Projet 2

Dalissier Eric

1 Introduction

On a sous quelque hypothèse et raisonnement issue des mathémathiques probabilistes, l'équation de *Black-Scholes* sous la forme suivante :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\sigma_1^2 x^2}{2}\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\sigma_2^2 y^2}{2}\dfrac{\partial^2 u}{\partial x^2} - q\sigma_1\sigma_2 xy\dfrac{\partial^2 u}{\partial x \partial y} - rx\dfrac{\partial u}{\partial x} - ry\dfrac{\partial u}{\partial y} + ru = 0 \\ u(x,y,0) = (K-x-y)^+ \ sur \ \partial\Omega \\ u(L,y,t) = u(x,L,t) = 0 \ sur \ \partial\Omega \end{cases} \tag{1}$$

Ici, on pose Ω=$(0,L)^2$.
On doit écrire l'équation (1) sous forme divergentiele, c'est à dire de la forme :

$$\frac{\partial u}{\partial t} - \mathrm{div}(K\nabla u) + \mu\nabla u + \nu u = 0 \tag{2}$$

On déduit tout de suite que les termes : $\dfrac{\partial u}{\partial t}$ et u restent inchanger.
D'où :

$$\nu = r$$

Il nous reste donc à identifier :

$$\frac{\sigma_1^2 x^2}{2}\frac{\partial^2 u}{\partial x^2} - \frac{\sigma_2^2 y^2}{2}\frac{\partial^2 u}{\partial x^2} - q\sigma_1\sigma_2 xy\frac{\partial^2 u}{\partial x\partial y} - rx\frac{\partial u}{\partial x} - ry\frac{\partial u}{\partial y} \quad \text{avec} \quad -\mathrm{div}(K\nabla u) + \mu\nabla u$$

Pour cela nous avons développé $-\mathrm{div}(K\nabla u)$ puis identifié les dérivées partielles secondes. Cela nous a donné :

$$\mathrm{K} = \begin{pmatrix} \dfrac{\sigma_1^2 x^2}{2} & \dfrac{q\sigma_1\sigma_2 xy}{2} \\ \dfrac{q\sigma_1\sigma_2 xy}{2} & \dfrac{\sigma_2^2 y^2}{2} \end{pmatrix}$$

Comme les composantes de K dépendent de x et de y, ces dérivées par rapport a l'une où l'autre des variables ne sont pas nulles. C'est d'ailleurs ce qui nous permet de trouver ν.
On a :

$$\mu\nabla u - K'_{00}\tfrac{\partial u}{\partial x} - K'_{01}\tfrac{\partial u}{\partial y} - K'_{10}\tfrac{\partial u}{\partial x} - K'_{11}\tfrac{\partial u}{\partial y} = \begin{pmatrix} -rx & -ry \end{pmatrix}\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix}$$

On a après avoir remplacé et identifié :

$$\mu = \begin{pmatrix} x(-\dfrac{q\sigma_1\sigma_2}{2} - \sigma_1^2 + r) \\ y(-\dfrac{q\sigma_1\sigma_2}{2} - \sigma_2^2 + r) \end{pmatrix} \tag{3}$$

On va ensuite écrire sous forme variationnelle l'équation (3). Comme le temps et l'espace n'ont pas du tout le même comportement. On va se servir d'une approximation par différence finie pour la partie en temps de l'équation, et d'élement fini pour la partie en espace. Dans le calcule de la formulation variationnelle, on ne prendra pas en compte le terme $\frac{\partial u}{\partial t}$:
On prend $V = \{v \in H^1$ tel que $x\frac{\partial v}{\partial x} \in L^1(\Omega)$ et $v = 0$ si $x = L$ ou $y = L\}$.
On multiplie l'équation (3) par $v \in V$ puis on intégre en espace donc sur $[0, L]$.

$$\int_\Omega(-div(K\nabla u)v + \mu\nabla uv + \nu uv)\ dxdy = 0$$

On utilise une formule de Green :

$$\int_\Omega(K^t\nabla u\nabla v + \mu\nabla u\ v + \nu uv\ )dxdy - \int_\Gamma K\nabla u.\overrightarrow{n}vd\Gamma = 0$$

On veut que le terme en $\int_\Gamma K\nabla u.\overrightarrow{n}vd\Gamma$ soit nul. On a déjà la condition $v = 0$ en $x = L$ et $y = L$. Il nous reste donc :

$$\int_{\Gamma_{x=0}} K\nabla u.\overrightarrow{n}vd\Gamma \text{ et } \int_{\Gamma_{y=0}} K\nabla u.\overrightarrow{n}vd\Gamma$$

