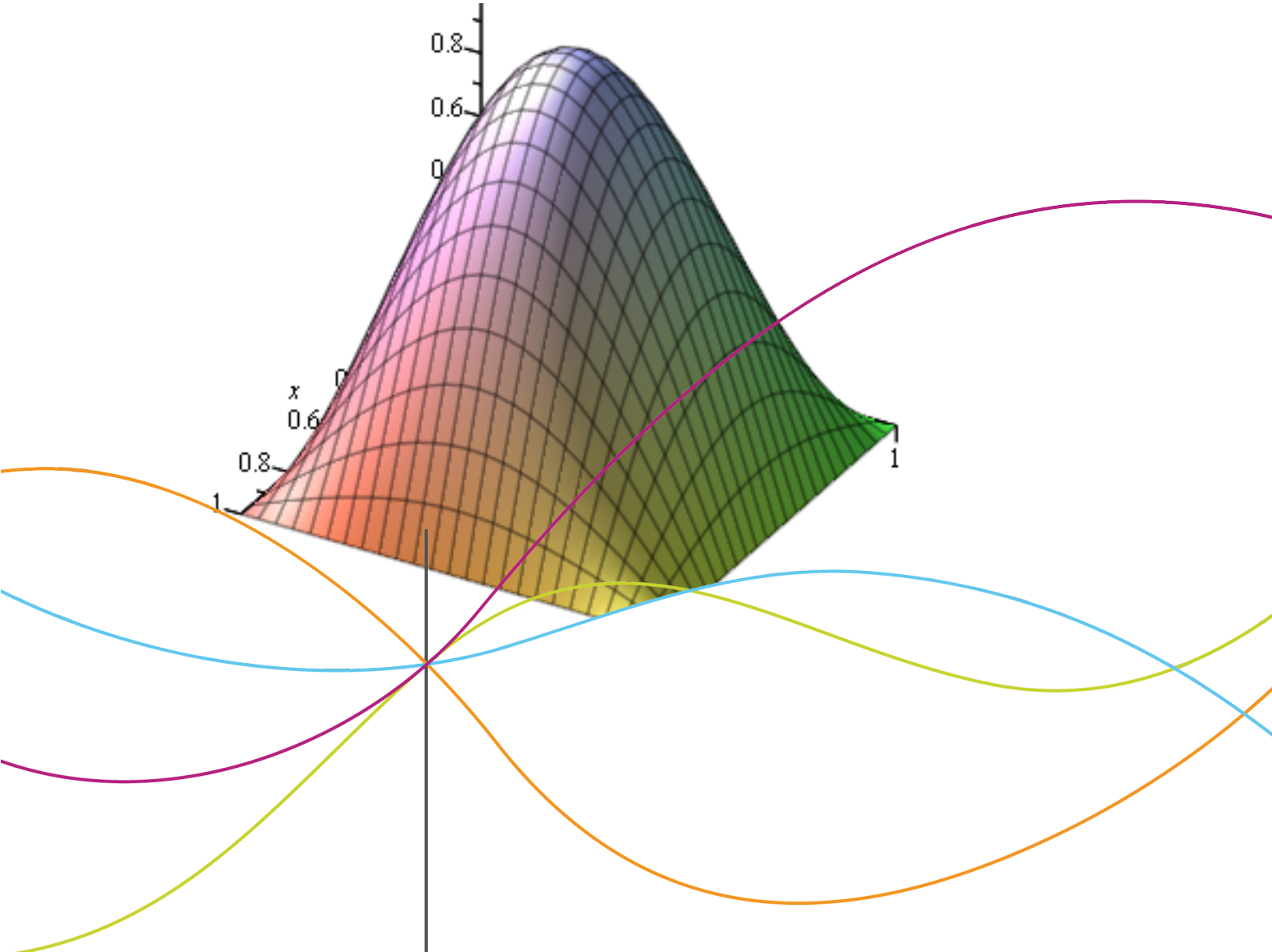
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6
STPI/P6/2015-3

Déformation d'une membrane



Enseignant responsable
Bernard Gleyse

Étudiants :
Runli Ding
Metodi Kanov
Elliot Renaud
Ivan Kanov
Marème Thiam

Date de remise du rapport : 15/06/15

Référence du projet : STPI[1]/P6/2015-3

Intitulé du projet : Déformation d'une membrane

Type de projet : *Bibliographie, modélisation*

Objectifs du projet : L'objectif de ce projet est de modéliser la déformation d'une membrane élastique rectangulaire. Pour cela, notre étude a été décomposée en trois parties : une modélisation physique, une résolution mathématique et une simulation numérique.

1. INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN
DÉPARTEMENT SCIENCES ET TECHNIQUES POUR L'INGÉNIEUR
685 AVENUE DE L'UNIVERSITÉ BP 08- 76801 SAINT-ETIENNE-DU-ROUVRAY
TÉL : 33 2 32 95 66 21 - FAX : 33 2 32 95 66 31

Table des matières

Introduction

Le but de ce projet est de modéliser la déformation d'une membrane élastique rectangulaire. Pour se faire, notre étude a été décomposée en trois parties : une modélisation physique, une résolution mathématique et une simulation numérique. La modélisation physique définit les différents paramètres et les conditions initiales de l'étude. La résolution mathématique vise à trouver les solutions de notre problème posé grâce à la modélisation physique. Pour arriver au résultat voulu, il existe une résolution implicite ou approchée et une résolution explicite ou exacte ; toutes deux traitant de méthodes différentes. L'outil informatique sera également important pour l'aboutissement de ce projet. En effet, un programme informatique en Pascal permettra d'approcher la solution exacte et de la comparer avec celle trouvée par d'autres méthodes. La modélisation numérique consiste à représenter les solutions de la modélisation. Elle permet d'avoir une vue graphique de l'objet étudier.

Chapitre 1

Travail réalisé et résultats

1.1 Modélisation physique

La membrane est rectangulaire, horizontale et élastique. On la met sous tension en la fixant par les bords et en exerçant une force transversale. Sa surface est représentée par une fonction de deux variables f, solution d'une équation de Poisson de la forme suivante :

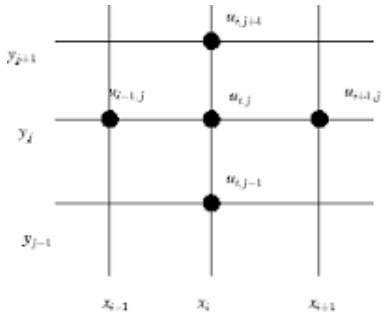
$$-a \triangle f = P$$

où P est le terme source et a est le coefficient de contusion.

