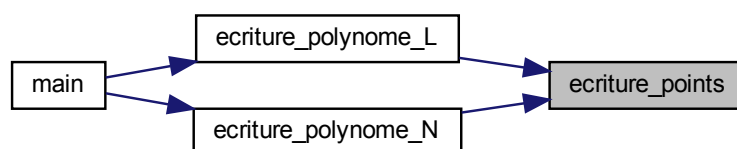
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>x</i>	Valeur à écrire.
in	<i>y</i>	Valeur à écrire.

Définition à la ligne 204 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.3.1.2 `ecriture_points_approx()`

```
void ecriture_points_approx (
    char * nom_fichier,
    float * tab )
```

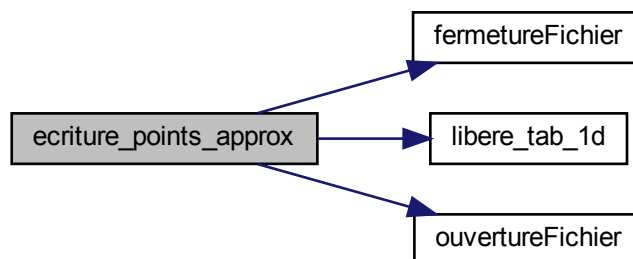
Procédure qui permet d'ajouter deux points dans un fichier (à la fin), ces deux points sont stockés dans un tableau.

Paramètres

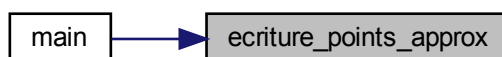
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>tab</i>	tableau des deux points à écrire.

Définition à la ligne 222 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.3.1.3 fermetureFichier()

```
void fermetureFichier (
    FILE * descripteur )
```

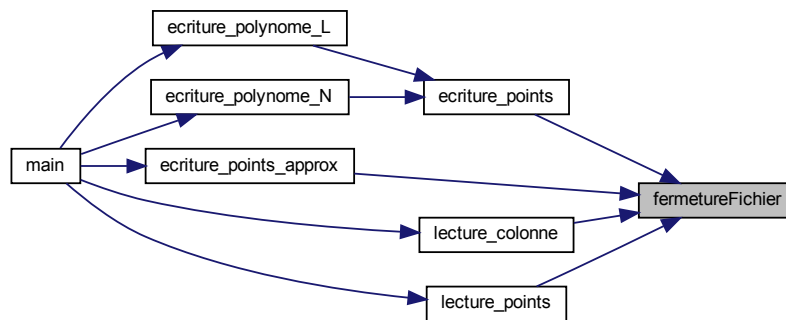
Procédure permettant de fermer un fichier.

Paramètres

in	<i>descripteur</i>	Descripteur de fichier
----	--------------------	------------------------

Définition à la ligne 104 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :



4.3.1.4 freeMat()

```
void freeMat (
    double ** tab,
    unsigned int n )
```

Procédure qui permet de libérer une matrice.

Paramètres

in	<i>n</i>	Nombre de ligne de la matrice à créer.
----	----------	--

Définition à la ligne 61 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :



4.3.1.5 lecture_colonne()

```
float* lecture_colonne (
    char * nom_fichier,
```

```
int num_col,
int nb_points )
```

Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.

Paramètres

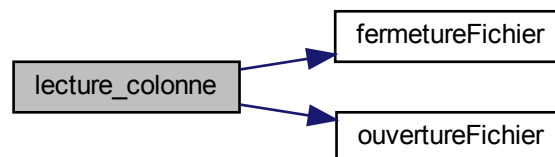
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>num_col</i>	Numero de colonne dans laquelle lire.
in	<i>nb_points</i>	Nombre de points dans un fichier.

Renvoie

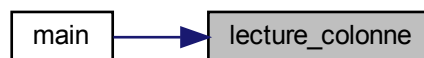
Renvoie un pointeur sur la première case de la matrice contenant les points.

Définition à la ligne 158 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.3.1.6 lecture_points()

```
double** lecture_points (
    char * nom_fichier,
    unsigned int nb_points )
```

Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.

Paramètres

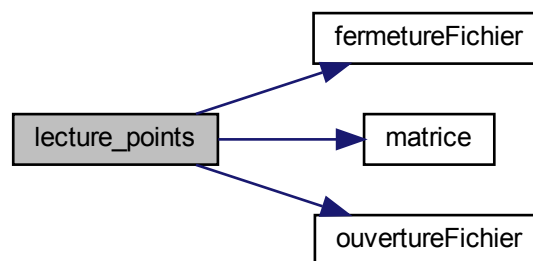
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>nb_points</i>	Nombrede points dans un fichier.

Renvoie

Renvoie un pointeur sur la première case de la matrice contenant les points.