On pose donc la condition $\nabla u = 0$ pour $x = 0$ ou $y = 0$. Ainsi on retombe sur l'équation que l'on trouve par les distributions.
On a donc le problème variationnelle de (3) suivant :

Trouver $u\in$ V tel que pour tout $v\in$ V :
Avec $V = \{v \in \mathcal{L}^2(\Omega), \frac{\partial v}{\partial x} \in \mathcal{L}^2(\Omega), \frac{\partial v}{\partial y} \in \mathcal{L}^2(\Omega), \frac{\partial v}{\partial \overrightarrow{\nu}} \in \mathcal{L}^2(\Gamma_1)\bigcup\mathcal{L}^2(\Gamma_4)\}$

$$\begin{cases} \int_\Omega(\ \frac{v\partial u}{\partial t} + K^t\nabla u\nabla v + \mu\nabla u\ v + \nu uv\ )dxdy = 0 \\ K\nabla u.\overrightarrow{n} = 0 \text{ pour } x = 0 \\ K\nabla u.\overrightarrow{n} = 0 \text{ pour } y = 0 \end{cases} \tag{4}$$

On discrétise la formulation variationnelle afin de pouvoir l'étudier numériquement.
Pour le temps, on utilisera un schéma d'Euler semi-implicite.
Pour l'espace, on utilisera la méthode des éléments finis de degré 1 sur des triangles.
On écrit les w_h sur chaque triangle car si $x \in T_k$ (le $k^{\text{ème}}$ triangle) alors :

$$w_h^k(x,y) = w_h^k(q_0^k)\lambda_0^k(x,y) + w_h^k(q_1^k)\lambda_1^k(x,y) + w_h^k(q_2^k)\lambda_2^k(x,y)$$

q_i^k étant les sommets du triangle T_k.
$\lambda_i^k(x,y)$ sont les coordonnées barycentrique de T_k.
Comme la restriction au triangle T_k est P^1 et que les $w_h^k(q_i^k) \in \mathbb{R}$, on a que les $\lambda_i^k(x,y) \in P^1$.
Donc on peut les écrire sous la forme :

$$\lambda_i^k(x,y) = a_i + b_ix + c_iy \ \text{ pour i = 0, 1, 2}$$

On a enfin que $\lambda_i^k(q_j^k) = \delta_{ij}$. (cela traduit la propriété que les coordonnées barycentrique d'un sommet valent 1 en ce sommet et 0 ailleurs. Avec ces deux propriétés de λ_i^k, on a que :
$w_h^k(q_i^k) = w_h^k(q_0^k)\lambda_0^k(q_i^k) + w_h^k(q_1^k)\lambda_1^k(q_i^k) + w_h^k(q_2^k)\lambda_2^k(q_i^k) = \delta_{ij}$ avec j=0,1,2

On se ramène donc a la même situation qu en 2 dimensions. Avec w_h la base canonique de $V_{0h} = \{v_h \in \mathcal{L}^2(\Omega),\ \frac{\partial v_h}{\partial x} \in \mathcal{L}^2(\Omega),\ \frac{\partial v_h}{\partial y} \in \mathcal{L}^2(\Omega), \frac{\partial v}{\partial \overrightarrow{\nu}} \in \mathcal{L}^2(\Gamma_1)\bigcup\mathcal{L}^2(\Gamma_4),\ v_h = 0\ pour\ t = 0\}$.
Pour déterminer les a_i, b_i ,c_i, on se sert de la propriété suivante : les λ_i^k valent 1 au sommet q_i^k et 0 au sommet q_j^k pour $j \neq i$. On obtient ainsi le système suivant :

$$\begin{cases} \lambda_0^k(q_0^k) = a_0 + b_0x_0^k + c_0y_0^k = 1 \\ \lambda_0^k(q_1^k) = a_0 + b_0x_1^k + c_0y_1^k = 0 \\ \lambda_0^k(q_2^k) = a_0 + b_0x_2^k + c_0y_2^k = 0 \end{cases}$$

