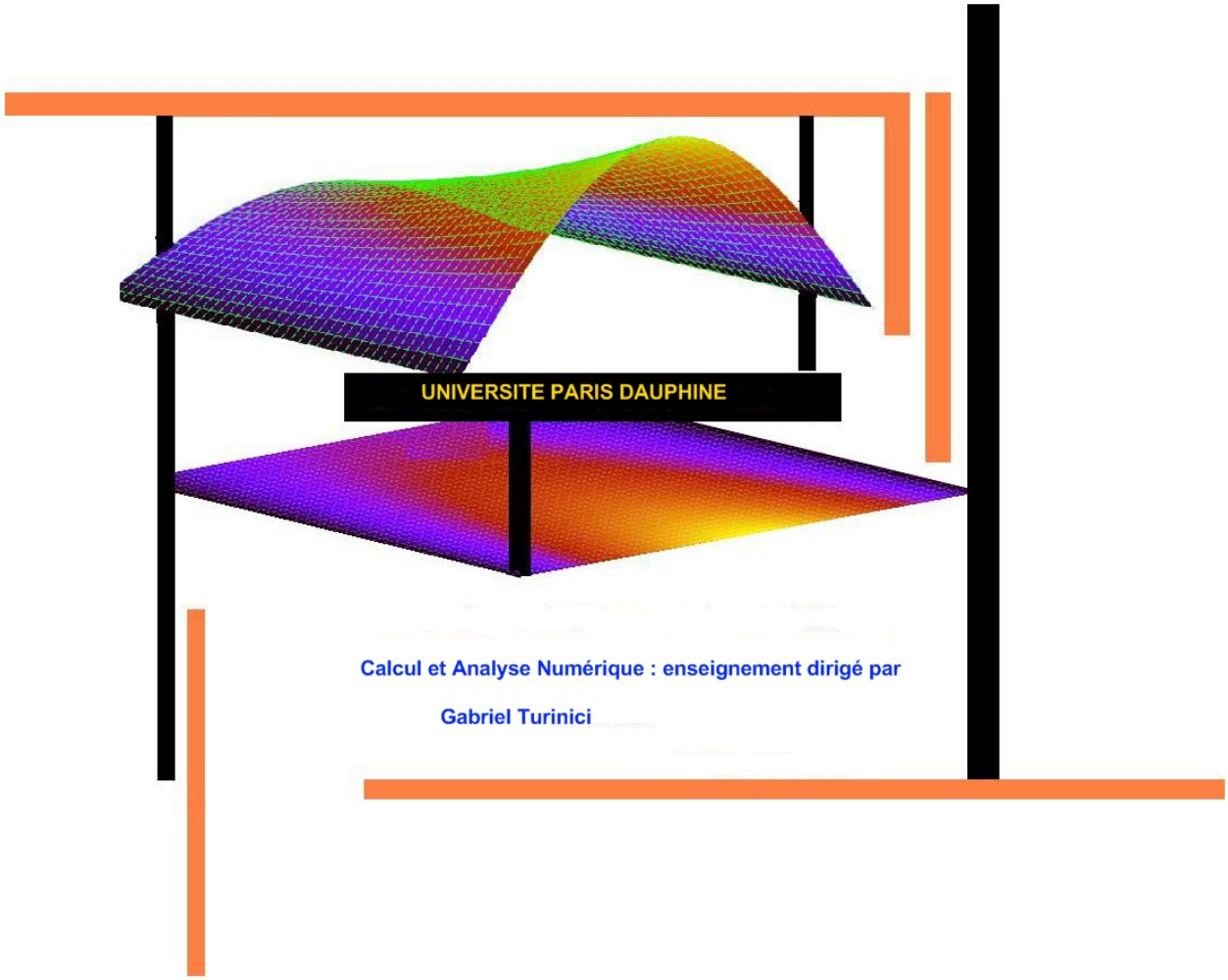
Calcul et Analyse Numérique
Projet : Choix de stratégie avec gestion de risques *