Définition à la ligne 119 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.3.1.7 liberation_points()

```
int liberation_points (  
    double ** points,  
    unsigned int nb_points )
```

Fonction qui permet de libérer, un tableau de double.

Paramètres

in	<i>points</i>	Tableau des nombres de points à liberer.
in	<i>nb_points</i>	Nombre de ligne de la matrice.

Définition à la ligne 250 du fichier gestionFichier.c.

4.3.1.8 matrice()

```
double** matrice (
    unsigned int n,
    unsigned int m )
```

Permet de creer une matrice de double.

Paramètres

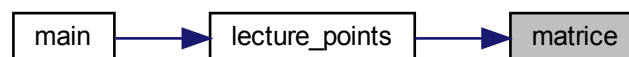
in	<i>n</i>	Nombre de ligne de la matrice à créer.
in	<i>m</i>	Nombre de colonne de la matrice à créer.

Renvoie

Renvoie un pointeur sur la première case de la matrice.

Définition à la ligne 23 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

**4.3.1.9 ouvertureFichier()**

```
FILE* ouvertureFichier (
    char * nom_fichier,
    char * mode )
```

Permet d'ouvrir un fichier.

Paramètres

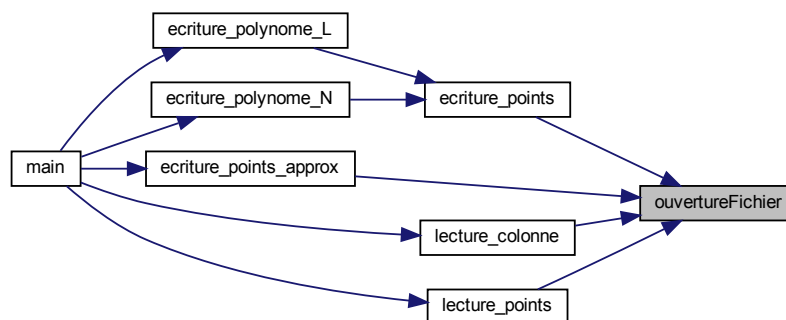
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>mode</i>	Chaine de caractère désignant le mode d'ouverture du fichier.

Renvoie

Renvoie le descripteur du fichier (pointeur).

Définition à la ligne 86 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

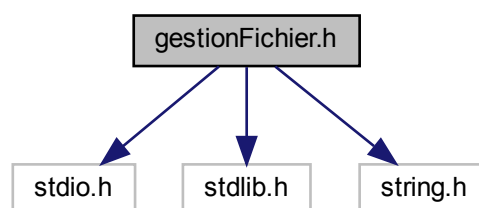


4.4 Référence du fichier gestionFichier.h

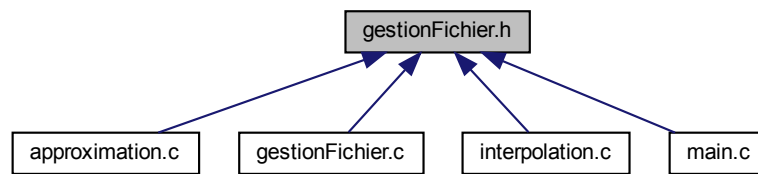
Contient les fonctions de gestion des fichiers.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Grappe des dépendances par inclusion de gestionFichier.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Macros

- #define `NOMBRES_POINTS_1` 20
Correspond au nombre de points pour le jeu de données 1.
- #define `NOMBRES_POINTS_2` 21
Correspond au nombre de points pour le jeu de données 2.
- #define `NOMBRES_POINTS_3` 11
Correspond au nombre de points pour le jeu de données 3.
- #define `NOMBRES_POINTS_4` 7
Correspond au nombre de points pour le jeu de données 4.

Fonctions

- double ** `matrice` (unsigned int, unsigned int)
Permet de creer une matrice de double.
- void `freeMat` (double **, unsigned int)
Procédure qui permet de libérer une matrice.
- FILE * `ouvertureFichier` (char *, char *)
Permet d'ouvrir un fichier.
- void `fermetureFichier` (FILE *)
Procédure permettant de fermer un fichier.
- void `ecriture_points` (char *, double, double)
Procédure qui permet d'ajouter deux points dans un fichier (à la fin).
- float * `lecture_colonne` (char *, int, int)
Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.
- double ** `lecture_points` (char *, unsigned int)
Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.
- void `ecriture_points_approx` (char *, float *)
Procédure qui permet d'ajouter deux points dans un fichier (à la fin), ces deux points sont stockés dans un tableau.

4.4.1 Description détaillée

Contient les fonctions de gestion des fichiers.

Contient les prototypes des fonctions de gestion des fichiers.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

08/11/2022

4.4.2 Documentation des macros

4.4.2.1 NOMBRES_POINTS_1

```
#define NOMBRES_POINTS_1 20
```

Correspond au nombre de points pour le jeu de données 1.