En posant $f_{ij} = f(x_i, y_j)$, on discrétise la membrane. Désignons par L_x sa longueur et L_y sa largeur. La membrane sera désignée par $\Omega = [0, L_x][0, L_y]$. N et M sont le nombre de divisions respectif des côtés de la membrane. On obtient alors deux pas de discrétisation h et k s'exprimant tel que :

$$h = \frac{L_x}{N+1} \text{ et } k = \frac{L_y}{M+1}$$

On a : $x_i = ih$ et $y_j = jk$ où $i, j = 0, .., 1$.



1.2 Résolution mathématique

La résolution mathématique du problème peut se faire de deux façons : une résolution approchée et un résolution exacte.

1.2.1 Résolution approchée

L'équation de Poisson est écrite comme suit :

$$-c_1 \triangle u = f \tag{1.1}$$

Par discrétisation de la grille $\Omega = [0, L_x][0, L_y]$, on a $x_i = ih$ et $y_j = jk$ où h et k sont les pas définis dans lors de la modélisation physique.
On travaillera avec la condition initiale $|_{\partial\Omega} = 0$; c'est à dire qu'il n'y a aucune contrainte aux bords de la membrane.

Méthode des différences finies

L'objectif de cette section est de faire une résolution locale pour un couple $(x_i; y_j)$ considéré en approchant la solution par des dérivées partielles secondes de u par des développements de Taylor. Cette manière de procéder utilise la méthode des différences finies.

Considérons un accroissement local en x.

$$u(x_{i+1}; y_j) = u(x_i; y_j) + h\frac{\partial u}{\partial x}(x_i; y_j) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i; y_j) + h^2\varepsilon(h; y_j) \tag{1.2}$$

$$u(x_{i-1}; y_j) = u(x_i; y_j) - h\frac{\partial u}{\partial x}(x_i; y_j) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i; y_j) + h^2\varepsilon(h; y_j) \tag{1.3}$$

En additionnant les équations *(1.2)* et *(1.3)* et en négligeant les restes, on obtient :

$$\frac{1}{h^2}[u(x_{i+1}; y_j) - 2u(x_i; y_j) + u(x_{i-1}; y_j)] \simeq \frac{\partial^2 u}{\partial x^2}(x_i; y_j) \tag{1.4}$$

Par analogie, considérons un accroissement local en y.

$$u(x_i; y_{j+1}) = u(x_i; y_j) + k\frac{\partial u}{\partial y}(x_i; y_j) + \frac{k^2}{2}\frac{\partial^2 u}{\partial y^2}(x_i; y_j) + k^2\varepsilon(x_i; k) \tag{1.5}$$

$$u(x_i; y_{j-1}) = u(x_i; y_j) - k\frac{\partial u}{\partial y}(x_i; y_j) + \frac{k^2}{2}\frac{\partial^2 u}{\partial y^2}(x_i; y_j) + k^2\varepsilon(x_i; k) \tag{1.6}$$

En additionnant les équations *1.5* et *1.6* et en négligeant les restes, on obtient :

$$\frac{1}{k^2}[u(x_i; y_{j+1}) - 2u(x_i; y_j) + u(x_i; y_{j-1})] \simeq \frac{\partial^2 u}{\partial y^2}(x_i; y_j) \tag{1.7}$$

Or on sait que $-c_1 \triangle u = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f$

$\Leftrightarrow -c_1 \left(\frac{1}{h^2}\left(u(x_{i+1}; y_j) - 2u(x_i; y_j) + u(x_{i-1}; y_j) + \frac{1}{k^2}\left(u(x_i; y_{j+1}) - 2u(x_i; y_j) + u(x_i; y_{j-1})\right)\right)\right) = f_{i,j}$

Pour la suite de la résolution, nous allons prendre $M = N$, $h = k$ et $c_1 = 1$. On obtient des équations linéaires sous la forme :

$$-u_{i+1,j} - u_{i,j+1} + 4u_{i,j} - u_{i-1,j} - u_{i,j-1} = h^2 f_{i,j}$$

Pour $i, j = 1..N-1$ (les conditions initiales veulent que $f = 0$ aux frontières de la membrane).

Représentons ces équations linéaires sous forme matricielle, c'est-à-dire $Au = b$ où A est la matrice des coefficients des équations linéaires, u est un vecteur des points de la grilles et $b = f$.

On obtient alors une matrice carrée tridiagonale par blocs de dimension $(N-1)^2$ de la forme :

$$\begin{pmatrix} G & -I & 0 & \dots & 0 \\ -I & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -I \\ 0 & \dots & 0 & -I & G \end{pmatrix}$$

Le bloc I est la matrice identité de dimension $N-1$. Le bloc G est explicité ci-dessous :

$$\begin{pmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{pmatrix}$$

Pour mieux visualiser la représentation matricielle, donnons des valeurs numériques aux paramètres du problème. Prenons $N = 4$, $h = \frac{1}{4}$ et $f = 16$.
Les équations linéaires seront alors de la forme :

$$-u_{i+1,j} - u_{i,j+1} + 4u_{i,j} - u_{i-1,j} - u_{i,j-1} = 1$$

Construisons d'abord la matrice A en donnant des valeurs à i et j dans l'équation ci-dessus.

$i, j = 1, 1$	$-u_{2,1} - u_{1,2} + 4u_{1,1} - u_{0,1} - u_{1,0} = 1$
$i, j = 2, 1$	$-u_{3,1} - u_{2,2} + 4u_{2,1} - u_{1,1} - u_{2,0} = 1$
$i, j = 3, 1$	$-u_{4,1} - u_{3,2} + 4u_{3,1} - u_{2,1} - u_{3,0} = 1$
$i, j = 1, 2$	$-u_{2,2} - u_{0,2} - u_{1,1} - u_{1,3} + 4u_{1,2} = 1$
$i, j = 2, 2$	$-u_{3,2} - u_{1,2} - u_{2,1} - u_{2,3} + 4u_{2,2} = 1$
$i, j = 3, 2$	$-u_{4,2} - u_{2,2} - u_{3,1} - u_{3,3} + 4u_{3,2} = 1$
$i, j = 1, 3$	$-u_{2,3} - u_{0,3} - u_{1,2} - u_{1,4} + 4u_{1,3} = 1$
$i, j = 2, 3$	$-u_{3,3} - u_{1,3} - u_{2,2} - u_{2,4} + 4u_{2,3} = 1$
$i, j = 3, 3$	$-u_{4,3} - u_{2,3} - u_{3,2} - u_{3,4} + 4u_{3,2} = 1$

NB : Les termes aux frontières de la membrane sont nuls.