Rhita Cherkaoui
Marine Ducourant
Pouya Eshaghi
Florian Gauch
Florent Heitzmann
Charlotte Laurent

10 juin 2009



*Toute la bibliothèque des classes utilisées ainsi que l'ensemble des codes sources de ce projet se trouvent sur le site http://starbrood.free.fr/projetfinance

Table des matières

1 Analyse de données financières

L'objet de cette partie est d'analyser les cours de trois données financières sur une période d'environ 10 ans. Nous nous intéresserons tout d'abord à l'évolution du taux de change EUR/USD depuis la création de l'euro (données collectées sur le site de la Banque Centrale Européenne : http ://www.ecb.int), puis à l'évolution de l'indice CAC 40 depuis le deuxième semestre 2000, et enfin au cours de l'action Teleperformance, côtée sur le marché parisien hors CAC 40, également depuis le deuxième semestre 2000 (les données concernant le CAC et Teleperformance ont été trouvées sur le site de *ABC Bourse* : http ://www.abcbourse.com). Pour simplifier, on considèrera pour la simple étude de la variance et de la moyenne, les indices comme des actifs financiers. Les trois séries chronologiques sont analysées ci-dessous.

1.1 Analyse des séries chronologiques

1.1.1 Moyenne et Variance des incréments logarithmiques

Pour un actif ou un indice donné on note N le nombre de jours de la période d'étude, donc le nombre de données dont on dispose. Le cours de l'actif au jour i est noté $S_i,\ 1 \leqslant i \leqslant N$, et l'on va s'intéresser aux incréments logarithmiques $\ln(S_{i+1}) - \ln(S_i) = \ln(S_{i+1}/S_i)$. On modélisera nos actifs financiers (en temps continu) chacun comme un processus d'Itô s'écrivant :

$$S_t = S_0 + \int_0^t \mu S_s \, ds + \int_0^t \sigma S_s \, dW_s \tag{1}$$

Où W_t est un mouvement Brownien standard. Il vérifie donc l'équation différentielle stochastique :

$$dS_t = S_t(\mu dt + \sigma dW_t) \tag{2}$$

Dont on sait qu'elle admet une solution unique donnée par :

$$S_t = S_0 \exp\left\{(\mu - \frac{\sigma^2}{2})t + \sigma W_t\right\} \tag{3}$$

Pour introduire un schéma discret, nous choisirons un incrément de temps h. Pour i fixé, on s'intéresse donc à $\ln(S_{i+h}) - \ln(S_i)$. En utilisant la formule (3), nous obtenons pour l'espérance :

$$\begin{aligned}
\mathbb{E}[\ln(\frac{S_{i+h}}{S_i})] &= \mathbb{E}[\ln(S_{i+h}) - \ln(S_i)] \\
&= \mathbb{E}[(\mu - \frac{\sigma^2}{2})h + \sigma(W_{i+h} - W_i)] \\
&= (\mu - \frac{\sigma^2}{2})h
\end{aligned}$$

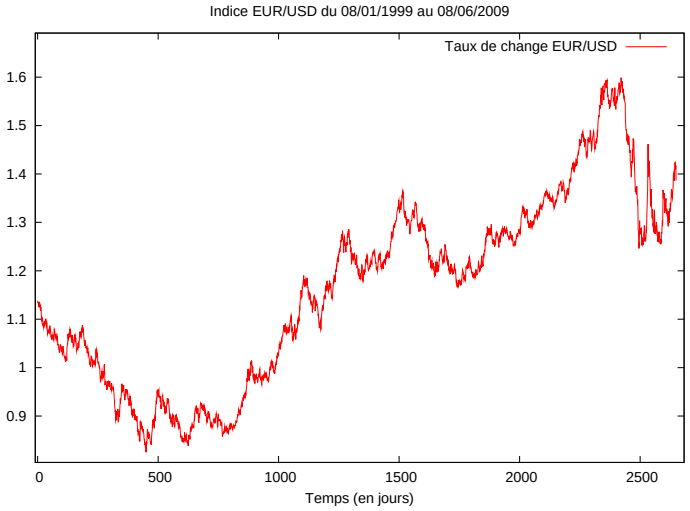
En utilisant le fait que $(W_{i+h} - W_i)$ suit une loi $\mathcal{N}(0, h)$. Pour la variance nous obtenons de même, grâce à (3) :

$$\mathbb{V}[\ln(\frac{S_{i+h}}{S_i})] = \mathbb{V}[(\mu - \frac{\sigma^2}{2})h + \sigma(W_{i+h} - W_i)] = h\sigma^2$$