Définition à la ligne 19 du fichier gestionFichier.h.

4.4.2.2 NOMBRES_POINTS_2

```
#define NOMBRES_POINTS_2 21
```

Correspond au nombre de points pour le jeu de données 2.

Définition à la ligne 24 du fichier gestionFichier.h.

4.4.2.3 NOMBRES_POINTS_3

```
#define NOMBRES_POINTS_3 11
```

Correspond au nombre de points pour le jeu de données 3.

Définition à la ligne 29 du fichier gestionFichier.h.

4.4.2.4 NOMBRES_POINTS_4

```
#define NOMBRES_POINTS_4 7
```

Correspond au nombre de points pour le jeu de données 4.

Définition à la ligne 34 du fichier gestionFichier.h.

4.4.3 Documentation des fonctions

4.4.3.1 ecriture_points()

```
void ecriture_points (
    char * nom_fichier,
    double x,
    double y )
```

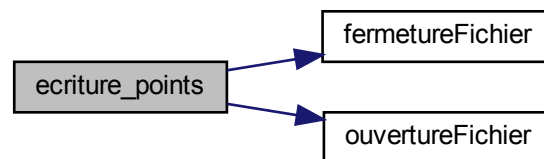
Procédure qui permet d'ajouter deux points dans un fichier (à la fin).

Paramètres

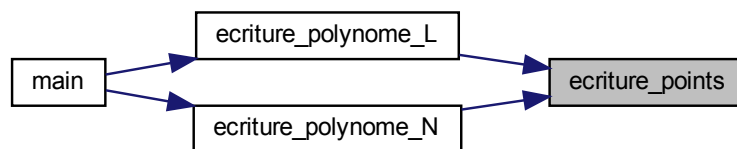
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>x</i>	Valeur à écrire.
in	<i>y</i>	Valeur à écrire.

Définition à la ligne 204 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**4.4.3.2 ecriture_points_approx()**

```

void ecriture_points_approx (
    char * nom_fichier,
    float * tab )
  
```

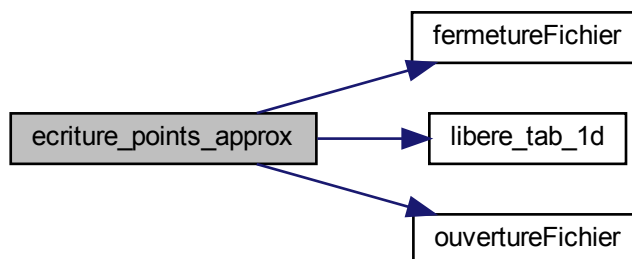
Procédure qui permet d'ajouter deux points dans un fichier (à la fin), ces deux points sont stockés dans un tableau.

Paramètres

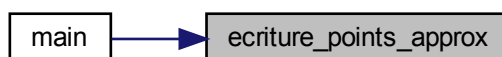
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>tab</i>	tableau des deux points à écrire.

Définition à la ligne 222 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.4.3.3 fermetureFichier()

```
void fermetureFichier (
    FILE * descripteur )
```

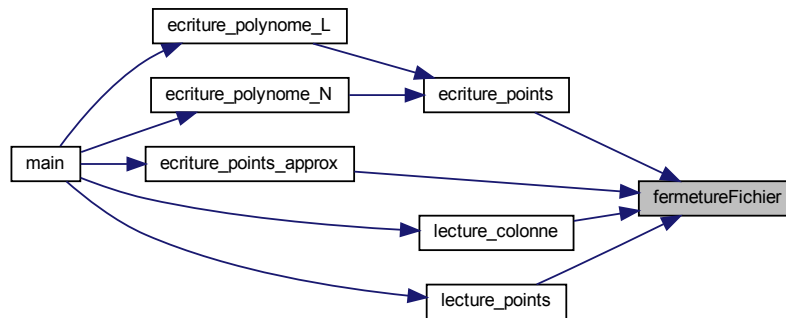
Procédure permettant de fermer un fichier.

Paramètres

in	<i>descripteur</i>	Descripteur de fichier
----	--------------------	------------------------

Définition à la ligne 104 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :



4.4.3.4 freeMat()

```
void freeMat (
    double ** tab,
    unsigned int n )
```

Procédure qui permet de libérer une matrice.

Paramètres

in	<i>n</i>	Nombre de ligne de la matrice à créer.
----	----------	--

Définition à la ligne 61 du fichier `gestionFichier.c`.

Voici le graphe des appelants de cette fonction :



4.4.3.5 lecture_colonne()

```
float* lecture_colonne (
    char * nom_fichier,
```



```
int num_col,
int nb_points )
```

Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.