Remplissons maintenant la matrice.

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

On voit bien l'illustration de la matrice tridiagonale par blocs de dimension $9 = (N-1)^2$

Élimination de Gauss

Le système auquel on va s'intéresser ici est le système $AX = B$ avec A une matrice de taille $n * n$, X un vecteur à n composantes à déterminer, B un vecteur à n composantes connues.Il y a quatre méthodes pour résoudre ce système. Il s'agit de la méthode d'élimination de Gauss,la méthode par décomposition LU, la méthode de Cholesky et la méthode de la factorisation QR.On s'intéresse ici seulement à l'élimination de Gauss et à la décomposition LU.

Cette méthode a pour but de résoudre un système linéaire dont la matrice est triangulaire supérieure, on obtient ensuite la solution par simple remontée. Soit un système ayant n équations avec a_{ij} où i,j appartiennent à $[1, n]$, des coefficients connus,$(x_1, ..., x_n)$ un vecteur à déterminer,$(b_1, ..., b_n)$ un vecteur donné ,du type

$$\begin{cases} a_{11} * x_1 + a_{12} * x_2 + \ldots + a_{1n} * x_n = & b_n \\ a_{21} * x_1 + a_{22} * x_2 + \ldots + a_{2n} * x_n = & b_n \\ a_{31} * x_1 + a_{32} * x_2 + \ldots + a_{3n} * x_n = & b_n \\ a_{n1} * x_1 + a_{n2} * x_2 + \ldots + a_{nn} * x_n = & b_n \end{cases} \tag{1.8}$$

Ce système est appelé " système d'équations linéaires à coefficients réels". La méthode de Gauss, consiste à éliminer x_1 des lignes $2, 3, 4, \ldots, n$ (On notera L_2,L_3,...,L_n) , puis x_2 des lignes $3, 4, \ldots, n$, on le fait par réitération donc finalement,on élimine x_{n-1} de la ligne n. On obtient alors aisément la valeur de x_n et on en déduit les autres valeurs en remontant les lignes. Le système (1.8) s'écrit sous forme matricielle $Ax = b$:

Élimination de Gauss :
Étape 1 :

On pose $A_1 = A$ et on note $a_{i,j}$, $1 \leq i \leq 4$, $1 \leq j \leq 4$ l'élément (i, j) de la matrice A. Lorsqu'il est non nul, on nomme pivot (de la première étape) l'élément a_{11}.

Étape 2 : On réitère la même méthode pour les autres lignes $(L_2, L_3, \ldots, L_n)$,mais ici,on doit faire attention au pivot de chaque ligne,on garde toujours cette ligne-là comme le pivot et les autres lignes s'annulent en une certaine valeur de x par rapport au pivot. Par exemple,

considérons le système suivant

$$\begin{cases} 2x + 3y + 3z + t = & 15 \\ -4x - 7y + 3z + 2t = & 3 \end{cases} \tag{1.9}$$

On met la deuxième équation en pivot. On élimine x de L_2,on a $L_2' \rightarrow L_2 + 2 * L_1$, et on obtient $2x + 3y + 3z + t = 15$ (fin de l'exemple).

$$\begin{cases} 2x + 3y + 3z + t = & 15 \\ -y + 9z + 4t = & 33 \end{cases} \tag{1.10}$$

On réitère cette méthode pour le reste de la résolution de l'équation. Finalement, on obtient le système suivant :

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & \dots & a'_{1n} \\ 0 & a'_{22} & a'_{23} & \dots & a'_{2n} \\ 0 & 0 & a'_{33} & \dots & a'_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a'_{nn} \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_n \end{pmatrix}$$

Étape 3

On voit que la matrice obtenue est maintenant triangulaire supérieure, et on peut aisément résoudre le système par remonte les lignes.

Décomposition en LU Présentation : La décomposition LU d'une matrice A de taille $n * n$ est une méthode de décomposition très utilisée dans le domaine de l'analyse numérique. La décomposition LU, qui consiste à déterminer une matrice L (respectivement U) triangulaire inférieure (respectivement supérieure) telle que $A = L * U$. On impose de plus que la diagonale de L soit remplie par des 1. Ceci permet d'insérer L et U dans une seule matrice de taille $n * n$.
Une fois cette décomposition faite, on peut résoudre ce système facilement : on résout tout d'abord $U * X = y$, puis ensuite $L * y = B$. Le calcul de cette résolution de systèmes est beaucoup plus évident quand les matrices sont triangulaires,on peut remonter les lignes,comme fait précédemment avec l'élimination de Gauss,on résout tout d'abord y et après,sachant y,on utilise la même technique avec le système $UX = y$.
Considérons l'exemple suivant :

$$A = \begin{pmatrix} 4 & -9 & 2 \\ 2 & -4 & 4 \\ -1 & 2 & 2 \end{pmatrix}$$

Nous avons alors $l_{21} = \frac{2}{4} = 0.5$ et $l_{31} = \frac{-1}{4} = -0.25$.Ce qui nous donne :

$$A = \begin{pmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 3 \\ 0 & -0.25 & 0.5 \end{pmatrix}$$

Nous continuons l'élimination en posant $l_{32} = \frac{-0.25}{0.5} = -0.5$ et obtenons que

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.25 & -0.5 & 1 \end{pmatrix}$$

et

$$U = \begin{pmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 3 \\ 0 & 0 & 4 \end{pmatrix}$$

Nous avons donc obtenu la décomposition LU de cet exemple.

1.2.2 Résolution exacte

Méthode de séparation des variables

L'équation de Poisson est la suivante :

$$-c_1 \Delta u = f \qquad \text{où } \Delta u \text{ est l'opérateur Laplacien}$$

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad \text{et} \qquad 0 < x < L_x \qquad 0 < y < L_y$$

conditions aux limites :

$$u(0, y) = u(L_x, y) = 0$$
$$u(x, 0) = u(x, L_y) = 0$$

On va utiliser la méthode de séparation des variables pour trouver la solution exacte de l'équation sous les conditions initiales $|_{\partial\Omega} = 0$.

