README

Ci-joint quelques lignes sur le fonctionnement de notre code.
Après avoir compilé, il suffit d'exécuter ./pde_solver .
Pour ne pas avoir à rentrer dans le code, le programme demande les paramètres de l'option a pricer a l'utilisateur lors de l'exécution. Ces entrées ne sont pas robustes à une typo de l'utilisateur, il faut les rentrer dans le format attendu.

L'utilisateur doit aussi rentrer le nombre de points de discrétisation du spot et du temps. Merci d'entrer un nombre impair pour ndx (pas d'interpolation linéaire dans le cas pair). Libre à l'utilisateur d'entrer ndx et ndt relativement grand (une 15aine de seconde d'exécution pour ndx = 1001 ; ndt = 1000).

Il est possible d'utiliser des paramètres non constants. L'utilisateur est libre d'implémenter un modèle heston pour la volatilité. Pour le taux d'intérêt, il est possible de spécifier une courbe linéaire qui croit (ou décroit) à un certain taux à chaque dt.

Le programme affiche ensuite les valeurs des boundary conditions, le prix et enfin les greeks.

Pour ce qui est de la résolution du système tout se passe dans solver.cpp, où nous utilisons l'algorithme de matrice tridiagonale. Nous avons préféré la solution d'implémenter une classe matrix plutôt que d'utiliser la librairie eigen.