Paramètres

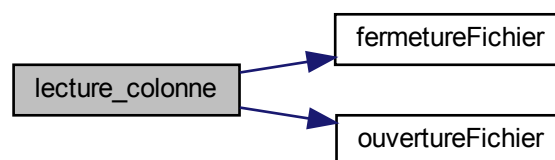
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>num_col</i>	Numero de colonne dans laquelle lire.
in	<i>nb_points</i>	Nombre de points dans un fichier.

Renvoie

Renvoie un pointeur sur la première case de la matrice contenant les points.

Définition à la ligne 158 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.4.3.6 lecture_points()

```
double** lecture_points (
    char * nom_fichier,
    unsigned int nb_points )
```

Permet de lire les points d'un fichier test et d'écrire dans un tableau les résultats.

Paramètres

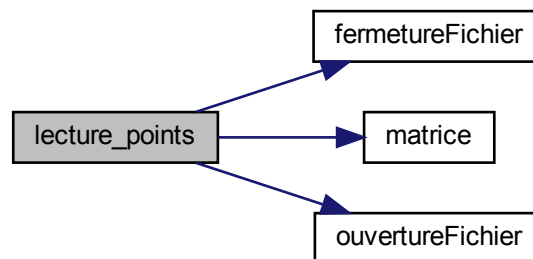
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>nb_points</i>	Nombrede points dans un fichier.

Renvoie

Renvoie un pointeur sur la première case de la matrice contenant les points.

Définition à la ligne 119 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**4.4.3.7 matrice()**

```
double** matrice (  
    unsigned int n,  
    unsigned int m )
```

Permet de creer une matrice de double.

Paramètres

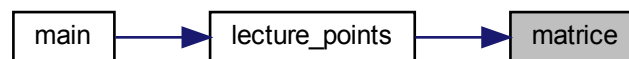
in	<i>n</i>	Nombre de ligne de la matrice à créer.
in	<i>m</i>	Nombre de colonne de la matrice à créer.

Renvoie

Renvoie un pointeur sur la première case de la matrice.

Définition à la ligne 23 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

**4.4.3.8 ouvertureFichier()**

```
FILE* ouvertureFichier (
    char * nom_fichier,
    char * mode )
```

Permet d'ouvrir un fichier.

Paramètres

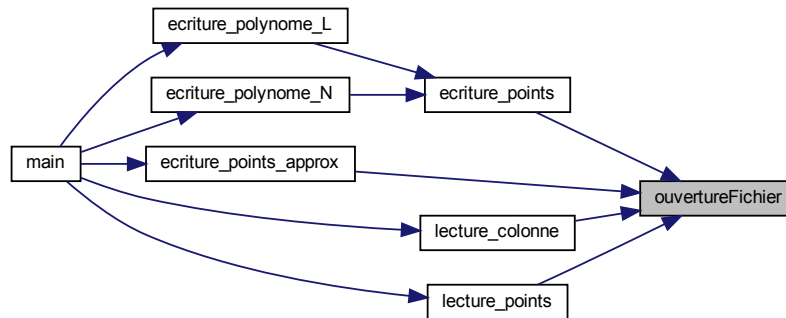
in	<i>nom_fichier</i>	Chaine de caractère désignant le nom fichier.
in	<i>mode</i>	Chaine de caractère désignant le mode d'ouverture du fichier.

Renvoie

Renvoie le descripteur du fichier (pointeur).

Définition à la ligne 86 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

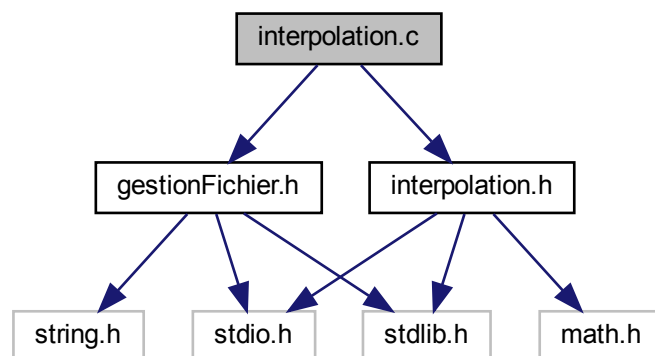


4.5 Référence du fichier interpolation.c

Contient les prototypes des fonctions de gestion des fichiers.

```
#include "interpolation.h"  
#include "gestionFichier.h"
```

Graphe des dépendances par inclusion de `interpolation.c`:



Fonctions

- double `methode_lagrange` (double `**points`, unsigned int `nb_points`, double `x`)
Permet de calculer la valeur du polynome de Lagrange en 1 point considéré.
- double `methode_neville` (double `**points`, unsigned int `nb_points`, double `x`)
Permet de calculer la valeur du polynome de Neville en 1 point considéré.
- void `ecriture_polynome_L` (char `*nom_fichier`, double `**points`, unsigned int `nb_points`, double `borne_inf`, double `borne_sup`)
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considéré (grâce au polynome de Lagrange).
- void `ecriture_polynome_N` (char `*nom_fichier`, double `**points`, unsigned int `nb_points`, double `borne_inf`, double `borne_sup`)
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considéré (grâce au polynome de Lagrange).