On cherche des solutions de la forme :

$$u(x, y) = X(x)Y(y)$$

On fait les doubles dérivées partielles :

$$\frac{\partial^2 u}{\partial x^2}(x, y) = \frac{\partial^2 X(x)}{\partial x^2} Y(y)$$
$$\Rightarrow \frac{\partial^2 u}{\partial y^2}(x, y) = \frac{\partial^2 Y(y)}{\partial y^2} X(x)$$
$$\Rightarrow f = -c_1 \left(\frac{\partial^2 X(x)}{\partial x^2} Y(y) + \frac{\partial^2 Y(y)}{\partial y^2} X(x)\right)$$

par l'équation de Helmholtz :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -q^2 u(x, y)$$

$$\frac{\partial^2 X(x)}{\partial x^2} Y(y) + \frac{\partial^2 Y(y)}{\partial y^2} X(x) = -q^2 X(x).Y(y) \qquad (1)$$

Comme les deux fonctions ont des variables différentes, elles doivent être constantes. En divisant par $X(x)Y(y)$ l'équation (1), on peut écrire :

$$\frac{\partial^2 X(x)}{\partial x^2}\frac{1}{X(x)} = -q^2 - \frac{\partial^2 Y(y)}{\partial y^2} = -\eta^2 \text{ et } \frac{\partial^2 Y(y)}{\partial y^2}\frac{1}{Y(y)} = \eta^2 - q^2 = -\nu^2.$$

D'où $q^2 = \eta^2 + \nu^2$

On obtient donc des équations différentielles ne faisant intervenir qu'une seule variable et qui sont donc simples à résoudre :

$$\frac{\partial^2 X(x)}{\partial x^2} + \eta^2 X(x) = 0 \text{ (2)}$$

$$\frac{\partial^2 Y(y)}{\partial y^2} + \nu^2 Y(y) = 0 \text{ (3)}$$

Nous allons détailler la solution de l'équation (2) puis nous donnerons celles des équations (3) qui se résolvent de manière analogue.
On commence par calculer le déterminant : $\Delta = -4\eta^2$. On s'aperçoit que $\Delta < 0$.

L'équation caractéristique $p + \eta^2 p = 0$ possède donc deux solutions imaginaires et une solution réelle qui sont :

$p_1 = 0$, $p_2 = \frac{-\eta^2 + 2\eta.i}{2}$ et $p_3 = \frac{-\eta^2 - 2\eta.i}{2}$

On obtient alors :

$$X(x) = (acos(\eta.x) + bsin(\eta.x))e^{0x}$$

$$X(x) = acos(\eta.x) + bsin(\eta.x)$$

De même,

$$Y(y) = ccos(\nu.y) + dsin(\nu.y)$$

On peut maintenant obtenir une forme générale, que l'on nomme $u(x, y)$, du déplacement de l'onde en fonction de x et y :

$$u(x, y) = X(x)Y(y)$$

$$u(x, y) = (acos(\eta.x) + bsin(\eta.x))(ccos(\nu.y) + dsin(\nu.y))$$

$$u(x, y) = accos(\eta.x)cos(\nu.y) + adcos(\eta.x)sin(\nu.y) + bcsin(\eta.x)cos(\nu.y) + bdsin(\eta.x)sin(\nu.y)$$

On pose quatre constantes que l'on déterminera grâce aux conditions initiales :

$$C_1 = bd \text{ et } C_2 = bc \text{ et } C_3 = ad \text{ et } C_4 = ac.$$

Pour les conditions initiales nous allons considérer que la membrane est fixée aux bords.

En utilisant la condition aux limites :

$$0 = u(0,y) = X(O)Y(y) = u(L_x,y) = X(L_x)Y(y)$$

$$\Rightarrow X(0) = X(L_x) = 0$$

$$0 = u(x,0) = X(x)Y(0) = u(x,L_y) = X(x)Y(Ly)$$

$$\Rightarrow Y(0) = Y(b) = 0$$

.

On commence en prenant $x = 0$,
D'où

$$X(0)Y(y) = C_3 cos(\nu.y) + C_4 sin(\nu.y)$$

Or

$$u(x,y) = 0$$

donc

$$C_3 cos(\nu.y) + C_4 sin(\nu.y) = 0$$

On en conclut que pour tout y on a $C_3 = C_4 = 0$

· On prend maintenant $x = L_x$

$$X(L_x)Y(y) = C_1 sin(\eta.L_x) sin(\nu.y) + C_2 sin(\eta.L_x) cos(\eta.y)$$

Cette relation doit être nulle d'où :

$$sin(\eta.L_x)(C_1 sin(\nu.y) + C_2 cos(\eta.y)) = 0$$

Deux solutions sont possibles :
- soit $C_1 = C_2 = 0$ ce qui est absurde
- $kL_x = 0 + m\pi$ soit $sin(\eta.L_x) = 0$

Donc

$$\eta_m = \frac{m\pi}{L_x} \text{ avec } m \in N$$

· On fait de même avec $y = 0$
on obtient :

$$X(x)Y(y) = C_2 sin(\eta.x) = 0$$

donc pour tout x on a $C_2 = 0$

Et enfin on sait que $y = L_y$ et en faisant de même qu'avec $x = L_x$ on obtient :

$$sin(\nu.L_x) = 0$$

D'où

$$\nu.L_y = n\pi$$

Donc

$$\nu_n = \frac{n\pi}{L_y}$$

En bref, on obtient les résultats suivants :

$$X_m(x) = sin(\eta_m x) \quad \text{où} \quad \eta_m = \frac{m\pi}{L_x} \quad m = 1, 2, 3...$$
$$Y_n(y) = sin(\nu_n y) \quad \text{où} \quad \nu_n = \frac{n\pi}{L_y} \quad n = 1, 2, 3...$$

On revient à l'équation de Poisson et on prend le coefficient $c_1 = 1$ par condition. L'équation de Helmholtz (1) devient de la forme :

$$-\Delta u(x, y) = \lambda u(x, y) = f(x, y)$$

où on a pris $\lambda = -q^2$

En remplaçant dans l'équation par les fonctions $X_m(x)$ et $Y_n(y)$ et en prenant la fonction propre normalisée $u_{n,m}$, on obtient :

$$u_{n,m} = \frac{2}{\sqrt{ab}}.sin(\eta_m x).sin(\nu_n y)$$
$$f(x, y) = \sum_n \sum_m b_{n,m} u_{n,m}(x, y)$$

