ECOULEMENT DANS UN BASSIN

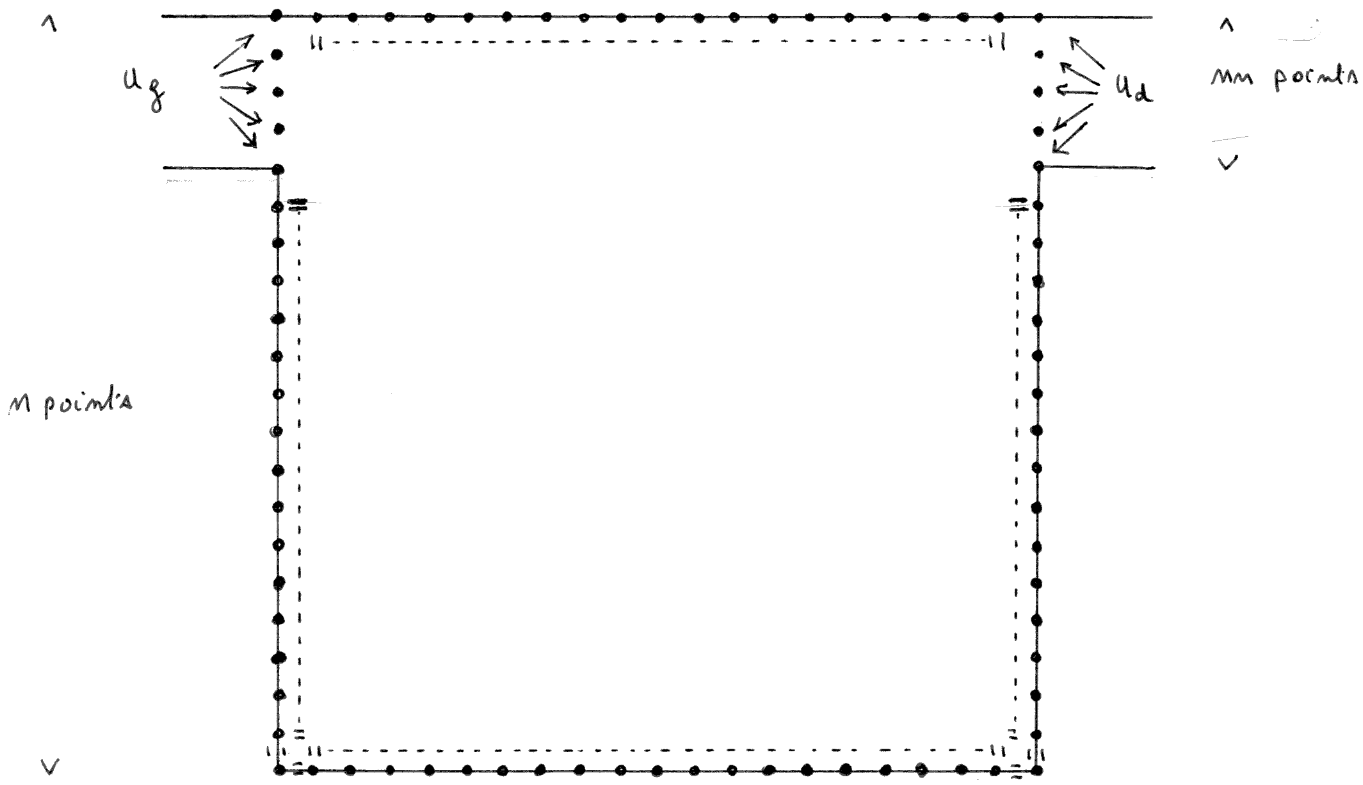
Conditions aux limites pour l'écoulement dans un bassin.

On cherche le potentiel des vitesses u qui obéit à l'équation $\Delta u = 0$.

On résoud itérativement en écrivant partout (sauf sur les bords)

$$u_{ij} = \frac{1}{4}\left(u_{ij-1} + u_{ij+1} + u_{i-1j} + u_{i+1j} \right)$$

avec les conditions aux limites suivantes :



u_g et u_d sont des constantes, valeurs choisies du potentiel des vitesses à l'entrée et à la sortie du bassin.

Calcul des lignes de champ.

1) Supposons qu'on connaisse le champ $\vec{E}(x,y)$ en tout point :

$$E_x = -\frac{\partial V}{\partial x} \qquad E_y = -\frac{\partial V}{\partial y}$$

On cherche les lignes de champ par l'intermédiaire de leurs équations paramétriques, $\begin{cases} x(s) \\ y(s) \end{cases}$ s étant l'abscisse curviligne sur la courbe.