4.5.1 Description détaillée

Contients les portotypes des fonctions de gestion des fichiers.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

08/11/2022

4.5.2 Documentation des fonctions

4.5.2.1 `ecriture_polynome_L()`

```
void ecriture_polynome_L (
    char * nom_fichier,
    double ** points,
    unsigned int nb_points,
    double borne_inf,
    double borne_sup )
```

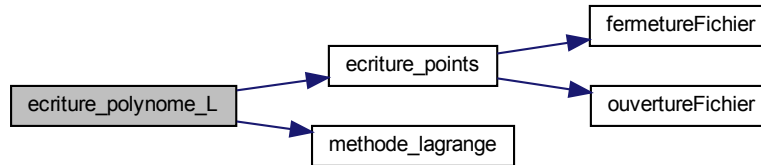
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considéré (grâce au polynome de Lagrange).

Paramètres

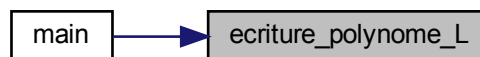
in	<code>nom_fichier</code>	Nom du fichier où écrire les points.
in	<code>points</code>	Tableau qui contient les points données du polynome.
in	<code>nb_points</code>	Nombre de point du polynome considéré.
in	<code>borne_inf</code>	Borne inf de la valeur en laquelle on souhaite évaluer le polynome.
in	<code>borne_sup</code>	Borne sup de la valeur en laquelle on souhaite évaluer le polynome.

Définition à la ligne 116 du fichier interpolation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.5.2.2 `ecriture_polynome_N()`

```

void ecriture_polynome_N (
    char * nom_fichier,
    double ** points,
    unsigned int nb_points,
    double borne_inf,
    double borne_sup )
  
```

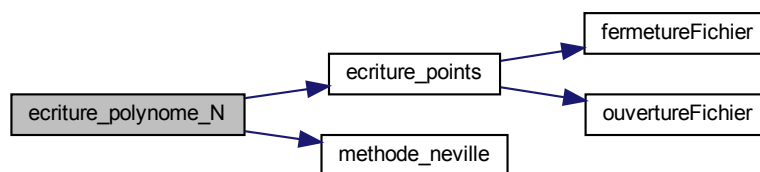
Procédure qui permet d'écrire les valeurs des **x** et de **f(x)** pour les points considérer (grâce au polynome de Lagrange).

Paramètres

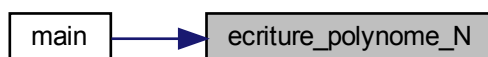
in	<i>nom_fichier</i>	Nomdu fichier où écrire les points.
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considérer.
in	<i>borne_inf</i>	Borne inf de la valeur en laquelle on souhaite évaluer le polynome.
in	<i>borne_sup</i>	Borne inf de la valeur en laquelle on souhaite évaluer le polynome.

Définition à la ligne 147 du fichier interpolation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.5.2.3 methode_lagrange()

```
double methode_lagrange (
    double ** points,
    unsigned int nb_points,
    double x )
```

Permet de calculer la valeur du polynome de Lagrange en 1 point considéré.

Paramètres

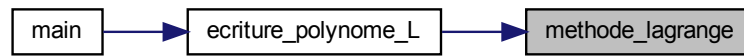
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considéré.
in	<i>x</i>	Nombreauel on souhaite connaitre l'évaluation.

Renvoie

Renvoie la valeur du polynome de Lagrange au points **x** point considéré.

Définition à la ligne 22 du fichier interpolation.c.

Voici le graphe des appelants de cette fonction :



4.5.2.4 methode_neville()

```
double methode_neville (
    double ** points,
    unsigned int nb_points,
    double x )
```

Permet de calculer la valeur du polynome de Neville en 1 point considéré.

Paramètres

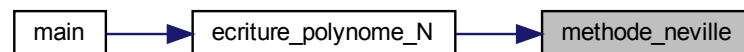
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considéré.
in	<i>x</i>	Nombre au quel on souhaite connaître l'évaluation.

Renvoie

Renvoie la valeur du polynome de Lagrange au points **x** point considéré.

Définition à la ligne 68 du fichier interpolation.c.