Nous allons étudier la moyenne et la variance empiriques des incréments logarithmiques de nos trois séries chronologiques, avec dans notre cas $h = 1$, les pas de temps seront donc $ih = i,\ 1 \leqslant i \leqslant N$.

1.1.2 La parité Euro-Dollar

Nous avons le cours de la parité EUR/USD du 29/01/1999 au 08/06/2009, ce qui correspond à 2650 jours de cotation. La série chronologique obtenue est présentée ci-dessous :



Nous calculons les moyenne et variance empirique de la série $L(S) = (\ln(\frac{S_{i+1}}{S_i})), 1 \leqslant i \leqslant N-1$, soit :

$$\tilde{\mathbb{E}}[L(S)] = \frac{\sum_{i=1}^{N-1} \ln(\frac{S_{i+1}}{S_i})}{N-1}$$
$$\tilde{\mathbb{V}}[L(S)] = \frac{\sum_{i=1}^{N-1} (\ln(\frac{S_{i+1}}{S_i}) - \tilde{\mathbb{E}}[L(S)])^2}{N-1}$$

Ces statistiques se calculent très simplement, et l'on renvoie à la section suivante pour le calcul numérique. Notons que l'estimateur de la variance empirique est celui de la méthode des moments. Il est biaisé, on pourrait le débiaiser en le multipliant par $(N-1)/(N-2)$, mais nous voulons surtout ici pouvoir calculer μ grâce à cet estimateur, et non l'étudier parfaitement. Le nombre de jours N est suffisamment grand pour que cet estimateur convienne et nous l'utiliserons toujours par la suite. Les valeurs de μ et σ se déduisent de ces estimateurs de la manière suivante :

$$\begin{cases} (\mu - \frac{\sigma^2}{2}) & = \tilde{\mathbb{E}}[L(S)] \\ \sigma^2 & = \tilde{\mathbb{V}}[L(S)] \end{cases}$$

Ce qui donne :

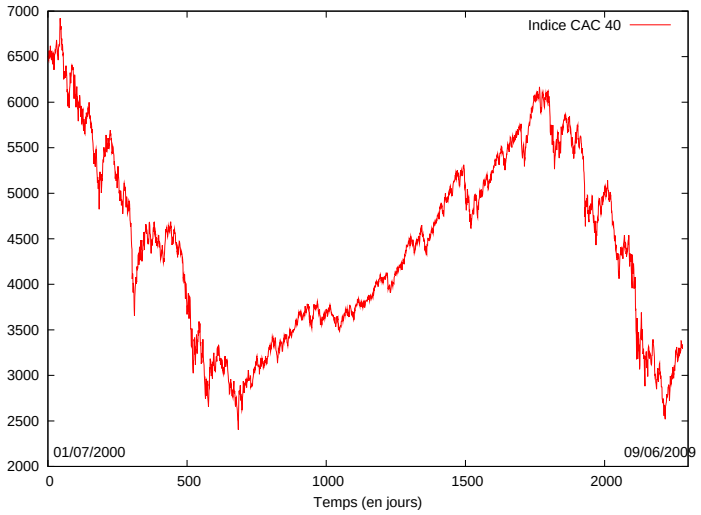
$$\begin{cases} \mu & = \tilde{\mathbb{E}}[L(S)] + \frac{\tilde{\mathbb{V}}[L(S)]}{2} \\ \sigma^2 & = \tilde{\mathbb{V}}[L(S)] \end{cases} \quad (4)$$

Nous obtenons pour le taux EUR/USD les résultats suivants.