Le vecteur $\left(\frac{dx}{ds}, \frac{dy}{ds}\right)$ tangent à la courbe doit être parallèle à $\vec{E}$ et il est unitaire. On a donc :

$$\frac{dx}{ds} = \frac{E_x}{\sqrt{E_x^2 + E_y^2}} \qquad \frac{dy}{ds} = \frac{E_y}{\sqrt{E_x^2 + E_y^2}}$$

$E_x(x,y)$ et $E_y(x,y)$ étant connus c'est un système de deux équations différentielles couplées du premier ordre de la forme :

$$x' = F(x,y) \qquad y' = G(x,y)$$

que l'on peut résoudre numériquement par Runge-Kutta.

2) En réalité ce qu'on connait c'est seulement le potentiel, sur un réseau de points $V(x_i, y_j)$ avec $\begin{cases} x_i = i \times hh & i = 0, \ldots M_i \\ y_i = j \times hh & j = 0, \ldots M_j \end{cases}$

hh étant le pas du réseau exprimé par exemple en mètres.

Pour avoir le champ en un point quelconque de coordonnées (x,y) on fait une interpolation linéaire à deux dimensions. Utiliser pour cela la fonction grad qui fait partie de la bibliothèque des fonctions du Magistère et dont le mode d'emploi est fourni à la page suivante.

```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<bibli_fonctions.h>
//****************************************************************************************************************************
// La fonction grad calcule le gradient d'une fonction de deux variables
// En entrée :
// ff=tableau-pointeur contenant, aux points du reseau [0,ni-1][0,nj-1], les valeurs de la fonction dont on veut calculer le gradient
// hh=longueur du pas du réseau, dans une unité au choix (mètres par exemple)
// x et y position du point où l'on veut le gradient de la fonction ff, dans la même unité que hh et à l'intérieur du rectangle [hh,(ni-2)hh][hh,(nj-2)hh]
// En sortie :
// Si le point (x,y) est à l'extérieur du rectangle [hh,(ni-2)hh][hh,(nj-2)hh], la valeur de la fonction grad est 0 et les valeurs de *gx et *gy sont sans signification
// Si le point (x,y) est à l'intérieur du rectangle [hh,(ni-2)hh][hh,(nj-2)hh], la valeur de la fonction grad est 1 et les valeurs de *gx et *gy sont les composantes du gradient de ff au point (x,y)
int grad(double **ff,int ni,int nj,double hh,double x,double y,double *gx,double *gy)
{
  int i,j; double t,u,g1x,g1y,g2x,g2y,g3x,g3y,g4x,g4y;
  t=x/hh; i=(int)floor(t); u=y/hh; j=(int)floor(u);
  if(i<1 || i>ni-3 || j<1 || j>nj-3) {printf("Le point (x,y) n'est pas dans le rectangle [%lg,%lg][%lg,%lg] : x=%lg y=%lg\n",hh,(ni-2)*hh,hh,(nj-2)*hh,x,y); return(0);}
  t=t-i; u=u-j;
  // Calcul des grads au quatre coins
  g1x=(ff[i+1][j]-ff[i-1][j])/2/hh; g2x=(ff[i+2][j]-ff[i][j])/2/hh; g3x=(ff[i+2][j+1]-ff[i][j+1])/2/hh; g4x=(ff[i+1][j+1]-ff[i-1][j+1])/2/hh;
  g1y=(ff[i][j+1]-ff[i][j-1])/2/hh; g2y=(ff[i+1][j+1]-ff[i+1][j-1])/2/hh; g3y=(ff[i+1][j+2]-ff[i+1][j])/2/hh; g4y=(ff[i][j+2]-ff[i][j])/2/hh;
  // Calcul du grad en x,y par interpolation linéaire
  *gx=g1x+t*(g2x-g1x)+u*(g4x-g1x)+t*u*(g3x+g1x-g4x-g2x); *gy=g1y+t*(g2y-g1y)+u*(g4y-g1y)+t*u*(g3y+g1y-g4y-g2y);
  return(1);
}
//****************************************************************************************************************************
int main()
{
  int i,j,ni=501,nj=701;
  double x,y,hh,ex,ey;
  double **pot=Mat_alloc_double(ni,nj);
  hh=7.2e-3;
  for(i=0;i<ni;i++) for(j=0;j<nj;j++) {x=i*hh; y=j*hh; pot[i][j]=x+2.7*y+8.4*x*y*y;}
  x=278.57*hh; y=393.19*hh;
  if(grad(pot,ni,nj,hh,x,y,&ex,&ey)!=1) exit(0);
  printf("ex=%lg ey=%lg\n",ex,ey);
  // Vérification :
  printf("ex=%lg ey=%lg\n",1+8.4*y*y,2.7+8.4*x*2*y);
  return(0);
}
```