Voici le graphe des appelants de cette fonction :

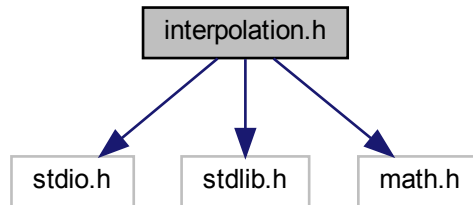


4.6 Référence du fichier interpolation.h

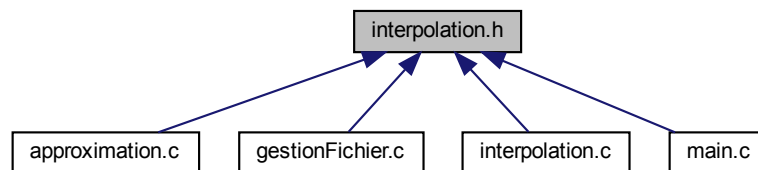
Contients les portotypes des fonctions pour l'interpolation.


```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Graphe des dépendances par inclusion de interpolation.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Fonctions

- double * [cree_polynome](#) (unsigned int)
- void [affiche_polynome](#) (double *, unsigned int)
- double * [produit_polynomes](#) (double *, double *, unsigned int, unsigned int)
- double [methode_lagrange](#) (double **, unsigned int, double)
Permet de calculer la valeur du polynome de Lagrange en 1 point considéré.
- double [methode_neville](#) (double **, unsigned int, double)
Permet de calculer la valeur du polynome de Neville en 1 point considéré.
- void [ecriture_polynome_L](#) (char *, double **, unsigned int, double, double)
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considéré (grâce au polynome de Lagrange).
- void [ecriture_polynome_N](#) (char *, double **, unsigned int, double, double)
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considéré (grâce au polynome de Lagrange).

4.6.1 Description détaillée

Contient les prototypes des fonctions pour l'interpolation.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

08/11/2022

4.6.2 Documentation des fonctions

4.6.2.1 affiche_polynome()

```
void affiche_polynome (
    double * ,
    unsigned int )
```

4.6.2.2 cree_polynome()

```
double* cree_polynome (
    unsigned int )
```

4.6.2.3 ecriture_polynome_L()

```
void ecriture_polynome_L (
    char * nom_fichier,
    double ** points,
    unsigned int nb_points,
    double borne_inf,
    double borne_sup )
```

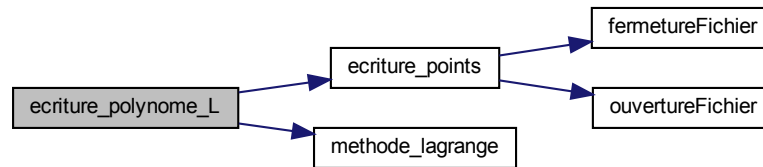
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considérés (grâce au polynôme de Lagrange).

Paramètres

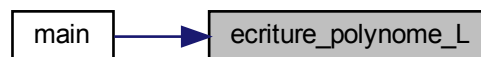
in	<i>nom_fichier</i>	Nom du fichier où écrire les points.
in	<i>points</i>	Tableau qui contient les points donnés du polynôme.
in	<i>nb_points</i>	Nombre de point du polynôme considéré.
in	<i>borne_inf</i>	Borne inf de la valeur en laquelle on souhaite évaluer le polynôme.
in	<i>borne_sup</i>	Borne sup de la valeur en laquelle on souhaite évaluer le polynôme.

Définition à la ligne 116 du fichier interpolation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.6.2.4 `ecriture_polynome_N()`

```

void ecriture_polynome_N (
    char * nom_fichier,
    double ** points,
    unsigned int nb_points,
    double borne_inf,
    double borne_sup )
  
```

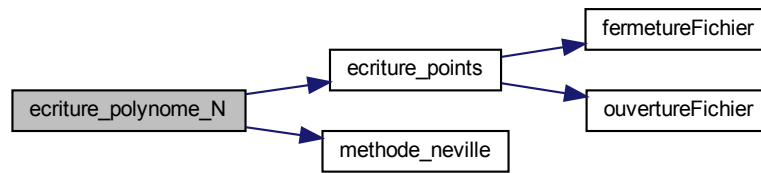
Procédure qui permet d'écrire les valeurs des x et de $f(x)$ pour les points considérer (grâce au polynome de Lagrange).

Paramètres

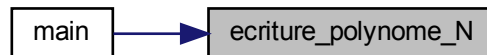
in	<i>nom_fichier</i>	Nomdu fichier où écrire les points.
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considérer.
in	<i>borne_inf</i>	Borne inf de la valeur en laquelle on souhaite évaluer le polynome.
in	<i>borne_sup</i>	Borne inf de la valeur en laquelle on souhaite évaluer le polynome.

Définition à la ligne 147 du fichier interpolation.c.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.6.2.5 methode_lagrange()

```
double methode_lagrange (
    double ** points,
    unsigned int nb_points,
    double x )
```

Permet de calculer la valeur du polynome de Lagrange en 1 point considérer.

Paramètres

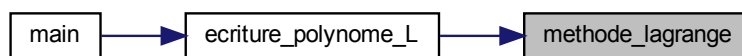
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considérer.
in	<i>x</i>	Nombreauquel on souhaite connaitre l'évaluation.

Renvoie

Renvoie la valeur du polynome de Lagrange au points **x** point considérer.

Définition à la ligne 22 du fichier interpolation.c.

Voici le graphe des appelants de cette fonction :



4.6.2.6 methode_neville()