On applique une méthode de Kramer pour résoudre ce système, on obtient :

$$\lambda_0^k = \frac{\overrightarrow{q_1^k q^k} \wedge \overrightarrow{q_2^k q^k}}{\overrightarrow{q_1^k q_0^k} \wedge \overrightarrow{q_2^k q_0^k}} \tag{5}$$

avec $q^k = \begin{pmatrix} x \\ y \end{pmatrix}$.
On a de même :

$$\lambda_1^k = \frac{\overrightarrow{q_2^k q^k} \wedge \overrightarrow{q_0^k q^k}}{\overrightarrow{q_2^k q_1^k} \wedge \overrightarrow{q_0^k q_1^k}}$$

$$\lambda_2^k = \frac{\overrightarrow{q_0^k q^k} \wedge \overrightarrow{q_1^k q^k}}{\overrightarrow{q_0^k q_2^k} \wedge \overrightarrow{q_1^k q_2^k}}$$

Ou écrit autrement :

$$\begin{aligned}
\lambda_0^k &= \frac{(x_1^k y_2^k - x_2^k y_1^k) - (y_2^k - y_1^k)x + (x_1^k - x_2^k)y}{(x_1^k - x_0^k)(y_2^k - y_0^k) - (y_1^k - y_0^k)(x_2^k - x_0^k)} \\
\lambda_1^k &= \frac{(x_2^k y_0^k - x_0^k y_2^k) - (y_0^k - y_2^k)x + (x_2^k - x_0^k)y}{(x_2^k - x_1^k)(y_0^k - y_1^k) - (y_2^k - y_1^k)(x_0^k - x_1^k)} \\
\lambda_2^k &= \frac{(x_0^k y_1^k - x_1^k y_0^k) - (y_1^k - y_0^k)x + (x_0^k - x_1^k)y}{(x_0^k - x_2^k)(y_1^k - y_2^k) - (y_0^k - y_2^k)(x_1^k - x_2^k)}
\end{aligned} \tag{6}$$

Et la propriété des coordonnées barycentrique : $\lambda_0^k + \lambda_1^k + \lambda_2^k = 1$

On aura ainsi les coordonnées barycentriques de chaque point dans chaque triangles du maillage facilement, en fonction des coordonnées des sommets du triangles.
Maintenant interessons-nous à la formule à discrétiser.

Donc on obtient la formulation suivante après avoir discrétisé avec le schéma d'Euler semi-implicite puis passé sous forme variationnelle :
$\forall v \in V_{0h}$

$$\int_\Omega (\frac{u^{m+1}v - u^m v}{\delta t} + K\nabla u^{m+1}\nabla v + \mu \nabla u^m v + \nu u^{m+1} v) dxdy = 0$$

puis on remplace les v et les u^{m+1} et u^m par leur expression dans la base w^i de V_{0h} :
$u_h = \sum_{i/q^j \notin \Gamma} u_h(q^{(j)}) w^{(j)}$ et $v_h = \sum_{i/q^i \notin \Gamma} w^{(i)}$
On obtient ainsi :

$$\begin{aligned}
\int_\Omega (&\frac{vu^{m+1} - vu^m}{\delta t} + \\
&\sum_{i/q^i \notin \partial\Omega} [(K\nabla(u_h^{m+1}(q^{(j)})w^{(j)})\nabla(w^{(i)}) + \\
&\mu\nabla(u_h^m(q^{(j)})w^{(j)})w^{(i)} + \\
&\nu u_h^{m+1}(q^{(j)})w^{(j)}w^{(i)}])dxdy = 0
\end{aligned}$$

On a que $U^m = \sum_{j/q^j \notin \partial\Omega} u_h^m(q^{(j)})$. De même pour U^{m+1}. En multipliant tout pour $\delta t \neq 0$, on a :

$$\sum_{k/T_k \notin \partial\Omega} \int_{T_k} ((w^j, w^i)U^{m+1} - (w^j, w^i)U^m + \delta t(K\nabla w^j, \nabla w^i)U^{m+1} + \delta t\mu(\nabla w^j, w^i)U^m + \delta t\nu(w^j, w^i)U^{m+1})dxdy = 0$$

On factorise ensuite par U^m et U^{m+1}, et met U^m d'un côté du signe égale et U^{m+1}.
Finalement on a :

$$((w^j, w^i) + \delta t[(K\nabla w^j, \nabla w^i) + \nu(w^j, w^i)])U^{m+1} = ((w^j, w^i) - \delta t[\mu(\nabla w^j, w^i)])U^m \tag{7}$$

Sur chaque T_k, on en prend l intégrale.