En utilisant les séries de Fourier, on obtient le coefficient de Fourier $b_{n,m}$ où $C = \frac{-2}{sqrtab}$.

$$b_{n,m} = C \iint_0^L f(x, y)\, dy\, dx$$

On obtient la solution exacte :

$$u(x, y) = \sum_{n,m} \frac{b_{n,m}}{\lambda_{n,m}} u_{n,m}(x, y).$$

Résolution sur MATLAB

Voici le code de la résolution du système matricielle d'équations linéaires dans $\Re^9$ sur MATLAB.

```
A=[4 -1 0 -1 0 0 0 0 0
  -1 4 -1 0 -1 0 0 0 0
  0 -1 4 0 0 -1 0 0 0
  -1 0 0 4 -1 0 -1 0 0
  0 -1 0 -1 4 -1 0 -1 0
  0 0 -1 0 -1 4 0 0 -1
  0 0 0 -1 0 0 4 -1 0
  0 0 0 0 -1 0 -1 4 -1
  0 0 0 0 0 -1 0 -1 4 ]

B=ones(9,1)
u= A\B
```

La solution obtenue est :

```
u =

    0.6875
    0.8750
    0.6875
    0.8750
    1.1250
    0.8750
    0.6875
    0.8750
    0.6875
```

1.3 Simulation numérique

1.3.1 Méthode de Cholesky - Programme Pascal

Algorithme

Sachant que nous devions résoudre une équation du type $Ax = b$, la méthode de Cholesky nous paraissait toute appropriée ! De plus, le logiciel Matlab l'utilise lors de l'appel de la commande $x = b/A$. Commençons par expliquer cette méthode. A est une matrice symétrique définie positive et x un vecteur :

$$A = A^T A$$
$$X^T Ax > 0 \qquad \forall x \neq 0$$

Le théorème de la décomposition de Cholesky dit que, pour une telle matrice A, il existe R tel que :

$$A = RR^T$$

Ainsi, on calcul R par identification en exprimant les r_{ij} en fonction des a_{ij} :

$$a_{ij} = \sum_{k=1}^{n} R_{ik} R_{kj}^T \qquad i \leqslant j$$

$$R = \begin{pmatrix} r_{11} & 0 & \dots & 0 & \dots & 0 \\ r_{21} & r_{22} & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ r_{i1} & r_{i2} & \dots & r_{ii} & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ r_{n1} & r_{n2} & \dots & r_{ni} & \dots & r_{nn} \end{pmatrix}$$

On identifie d'abord la 1er ligne de A (i= 1) :

$$a_{11} = r_{11} r_{11} = r_{11}^2 \rightarrow r_{11} = \sqrt{a_{11}}$$
$$a_{1j} = r_{11} r_{j1} \rightarrow r_{j1} = \frac{a_{1j}}{r_{11}} \qquad j = 2, \dots n$$

Et de même pour i = 2,. . . ,n :

$$a_{ii} = \sum_{k=1}^{i-1} r_{ik}^2 + r_{ii}^2 \rightarrow r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ik}^2}$$
$$a_{ij} = \sum_{k=1}^{i-1} r_{ik} r_{jk} + r_{ii} r_{ji} \rightarrow r_{ji} = \frac{1}{r_{ii}} \left[\sum_{k=1}^{i-1} r_{ik} r_{kj} \right]$$

On réécrit le système $A = RR^T$:

$$RR^T x = b$$

On peut donc décomposer le système de la manière suivante :

$$\begin{array}{ll} Ry = B & systeme\ triangulaire\ inferieur \\ R^T x = y & systeme\ triangulaire\ superieur \end{array}$$

On peut ainsi résoudre $Ax = b$ en résolvant les deux équations précédentes. Pour cela, nous procéderons par ordre décroissant des indices, comme dans la méthode de Gauss, pour la première puis par ordre croissant pour la première.

Programme Pascal

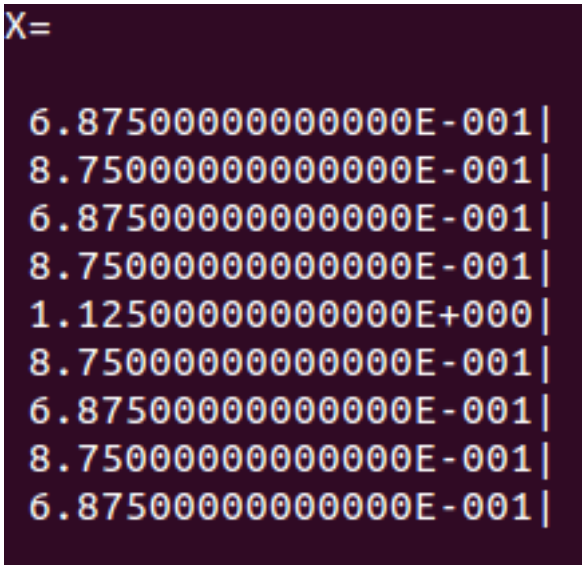
Nous avons modifié un code généreusement donné par l'enseignant en charge de notre groupe de manière à le faire fonctionner dans notre situation, c'est-à-dire avec des matrices tridiagonales par bloc. Le programme offrait à l'origine la possibilité de rentrer une matrice case par case, ce qui n'était évidemment pas du tout adapté. Par conséquent, nous avons ajouté la possibilité de générer une telle matrice en saisissant tout simplement sa dimension et les valeurs de chacune de ses diagonales (différentes de 0). Le programme final, qui se nomme $deformationMembrane.pas$ a pour but de résoudre un système du type $Ax = b$ en utilisant la méthode de Cholesky. On réutilise bien sur la même matrice A et le même vecteur b que dans la partie "Résolution exacte".

```
Rang de la matrice? rang 50 maximum et entier
9
Valeur pour la diagonale du bloc central : 4
Valeur pour les diagonales sup et inf du bloc central : -1
Valeur pour les diagonales des blocs sup et inf : -1
tapez 1 pour utiliser un vecteur b dont toutes les composantes valent 1 ou tapez
 un autre chiffre pour trouver x = 1
1

A =

 4.0 | -1.0 |  0.0 | -1.0 |  0.0 |  0.0 |  0.0 |  0.0 |  0.0
-1.0 |  4.0 | -1.0 |  0.0 | -1.0 |  0.0 |  0.0 |  0.0 |  0.0
 0.0 | -1.0 |  4.0 |  0.0 |  0.0 | -1.0 |  0.0 |  0.0 |  0.0
-1.0 |  0.0 |  0.0 |  4.0 | -1.0 |  0.0 | -1.0 |  0.0 |  0.0
 0.0 | -1.0 |  0.0 | -1.0 |  4.0 | -1.0 |  0.0 | -1.0 |  0.0
 0.0 |  0.0 | -1.0 |  0.0 | -1.0 |  4.0 |  0.0 |  0.0 | -1.0
 0.0 |  0.0 |  0.0 | -1.0 |  0.0 |  0.0 |  4.0 | -1.0 |  0.0
 0.0 |  0.0 |  0.0 |  0.0 | -1.0 |  0.0 | -1.0 |  4.0 | -1.0
 0.0 |  0.0 |  0.0 |  0.0 |  0.0 | -1.0 |  0.0 | -1.0 |  4.0
```