```
double methode_neville (
    double ** points,
    unsigned int nb_points,
    double x )
```

Permet de calculer la valeur du polynome de Neville en 1 point considéré.

Paramètres

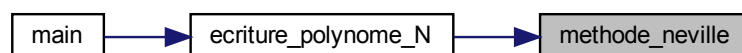
in	<i>points</i>	Tableau qui contient les points données du polynome.
in	<i>nb_points</i>	Nombre de point du polynome considéré.
in	<i>x</i>	Nombre au quel on souhaite connaître l'évaluation.

Renvoie

Renvoie la valeur du polynome de Lagrange au points **x** point considéré.

Définition à la ligne 68 du fichier interpolation.c.

Voici le graphe des appelants de cette fonction :



4.6.2.7 produit_polynomes()

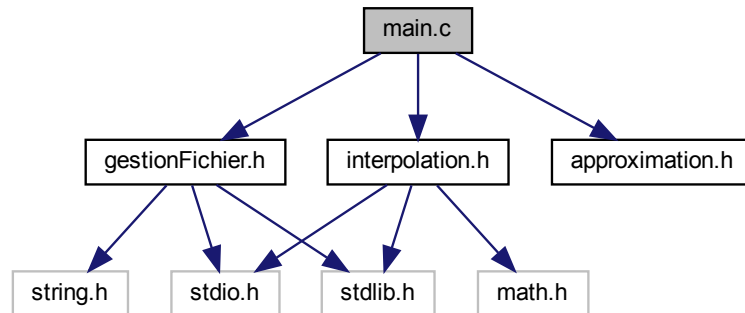
```
double* produit_polynomes (
    double * ,
    double * ,
    unsigned int ,
    unsigned int )
```

4.7 Référence du fichier main.c

Contients l'appel au différente focntion d'interpolation et de regression.

```
#include "interpolation.h"  
#include "gestionFichier.h"  
#include "approximation.h"
```

Graphe des dépendances par inclusion de main.c:



Fonctions

— int `main` ()

4.7.1 Description détaillée

Contients l'appel au différente focntion d'interpolation et de regression.

Auteur

Valentin PORTAIL et Jérémie VILLEPREUX

Version

1.0

Date

10/11/2022

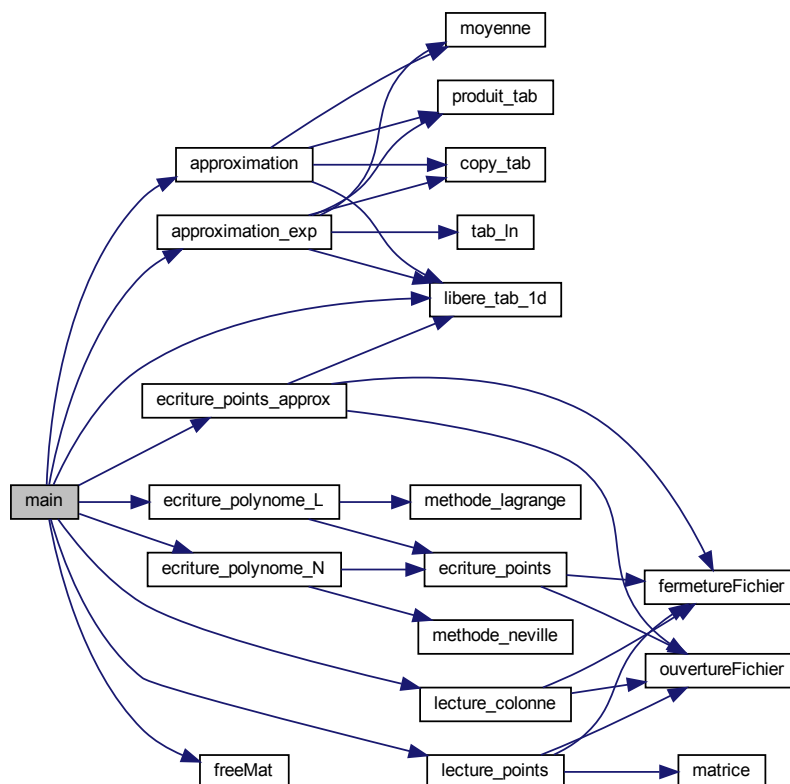
4.7.2 Documentation des fonctions

4.7.2.1 main()