Passons maintenant a la méthode de résolution numérique du problème. Pour $m = 0$, on connait U^0 d'après les conditions initiales du problème.Donc on va résoudre tout d'abord :

$$AU^1 = BU^0 \tag{8}$$

Avec $A_{i,j} = \sum_{i,q_i \in \Omega} \int_{T_k} ((w^j, w^i) + \delta t[(K\nabla w^j, \nabla w^i) + \nu(w^j, w^i)])dxdy$

et $B_{i,j} = \sum_{i,q_i \in \Omega} \int_{T_k} ((w^j, w^i) - \delta t[\mu(\nabla w^j, w^i)])dxdy$

Une fois ce systéme résolut en rajoutant pour les points du maillage sur le bord les conditions adéquates.
On résoud pour $m = 1$, car on connaîtra alors U^1. On peut calculer ainsi tout de manière récurcive tous les U^m. On laisse ensuite tourner notre programme sfemGC2.cpp. Qui résoudra tout cela.

2 Etude par différences finies

On va faire la dérivée par rapport à K et à q par différence fini. Tout d abord au lieu de recalculer une nouvelle fois le vecteur solution. Nous le stockons. Ensuite on va refaire le calcule pour une autre valeur de K ou de q, selon la dérivée voulue. Enfin pour terminer, on regarde quel était le K ou le q le plus grand en fonction de K et q initiale. Si la deuxième variable rentrée est plus grande, on prendra la valeur négative du vecteur solution. En résumer :

Pour la dérivée par rapport a K

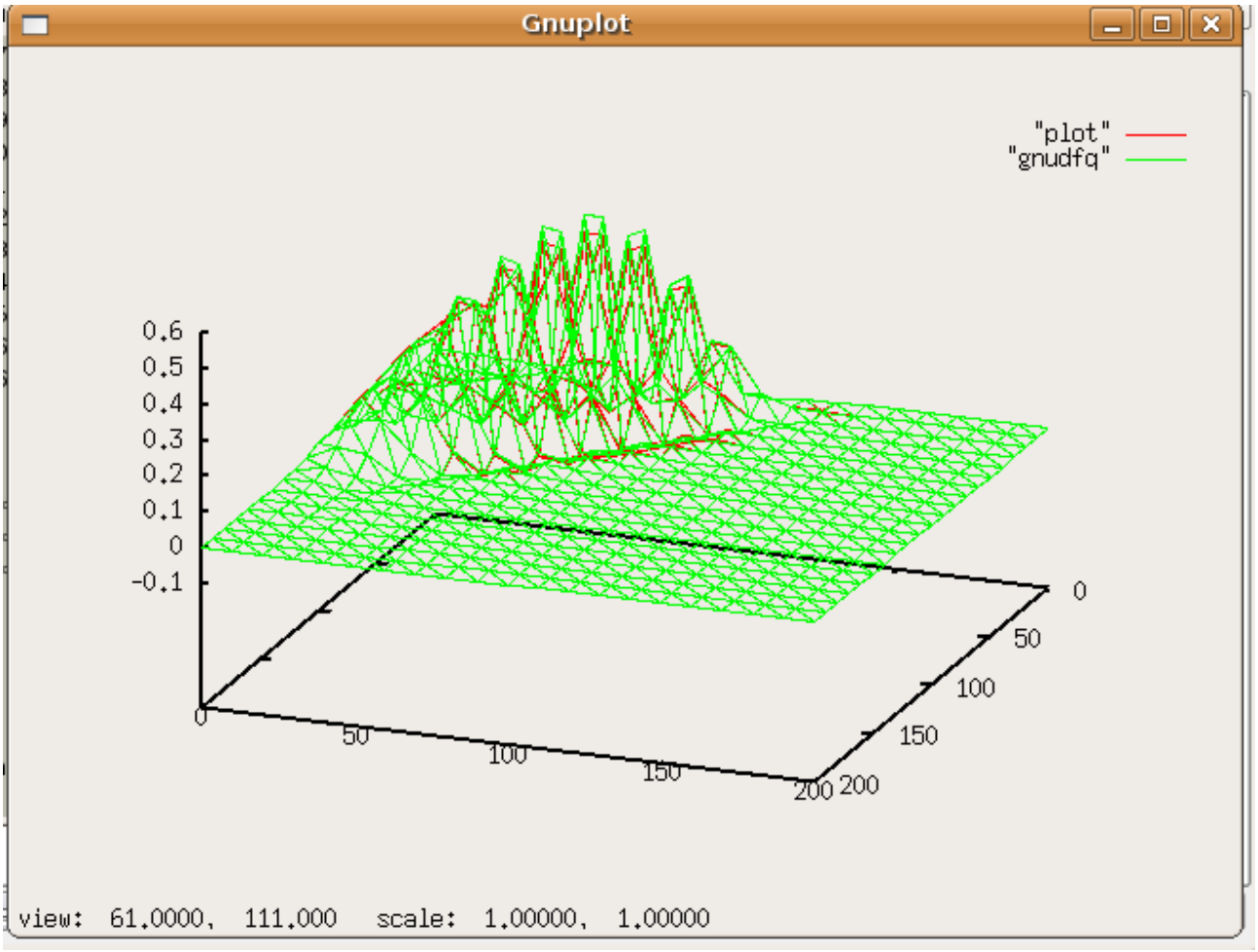
$$\text{On fait } u^M = \frac{u_K - u_{K'}}{K - K'} \text{ si } K > K'$$
$$\text{sinon } u^M = -\frac{u_K - u_{K'}}{K - K'}$$