Moyenne des incréments logarithmiques	Variance des incréments logarithmiques	μ	σ^2
$7{,}44.10^{-5}$	$4{,}50.10^{-5}$	$9{,}69.10^{-5}$	$4{,}50.10^{-5}$

1.1.3 L'indice CAC 40

Nous disposons pour l'indice CAC 40 d'un échantillon de 2281 valeurs, correspondant aux jours de cotations entre le 03/07/2000 et le 08/06/2009. La série chronologique est représentée ci-dessous.

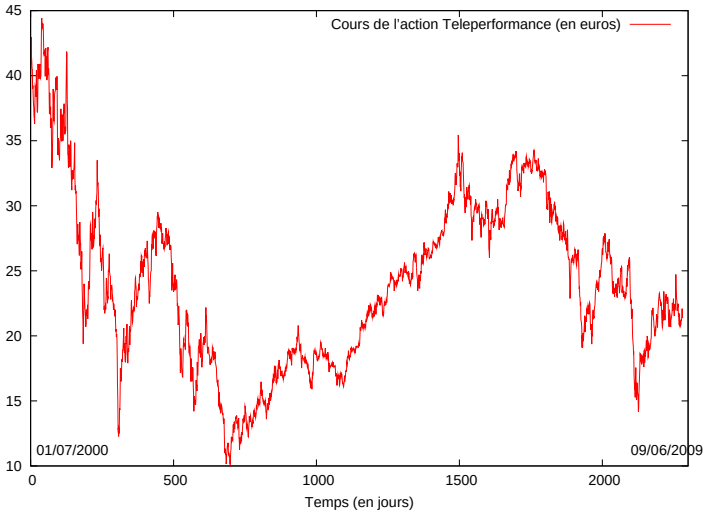


Comme pour l'Euro-Dollar, on utilise les relations (4) pour calculer μ et σ à partir des résultats obtenus. Ces derniers sont consignés dans le tableau suivant.

Moyenne des incréments logarithmiques	Variance des incréments logarithmiques	μ	σ^2
-2,99.10^{-4}	2,52.10^{-4}	-1,73.05^{-4}	2,52.10^{-4}

1.1.4 Le cours de l'action Teleperformance

Nous disposons d'un echantillon de l'action Teleperformance (code RCF) semblable à celui du CAC 40 : 2281 valeurs correspondant aux jours de cotation entre le 03/07/2000 et le 08/06/2009. La série chronologique est représentée ci-dessous.



Grâce à (4), on calcule μ et σ^2 pour ce titre.

Moyenne des incréments logarithmiques	Variance des incréments logarithmiques	μ	σ^2
$2{,}65.10^{-4}$	$6{,}85.10^{-4}$	$7{,}75.10^{-4}$	$6{,}84.10^{-4}$

1.2 Programmation

1.2.1 Principes et choix de programmation

Nous avons choisi de programmer nos méthodes en C++. L'inconvénient par rapport à Matlab repose sur le fait que la visualisation des résultats se fait par l'intermédiaire de fichiers texte de coordonnées de points `.dat`. Pour le rendu graphique, nous utilisons principalement Gnuplot. Pour la compilation C++, nous compilons sous cygwin par la commande g++. Après compilation, nos programmes sont exécutables par un double clique sous la condition d'avoir cygwin.dll présent sur le disque et chargé dans la registrie windows ou la console de commande. Il est également possible de créer une interface java (appletscript) pour éxécuter le programme et afficher les résultats sous forme de texte.
Résumons le but du programme de cette partie. À partir d'un fichier texte qui contient une série chronologique sous la forme `[date][cotation]`, on va calculer les incréments logarithmiques $\ln(\frac{S_{i+1}}{S_i}),\ i \in \{1, .., N-1\}$ (on renvoie à la section suivante pour le formalisme des calculs). i représente la i-ème journée de cotation en partant du jour $t_0 = 0$. Pour garder la correspondance date réelle $\leftrightarrow$ journée de cotations, on stocke les dates dans un vecteur dates indexé de la même façon que les cotations. On va ensuite calculer les moyennes et variances empiriques (estimateurs des moments) des incréments logarithmiques précédents. On calculera μ et σ sur l'ensemble de la période puis par périodes mobiles de 3, 6, 9 ou 12 mois.