La partie du code nous permettant de générer des matrices tridiagonales par bloc a également été adapté de manière à créer un programme $genMatlab2.pas$ nous permettant de les écrire sous forme de matrices Matlab dans un terminal. Il était ainsi possible de faire nos calculs sur Matlab avec des matrices de grande dimension.

Comparaison des résultats obtenus avec Matlabl

Voici les résultats données par notre programme :



Nous retrouvons les mêmes valeurs que celles obtenues avec Matlab.

Comparaison des résultats obtenus avec Maple

Au tout début de notre raisonnement, les longueurs et largeur de la membrane ont été divisé par N, afin d'obtenir un ensemble de points se situant sur la membrane. Dans notre cas, N = 4, nous avons :

$$u = \begin{pmatrix} u_{1;1} \\ u_{2;1} \\ u_{3;1} \\ u_{1;2} \\ u_{2;2} \\ u_{3;2} \\ u_{1;3} \\ u_{2;3} \\ u_{3;3} \end{pmatrix}$$

avec f la fonction donnant le déplacement verticale de la membrane en fonction des coordonnées $(x; y)$ d'un point de celle-ci. Les points ayant 0 ou 4 comme ordonnée ou abscisse ne sont pas présents puisque le déplacement de la membrane en ces points valent 0 (points du bord de la membrane). Nous pouvons dessiner le graphique suivant pour décrire la répartition des composantes du vecteur u sur la membrane :

A l'aide du logiciel Maple et d'un code fournit par notre professeur (fournit en annexe), nous avons pu vérifier les valeurs du vecteur u obtenues à l'aide de Matlab et de notre programme Pascal. Soit f la fonction qui donne le déplacement vertical pour des coordonnées $(x; y)$ de la membrane. Notre code utilise cette fonction pour trois points :

$$\begin{array}{rl} u_{1;1} = 0.6875 & f(\frac{1}{4}; \frac{1}{4}) = 0.6570 \\ u_{2;2} = 1.113 & (\frac{1}{2}; \frac{1}{2}) = 1.314 \\ u_{3;3} = 0.6875 & f(\frac{3}{4}; \frac{3}{4}) = 0.6570 \end{array}$$

On retrouve donc nos valeurs environ à 10^{-2} près. On peut donc en conclure que l'ensemble de nos résultats est cohérents. Remarquons que le code Maple retrouve également l'équation que nous avons écrite dans la partie $Rsolution\ exacte$. (Équation numéro 6 dans le code Maple en annexe)

Chapitre 2

Méthodologie et organisation du travail

Pour mener à bien ce projet, il nous aura mettre en place une organisation des tâches au sein du groupe. De manière globale, tous les membres du groupe ont participé à toutes les étapes du projet. Cependant, certains d'entre nous étant plus à l'aise avec certaines notions ont choisi de traiter celles-ci davantage. Ainsi donc, pour la résolution mathématique, Marème s'est chargée de la résolution approchée utilisant la méthode des différentielles, ainsi que la résolution exacte sur MATLAB. Par la suite, Runli a traité la résolution par l'élimination de Gauss. La solution exacte a été traitée par Ivan et Métodi. Ils ont également participé à la simulation numérique. Enfin, toute l'implémentation en Pascal s'aidant de la méthode de Cholesky a été prise en main par Elliot. Les tâches se sont réparties de la même manière pour la rédaction de ce rapport. Aussi, il nous a été très bénéfique de définir des objectifs hebdomadaires après chaque séance encadrée. Cela nous a permis d'avancer continuellement dans notre travail. Enfin, l'assistance de notre maître de projet M. Gleyse nous a été d'une grande utilité pour la bonne réalisation de cette étude.

Conclusion et perspectives

Cette étude sur la déformation a été axée autour de trois points : l'aspect physique, l'aspect mathématique et celui informatique. On a réussi à dégager les propriétés physiques caractérisant la membrane afin de pouvoir la modéliser mathématiquement. Cette modélisation a fait appel plusieurs notions : la méthode des différences finies, la méthode par élimination de Gauss, la résolution d'équations linéaires sous formes matricielles, la méthode de Cholesky pour les équations linéaires. Des compétences en programmation ont également été nécessaires pour l'élaboration du programme, en Pascal, qui permet lui même de déterminer une solution exacte. Enfin, nous avons été introduit au logiciel Maple qui permet de faire des simulations numériques de la déformation d'une membrane.
Au delà des connaissances scientifiques théoriques, ce projet nous a été profitable sur le plan personnel et humain. Il nous as permis de développer des capacités d'organisation, de communication et d'auto-discipline ; qualités indispensable à tout ingénieur.
Enfin, les résultats de ce projet pourrait être une base pour une application pratique. En effet, les baffles d'une enceinte sonore sont protéger par une membrane qui se déforme sous l'influence des ondes sonores. Il serait donc intéressant d'étudier ce cas précis de déformation de membrane.

Bibliographie

[1] NOM, Prénom *Titre*, edition, date.

[2] NOM DES AUTEURS, "*Titre de l'article*", Titre du journal, Volume, pages, année.