Pour la dérivée par rapport a q

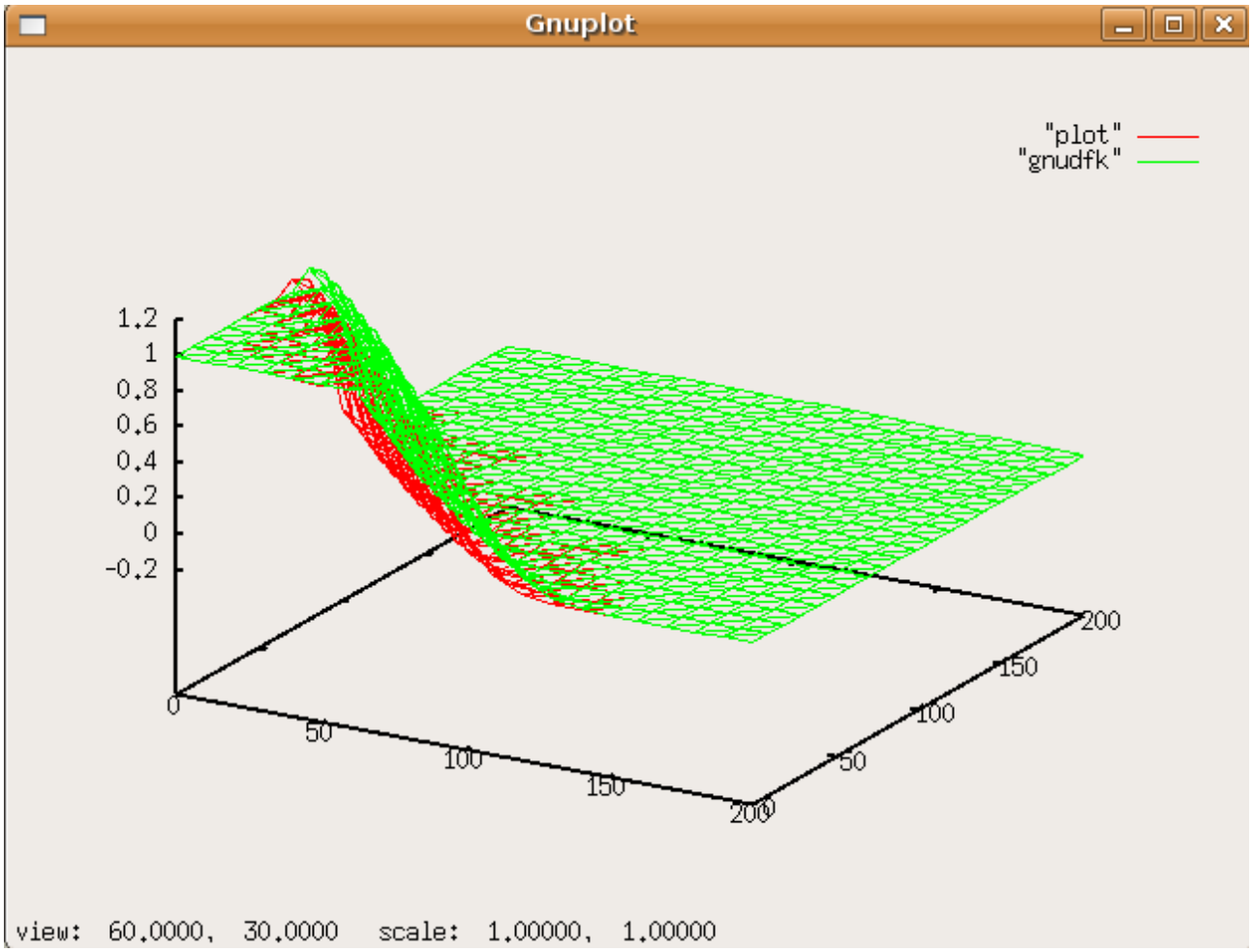
$$\text{On fait } u^M = \frac{u_q - u_{q'}}{q - q'} \text{ si } q > q'$$
$$\text{sinon } u^M = -\frac{u_q - u_{q'}}{q - q'}$$