1.2.2 Choix de la structure

Le problème entier est traité par la classe `Outils` que nous allons présenter. Son constructeur prend uniquement en paramètre le nombre de jours et le nom du fichier à considérer. Les noms des fichiers créés par la suite dépendront de cette chaîne de caractère.

```
class Outils{
  public :
    int N; //nbre de points attention !
    R h; // h=dT=1/(N-1) (et donc N-1 intervalles dans discrétisation)
    KN<R> cotations;  //vecteur de cotations
    KN<string> dates; //vecteur des dates correspondantes
    char* name;  //nom de l'instanciation concaténée à tous les noms de fichiers

    Outils(char* fname2,const int N2);  // constructeur, N2 : nombre de points

    void readFile(char* fname); // permet de lire le fichier de nom fname
    R2 getMeanAndVar(KN<R> vect); //renvoie (mean empirique, var empirique)
    KN<R2> meanVarMob(int mois);  //moyenne et var mobile. (21 jours par mois)
    void Vectorprinter(KN<R> vecteur,const char* fnamev); //pour printer vecteurs
    void printMeanAndVar(KN<R2> vecteur, int i); //pour manier vecteurs de mean
                                                 //mobiles et var mobiles
     //méthode pour trouver le nombre de jours pour le calcul des moyennes mobiles
     int getMobParam(KN<R2> vecteur); //en fonction du nombre de zéros
};
```

On observe que les moyennes et variances empiriques des incréments logarithmiques ne sont pas stockées dans des vecteurs. Ils sont par contre renvoyés par la méthode `KN<R> meanVarMob(int mois)` puis directement imprimés ou stockés dans le `main` pour d'éventuelles manipulations. Nous avons donc choisi d'importer la bibliothèque `RNM` fournie par le laboratoire Jacques-Louis Lions, rattaché à l'université Pierre et Marie Curie. Son code sera placé en annexe. Notons que cette bibliothèque permet l'implémentation de tableaux de tous types grâce à l'utilisation de templates. Dans la même optique, en important la classe `R2` par le fichier d'en-tête `R2.hpp`, nous pouvons implémenter des tableaux de `R2` par la déclaration `KN<R2>` qui permet dans un vecteur de stocker les moyennes et variances par exemple. Présentons tout de suite le code de cette partie :