[3] http://www-ljk.imag.fr/membres/Guillaume.James/sujet_TP_MN_09.pdf (Valide à la date du 04/04/2008)

bibitemLien Internet 01 http://www.phys.ens.fr/~dormy/Palais/node1.html (Valide en Janvier 1996)

bibitemLien Internet 01 http://www.ecf.utoronto.ca/~bentz/mhome.shtml (Valide en 2012)

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.79&rep=rep1&type=pdf (Date de validité indéterminable)

http://link.springer.com/article/10.1007%2FBF01177189

http://iecl.univ-lorraine.fr/~Nicolas.Champagnat/Harmo.pdf (Valide en octobre 2010)

https://www.ljll.math.upmc.fr/~ledret/M1/ComplementsM1ApproxEDP.pdf (Valide en janvier 2009)

http://www.ufrmeca.univ-lyon1.fr/~buffat/COURS/COURSDF_HTML/node32.html (Valide en avril 2008)

http://www.math.iit.edu/~fass/Notes461_Ch7.pdf(Valide en automne 2011)

http://www-solar.mcs.st-and.ac.uk/~alan/MT3601/Fundamentals/node60.html (Valide en juin 2000)

http://math.univ-lyon1.fr/~saleh/Docs/MEF/CoursUFE_KhaledSaleh.pdf (Valide en décembre 2011)

http://www.ufrmeca.univ-lyon1.fr/~buffat/COURS/COURSDF_HTML/node32.html (Valide en avril 2008)

http://www.polytech-lille.fr/cours-algos-calcul-scientifique/Cholesky.swf

Annexe A

Listings des programmes réalisés

Trois programmes ont été utiles dans le cadre de ce projet :

— genMatLab2.pas
— deformationMembrane.pas
— codeMaple.mw
Les codes de ces programmes sont joints en annexe.

Annexe B

Propositions de sujets de projets (en lien ou pas avec le projet réalisé)

Annexe C

Codes

```
program genMatlab2;


const q=100 ;
Type matrice = array [1..q,1..q] of Real; matrice2 = array [1..q] of Real ;
var A : matrice; i,j,p, rang : Integer ; valeur, valeur2 : Real;

procedure initialiserMatrice(dim : integer ; var A : Matrice);
var i, j : Integer;
begin
        For i:= 1 to dim do
                For j := 1 to dim do
                        A[i, j] := 0;
end;

procedure affichage_matrice (mat: matrice);
var i,k:Integer ;
begin
        For i:= 1 to p do
                begin
                        writeln();
                        For k:=1 to p-1 do
                                write(mat[i,k]:4:1,' | ') ;
                        write(mat[i,p]:4:1) ;
                end;
end;

procedure affichage_matrice2 ( mat: matrice2);
var k:Integer ;
begin
                        writeln();
                        For k:=1 to p do
                                writeln(mat[k],'|') ;
end;

begin
```

```
        writeln('Rang : ');
        readln(p);

        initialiserMatrice(p, A);

        write('Valeur pour la diagonale du bloc central : ');
        readln(valeur);
        For i := 1 to p do
                A[i, i] := valeur;

        write('Valeur pour les diagonales sup et inf du bloc central : ');
        readln(valeur2);


        For i := 1 to p do
                if (i MOD 3) <> 0 then
                        begin
                                A[i + 1, i] := valeur2;
                                A[i, i + 1] := valeur2;
                        end
                else
                        A[i, i] := valeur;

        write('Valeur pour les diagonales des blocs sup et inf : ');
        readln(valeur);

        rang := 4;
        For i := 1 to (1 + p - rang) do
                begin
                        A[i, rang + i - 1] := valeur;
                        A[rang + i - 1, i] := valeur;
                end;


        writeln();
        write('[');
        For i := 1 to p do
                begin
                        For j := 1 to p do
                                write(A[i, j]:2:2, ' ');
                        If i < p then
                                write('; ');
                end;
        write(']');

end.
```

```
program deformationMembrane;

const q=50 ;
```

```
Type matrice = array [1..q,1..q] of Real; matrice2 = array [1..q] of Real ;
var A,L,Lt : matrice; b,Y,X : matrice2; i,j,m,p, rang: Integer ; K, N, valeu

procedure initialiserMatrice(dim : integer ; var A : Matrice);
var i, j : Integer;
begin
        For i:= 1 to dim do
                For j := 1 to dim do
                        A[i, j] := 0;
end;

procedure affichage_matrice (mat: matrice);
var i,k:Integer ;
begin
        For i:= 1 to p do
                begin
                        writeln();
                        For k:=1 to p-1 do
                                write(mat[i,k]:4:1,' | ') ;
                        write(mat[i,p]:4:1) ;
                end;
end;

procedure affichage_matrice2 ( mat: matrice2);
var k:Integer ;
begin
                        writeln();
                        For k:=1 to p do
                                writeln(mat[k],'|') ;
end;



begin
        writeln('Rang de la matrice? rang ',q,' maximum et entier') ;
        readln(p);

        initialiserMatrice(p, A);


        write('Valeur pour la diagonale du bloc central : ');
        readln(valeur);
        For i := 1 to p do
                A[i, i] := valeur;

        write('Valeur pour les diagonales sup et inf du bloc central : ');
        readln(valeur2);

        For i := 1 to p do
                if (i MOD 3) <> 0 then
```

```
                        begin
                                A[i + 1, i] := valeur2;
                                A[i, i + 1] := valeur2;
                        end;


        write('Valeur pour les diagonales des blocs sup et inf : ');
        readln(valeur);
        rang := 4;
        For i := 1 to (1 + p - rang) do
                begin
                        A[i, rang + i - 1] := valeur;
                        A[rang + i - 1, i] := valeur;
                end;

writeln('tapez 1 pour utiliser un vecteur b dont toutes les composantes val
        readln(i);
        if i <> 1 then
                For i :=1 to p do
                        begin
                                K:= 0 ;
                                        For j := 1 to p do
                                                K := K + A[j, i];
                                b[i] := K ;
                        end
        else
                For i := 1 to p do
                        begin
                                b[i] := 1;
                        end;



        writeln();
        writeln('A = ');
        affichage_matrice(A);
        writeln();

        writeln();
        writeln('b = ');
        affichage_matrice2(b) ;
        writeln() ;

        For i := 1 to p do
                For j := 1 to p do
                        L[i,j] := 0 ;


        affichage_matrice( A);
```

```
L[1,1] := sqrt(A[1,1]);

affichage_matrice( L);

For i := 2 to p do
        For j := 1 to i do
                begin

                If j>1 then
                        begin
                                K:=0 ;
                                N:=0 ;
                                For m := 1 to j-1 do
                                        begin
                                                K := K + sqr(L[j,m]
                                                N := N + L[i,m]*L[j
                                        end;
                                If j = i then
                                        begin
                                                L[j,j] := sqrt( A[j
                                        end
                                Else
                                        if L[j, j] <> 0 then
                                                L[i,j] := (A[i,j] -
                                        else
                                                writeln('ERREUR : D
                        end
                Else
                        begin
                                If  j = i  then
                                L[j,j] := sqrt( A[j,j])
                                Else
                                        if L[j, j] <> 0 then
                                                L[i,j] := (A[i,j])/
                                        else
                                                writeln('ERREUR : D
                        end;
                end;


writeln('L =');
affichage_matrice( L);
writeln();

For i := 1 to p do
        For j := 1 to p do
                Lt[i,j] := L[j,i];
writeln();
writeln('Lt =');
affichage_matrice(Lt);
```

```
        writeln();

        Y[1] := b[1]/L[1,1] ;
        For i:= 2 to p do
                begin
                K := 0 ;
                For j := 1 to i-1 do
                        K := K + L[i,j]*Y[j] ;
                Y[i] := (b[i]-K)/L[i,i] ;
                end;
        writeln();
        writeln('Y = ');
        affichage_matrice2(Y) ;
        writeln()                          ;


        X[p] := Y[p]/Lt[p,p] ;

        For i := p-1 downto 1 do
                begin
                        K := 0 ;
                        For j := i to p do
                                K := K + Lt[i,j]*X[j] ;
                        X[i] := (Y[i]-K)/(Lt[i,i]);
                end;

        writeln('X=');
        affichage_matrice2( X) ;
        writeln();
        readln();
end.
```

```
#with(plots) :
#f:=piecewise(-1 ≤ x ≤ 1, x);
#plot(f, x=-1..1, scaling=constrained);
a := 1; b := 1; d := 16; assume(n, integer) : assume(m, integer) :
```