3 Etude par différentiation automatique

On a changé les *double* en *ddouble*, dans R2.hpp. On a aussi pour cela rajouter le *#include "ddouble.h"*. Puis on a retiré le *typedef R* dans DA.cpp. Enfin on a remplacé *#include "GC.hpp"* par *#include "GCDA.hpp"* dans DA.cpp. Pour le reste, cela a été du débugage, par exemple retirer les opérateurs qui posent des problémes de choix au compilateur.
Enfin pour le faire fonctionner, on initialise la $2^{\text{ième}}$ variable de chaque paramètre a 0.. Et l on met 1 a la variable K si l'on veut la dérivée par rapport à K.
La différentiation automatique est une manière numérique de calculer les dérivées, et justement elle est en cela beaucoup plus longue que la méthode des diffrences finies, car demande plus de calcul. Mais cela donne une bonne estimation de la dérivée que l'on doit obtenir. Cette méthode peut donc valider ou non le calcul d'une dérivée.



comparaison entre différence fini en vert et diffrentiation automatique en rouge pour q



comparaison entre différence fini en vert et diffrentiation automatique en rouge pour K

4 Open GL

Pour la programmation des isolignes, nous avons créé plusieurs sous programme. Tout d'abord *isolect* qui charge le fichier "isoval.dat" contenant la liste des isovaleurs à tracer. Il charge tout cela dans un vecteur auquel il rajoute en première et dernière ligne une valeur très petite , et une valeur trés grande respectivement.Cela permet d'éviter les surprises d'isolignes non coloriées.
A la suite de cela on a modifié le programme *SetColor*, pour qu'il puisse attribuer, une couleur a chaque isoligne.
Ensuite on a rencontré le problème de savoir où une isoligne coupait chacun des triangles. On a donc créé le programme *Labeliseur*. Enfin on a créé le programme *Calpoint* qui calcule le point d'intersection entre une isoligne et le coté d'un triangle. Pour se faire, on se sert d une méthode de proportionnalité entre l'isovaleur et la différence des coordonnées z de sommets de ce segment. Cette méthode a malheureusement ses limites dans le cas de surface courbe, si le maillage est trop grossier. Voila une visualisation de ce que l'openGL fait :

Visualisation de la solution de l'équation de black and scholes

5 Programmes et résultats :

Les codes rajoutés à $sfemGC2.cpp$ risquant d'encombrer le rapport. Nous demanderons au lecteur d'aller le voir si il veut plus d'information.
Dans les modifications faites, on a :
_ changé la condition initiale en temps
_ fait un chargement des données initiales grace au fichier "projet.txt"
_ une fonctionne qui calcul le barycentre, et peut le stocker dans "g.data"
_ calculer les constantes K μ à partir des coordonnées du barycentre de chaque triangle
_ prend en compte des conditions aux limites spécifiques du probléme (nulles sur Γ_1 et Γ_4)
_ modifié $matlap2d$ pour prendre en compte le terme en $(\nabla w^j, w^i)$ de B
_ rajouter la sortie du fichier "uh.txt" pour FreeFem++
_ creation de fichier d'execution pour FreeFemm++, et pour gnuplot
_ fait un menu intŕactif dans $sfemGC2$ qui permet d'obtenir tous les résultats du projet

Ensuite on a aussi écrit un programme en java, qui permet de rentrer les données. Voir pour le code $entreur.java$. Il possède même un bouton qui efface la fenêtre et quitte le programme. Les données sont stockées dans "projet.txt".

6 Remerciements

Je tiens à remercier tout particulierement, pour leur aide très précieuse, Messieur Alain PERRONNET et Hanen AMOR, sans qui ce projet n'aurait pu avancer.