Fiche n^0 7. Résolution d'un problème aux limites sur maillage non régulier

Exercice 1 – Méthode des différences finies

Etant donnés un réel $\lambda > 0$, deux réels a et b et une fonction f, on souhaite résoudre le problème aux limites $-u'' + \lambda u = f$ sur l'intervalle $]0;1[$ avec les conditions aux limites de Dirichlet $u(0) = a$ et $u(1) = b$ sur un maillage non régulier décrit par l'ensemble de ses N sommets de coordonnées x_i, où l'on aura $x_1 = 0$ et $x_N = 1$.

a) En écrivant des formules de Taylor autour du point x_i, déterminer les coefficients a_i, b_i et c_i tels que

$$a_i u(x_{i-1}) + b_i u(x_i) + c_i u(x_{i+1}) = -u''(x_i) + O(h),$$

où la fonction u est supposée suffisamment régulière, et où $h = \sup |x_{i+1} - x_i|$.

b) En déduire l'écriture matricielle d'un système de $(N-2)$ équations à $(N-2)$ inconnues discrétisant le problème aux limites ci-dessus (les valeurs de $u(x_1) = u(0)$ et $u(x_N) = u(1)$ étant connues, les inconnues correspondantes seront éliminées du système).

c) Montrer que la matrice de ce système est à diagonale strictement dominante, et en déduire l'existence pour tout second membre et l'unicité de la solution à ce système linéaire.

Exercice 2 – Résolution du système linéaire associé

On rappelle la méthode de Jacobi pour résoudre un système du type $Ax = b$ (on suppose A inversible) : Soit D la diagonale, supposée à coefficients tous non-nuls, de la matrice A et soit $x^{(0)}$ un vecteur quelconque. On établit la suite itérative suivante :

$$x^{(k+1)} = \left(I - D^{-1}A\right) x^{(k)} + D^{-1}b$$

a) Montrer que si cette suite itérative converge, c'est nécessairement vers la solution x^* du système $Ax = b$, et ce pour tout itéré initial $x^{(0)}$.

b) On pose $e^{(k)} = x^{(k)} - x^*$. Ecrire la relation qui lie $e^{(k+1)}$ à $e^{(k)}$.

c) En déduire une condition nécessaire et suffisante de convergence de cet algorithme.

d) Vérifier que si A est à diagonale strictement dominante, alors cette condition est remplie.

Exercice 3 – Génération d'un maillage aléatoire

Il s'agit de générer un maillage du segment $[x_{\text{deb}}; x_{\text{fin}}]$ avec N sommets, de telle sorte que le rapport de la plus grande cellule sur la plus petite ne dépasse pas x_{rap}. On propose l'implémentation suivante :

```
// fichier  gen_mesh.c

#include <stdio.h>
#include <stdlib.h>
#include <time.h> //Pour utiliser la fonction time
#include "gen_mesh.h"
```

```
double* gen_mesh (int N, double x_deb, double x_fin, double x_rap){

double* maillage = NULL;
double* intervalles = NULL;

double xrenorm;
maillage = malloc (N * sizeof(double)); // les N coordonnées des points
intervalles = malloc ((N - 1) * sizeof(double)); // il y a donc N-1 intervalles

if (maillage == NULL)
{
        printf("Impossible d'allouer le tableau maillage dans gen_mesh \n");
         return(NULL);
}

if (intervalles == NULL)
{
        printf("Impossible d'allouer le tableau intervalles dans gen_mesh \n");
         return(NULL);
}

int i;
double longueur = 0.0;
double facteur = 1.0;

if (x_rap < 1){ // précaution si l'utilisateur a entré un réel plus petit que 1
facteur = fabs(x_rap/(1.0 - x_rap));
}
if (x_rap > 1){
facteur = 1.0/(x_rap - 1.0);
}


srand(time(NULL)); // initialisation de rand

for (i = 0; i < N-1; i++)
{
intervalles[i] = (facteur + rand()/(double)RAND_MAX );


// rand() retourne un entier entre 0 et RAND_MAX

// donc rand()/(double)RAND_MAX retourne un réel entre 0 et 1

// la longueur des intervalles sera donc comprise entre facteur et facteur + 1

longueur +=  intervalles[i]; // incrémente la longueur du maillage
}


// On construit le maillage qui doit avoir pour borne x_deb et x_fin

maillage[0] = x_deb;
```

```
xrenorm = (x_fin - x_deb)/longueur;

for (i = 0; i < N-1; i++)
{
maillage[i+1] = maillage[i]+intervalles[i]*xrenorm;
}

free(intervalles);
return(maillage);
}
```

Exercice 4 – **Allocation d'une matrice - Méthode de Jacobi**

On écrira une procédure pour allouer une matrice (pleine) de doubles, et l'initialiser à 0.
Ceci se fait de la façon suivante :

```
//fichier alloue_matrice.c

#include <stdio.h>
#include <stdlib.h>
#include "alloue_matrice.h"

double** alloue_matrice (int p, int q){
double** M = NULL;
int i,j;

M = malloc( p * sizeof( double* ) );
for(i = 0 ; i < p ; i++){
    M[i] = malloc( q * sizeof (double ) );
}

for(i = 0 ; i < p ; i++){
   for(j = 0 ; j < q ; j++){
      M[i][j] = 0.0; // acces aux éléments de la matrice par M[i][j]
   }
}
return M;
}
```

Dans le `main.c`, une fois la matrice allouée et remplie avec les valeurs calculées dans l'Exercice 1, assembler aussi le second membre (on pensera à utiliser un pointeur de fonction pour écrire une procédure indépendante de la fonction f), écrire la résolution d'un système linéaire par la méthode de Jacobi et calculer les valeurs approchées (on se donnera la fonction f et les valeurs des conditions aux limites), puis comparer celles-ci aux valeurs exactes de la solution analytique.