$$1$$

$$1$$

$$16 \tag{1}$$

$$unm := -\frac{2}{(a\cdot b)^{\frac{1}{2}}}\cdot\sin\left(\frac{n\cdot \mathrm{Pi}\cdot x}{a}\right)\cdot\sin\left(\frac{m\cdot \mathrm{Pi}\cdot y}{b}\right);\ \lambda nm := \pi^2\cdot\left(\frac{n^2}{a^2}+\frac{m^2}{b^2}\right);$$

$$-2\sin(n\sim\pi x)\sin(m\sim\pi y)$$

$$\pi^2\left(m\sim^2+n\sim^2\right) \tag{2}$$

$$bnm := -\frac{2}{(a\cdot b)^{\frac{1}{2}}}\cdot int\left(int\left(d\cdot\sin\left(\frac{n\cdot \mathrm{Pi}\cdot x}{a}\right)\cdot\sin\left(\frac{m\cdot \mathrm{Pi}\cdot y}{b}\right), x=0..a\right), y=0..b\right);$$

$$-\frac{32\left((-1)^{m\sim+n\sim}-(-1)^{m\sim}-(-1)^{n\sim}+1\right)}{n\sim\pi^2 m\sim} \tag{3}$$

```
bnm := simplify(bnm);
```

$$\frac{32\left((-1)^{1+m\sim+n\sim}+(-1)^{m\sim}+(-1)^{n\sim}-1\right)}{n\sim\pi^2 m\sim} \tag{4}$$

```
with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*] **(5)**

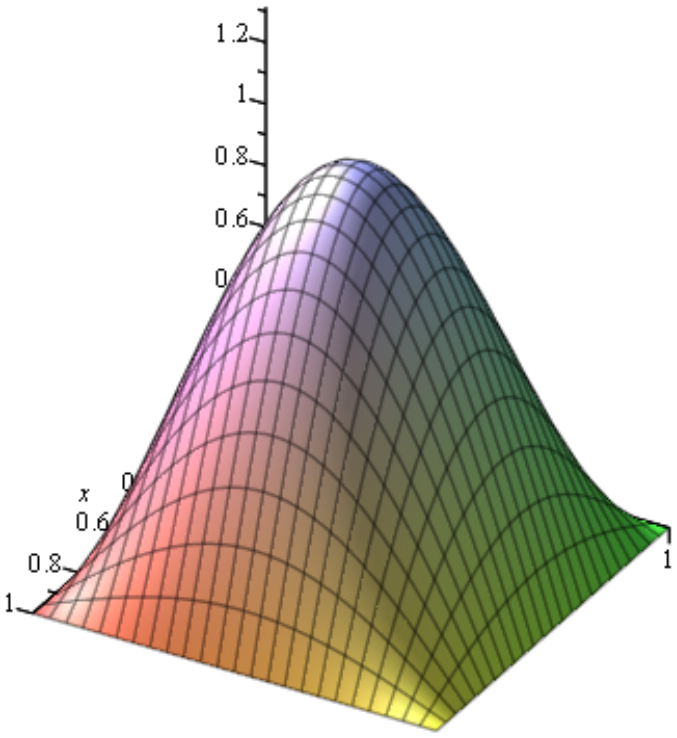
$$u11 := sum\left(sum\left(\frac{bnm}{\lambda nm}unm,\ n=1..1\right),\ m=1..1\right);$$

$$\frac{128\sin(\pi x)\sin(\pi y)}{\pi^4} \tag{6}$$

```
u11 := simplify(u11);
```

$$\frac{128\sin(\pi x)\sin(\pi y)}{\pi^4} \tag{7}$$

```
plot3d(u11, x=0..1, y=0..1, axes=NORMAL, orientation=[30, 60]);
```




```
w:=simplify(subs(x=1/2,y=1/2,u11));
```

$$\frac{128}{\pi^4} \tag{8}$$

$w := evalf(w);$

$$1.314045728 \tag{9}$$

$t := simplify(subs(x = 1/4, y = 1/4, u11));$

$$\frac{64}{\pi^4} \tag{10}$$

$t := evalf(t);$

$$0.6570228640 \tag{11}$$

$v := simplify(subs(x = 3/4, y = 3/4, u11));$

$$\frac{64}{\pi^4} \tag{12}$$

$v := evalf(v);$

0.6570228640 **(13)**