```
int main ( )
```

Définition à la ligne 13 du fichier main.c.

Voici le graphe d'appel pour cette fonction :



4.8 Référence du fichier TRACE PYTHON/trace.py

Espaces de nommage

— [trace](#)

Fonctions

— `def trace (fichier, fichierP, choix)`

4.9 Référence du fichier TRACE PYTHON/trace_approx.py

Espaces de nommage

— [trace_approx](#)

Fonctions

— `def trace_approx (fichier, fichierP, choix)`

Index

affiche_polynome
 interpolation.h, [44](#)

approximation
 approximation.c, [8](#)
 approximation.h, [15](#)

approximation.c, [7](#)
 approximation, [8](#)
 approximation_exp, [9](#)
 carre, [10](#)
 copy_tab, [11](#)
 libere_tab_1d, [11](#)
 moyenne, [12](#)
 produit_tab, [13](#)
 tab_In, [13](#)

approximation.h, [14](#)
 approximation, [15](#)
 approximation_exp, [16](#)
 carre, [17](#)
 libere_tab_1d, [18](#)
 moyenne, [18](#)
 produit_tab, [20](#)
 tab_In, [21](#)

approximation_exp
 approximation.c, [9](#)
 approximation.h, [16](#)

carre
 approximation.c, [10](#)
 approximation.h, [17](#)

copy_tab
 approximation.c, [11](#)

cree_polynome
 interpolation.h, [44](#)

ecriture_points
 gestionFichier.c, [22](#)
 gestionFichier.h, [31](#)

ecriture_points_approx
 gestionFichier.c, [23](#)
 gestionFichier.h, [32](#)

ecriture_polynome_L
 interpolation.c, [39](#)
 interpolation.h, [44](#)

ecriture_polynome_N
 interpolation.c, [40](#)
 interpolation.h, [45](#)

fermetureFichier
 gestionFichier.c, [24](#)
 gestionFichier.h, [33](#)

freeMat
 gestionFichier.c, [25](#)
 gestionFichier.h, [34](#)

gestionFichier.c, [21](#)
 ecriture_points, [22](#)
 ecriture_points_approx, [23](#)
 fermetureFichier, [24](#)
 freeMat, [25](#)
 lecture_colonne, [25](#)
 lecture_points, [26](#)
 liberation_points, [27](#)
 matrice, [28](#)
 ouvertureFichier, [28](#)

gestionFichier.h, [29](#)
 ecriture_points, [31](#)
 ecriture_points_approx, [32](#)
 fermetureFichier, [33](#)
 freeMat, [34](#)
 lecture_colonne, [34](#)
 lecture_points, [35](#)
 matrice, [36](#)
 NOMBRES_POINTS_1, [31](#)
 NOMBRES_POINTS_2, [31](#)
 NOMBRES_POINTS_3, [31](#)
 NOMBRES_POINTS_4, [31](#)
 ouvertureFichier, [37](#)

interpolation.c, [38](#)
 ecriture_polynome_L, [39](#)
 ecriture_polynome_N, [40](#)
 methode_lagrange, [41](#)
 methode_neville, [42](#)

interpolation.h, [42](#)
 affiche_polynome, [44](#)
 cree_polynome, [44](#)
 ecriture_polynome_L, [44](#)
 ecriture_polynome_N, [45](#)
 methode_lagrange, [46](#)
 methode_neville, [47](#)
 produit_polynomes, [47](#)

lecture_colonne
 gestionFichier.c, [25](#)
 gestionFichier.h, [34](#)

lecture_points
 gestionFichier.c, [26](#)
 gestionFichier.h, [35](#)

liberation_points
 gestionFichier.c, [27](#)

- libere_tab_1d
 - approximation.c, [11](#)
 - approximation.h, [18](#)
- main
 - main.c, [48](#)
- main.c, [48](#)
 - main, [48](#)
- matrice
 - gestionFichier.c, [28](#)
 - gestionFichier.h, [36](#)
- methode_lagrange
 - interpolation.c, [41](#)
 - interpolation.h, [46](#)
- methode_neville
 - interpolation.c, [42](#)
 - interpolation.h, [47](#)
- moyenne
 - approximation.c, [12](#)
 - approximation.h, [18](#)
- NOMBRES_POINTS_1
 - gestionFichier.h, [31](#)
- NOMBRES_POINTS_2
 - gestionFichier.h, [31](#)
- NOMBRES_POINTS_3
 - gestionFichier.h, [31](#)
- NOMBRES_POINTS_4
 - gestionFichier.h, [31](#)
- ouvertureFichier
 - gestionFichier.c, [28](#)
 - gestionFichier.h, [37](#)
- produit_polynomes
 - interpolation.h, [47](#)
- produit_tab
 - approximation.c, [13](#)
 - approximation.h, [20](#)
- tab_In
 - approximation.c, [13](#)
 - approximation.h, [21](#)
- trace, [5](#)
 - trace, [5](#)
- TRACE PYTHON/trace.py, [49](#)
- TRACE PYTHON/trace_approx.py, [49](#)
- trace_approx, [5](#)
 - trace_approx, [5](#)