1.2.3 Code C++ Partie I

```
Outils::Outils(char* fname2,const int N2):N(N2){
  readFile(fname2);h=1/((R)(N2-1));
}

void Outils::readFile(char* fname){
  name=fname;
  KN<string> datesaux(N);KN<R> cotationsaux(N);double cours=0;
  fstream readera(fname, ios_base::in);
  int compteur=0;
  if(readera.is_open()){  //while(true){ si pbleme de buffer
     for(int i=0;i<N;i++){
        readera >> datesaux[i] >> cotationsaux[i];
        if(readera.eof()){ break;}//developper ou insérer un assert
    }
  }
  //CAS SPECIAL car les dates sont décroissantes pour eurus.dat
  if(fname=="eurus.dat"){
  KN<string> datesinverses(N);
  KN<R> cotationsinverses(N);
  for(int i=0;i<N;i++){
    datesinverses[i]=datesaux[N-1-i];
    cotationsinverses[i]=cotationsaux[N-1-i];
  }
    dates=datesinverses;
    cotations=cotationsinverses;
  }else{
     dates=datesaux;
     cotations=cotationsaux;
  }
}

void Outils::Vectorprinter(KN<R> vecteur, const char* fnamev){
 ofstream plot(fnamev);
 for(int it=0;it<=N;it++){
   plot << it << " " << vecteur[it]<< endl;
 }
}

int Outils::getMobParam(KN<R2> vecteur){
    int compteur=0;R nul=0.;
    for(int i=0;i<N;i++){
      if(!vecteur[i][0]==nul){break;}
      compteur++;
    }
    return compteur;
}

void Outils::printMeanAndVar(KN<R2> vecteur,int i){
  // getMobParam renvoie le nombre de jours utilisés pour moyenne mobile
  int param=getMobParam(vecteur)/21;
  if(i==0){
    KN<R> m(N);
    for(int i=0;i<N;i++){
      m(i)=vecteur[i][0];
    }
    //cette partie permet de générer un nom de fichier
```

```
    //en fonction de param : nbre de jour des périodes glissantes
    string moyfname="moymob";
    stringstream bufferm;
    bufferm<<moyfname<<param<<"-"<<name;
    const char* fm = bufferm.str().c_str();
    Vectorprinter(m,fm);

  }else if(i==1){
     KN<R> v(N);
     for(int i=0;i<N;i++){
       v(i)=vecteur[i][1];}
     string varfname="varmob";
     stringstream bufferv;
     bufferv<<varfname<<param<<"-"<<name;
     const char* fv = bufferv.str().c_str();
     Vectorprinter(v,fv);
  }else if(i==2){ //pour imprimer dates
     string datesfname="table";
     stringstream bufferv;
     bufferv<<datesfname<<"-"<<name;
     const char* days = bufferv.str().c_str();
     ofstream plot(days);
   for(int it=0;it<N;it++){
   plot << it << " " << dates[it]<<" "<<cotations[it]<<endl; }
  }
}

R2 Outils::getMeanAndVar(KN<R> vect){
  R mean=0.;R var=0.; //var est en réalité un écart type
  for(int i=1;i<N;i++){
    mean+=log(vect[i]/vect[i-1]); }
  mean=mean/N;
  for(int k=1;k<N;k++){
    var+=pow(log((vect[k]/vect[k-1])-mean),2);}
  var=var/N;
  R2 result(mean,var);
  return result;
}

KN<R2> Outils::meanVarMob(int mois){
  KN<R2> meanvarmob(N);
  R meant;R vart;R2 temp(0.,0.);
  for(int k=0;k<21*mois;k++){
    meanvarmob[k]=temp;}
  for(int i=21*mois;i<N;i++){
    meant=0.;vart=0.;
    for(int j=i-21*mois+1;j<=i;j++){
      meant+=log(cotations[j]/cotations[j-1]);  }
    meant=meant/(21*mois-1);
    for(int j=i-21*mois+1;j<=i;j++){
      vart+=pow(log((cotations[j]/cotations[j-1])-meant),2); }
    vart=vart/(21*mois-1);
    R2 meanvaraux(meant,vart);
    meanvarmob[i]=meanvaraux;
  }
  return meanvarmob;
}
```

Arrêtons nous ici pour commenter ces méthodes. Le constructeur initialise `N` et la chaîne de caractère `fname` et fait appel à la méthode `readFile` qui lit le fichier de nom "fname" présent dans le même répertoire que le programme. La série chronologique est stockée dans `cotations` qui est un vecteur de type `KN<R>` de taille N. Le vecteur de `string` `KN<string> dates` stocke les dates correspondant à chaque jour de cotation sous forme de chaîne de caractère. On peut ainsi retrouver quelle était la date réelle du cours du `i`-ème jour de cotation enregistré dans `cotations[i]`. Le cas de l'Euro-Dollar est traité spécifiquement car les cotations sont classées par ordre décroissant (en temps). Une méthode aurait pu être programmée dans notre code pour vérifier si les cotations sont classées par ordre croissant, ou encore une grammaire pour traiter les dates et leur affecter des opérateurs. Mais restons ici concentrés plus sur les résultats que sur la syntaxe pour cette partie du problème.

Commentons le code pour les moyennes mobiles et les variances mobiles des incréments logarithmiques. La méthode `KN<R2> Outils::meanVarMob(int mois)` renvoie un vecteur de R2 `meanvarmob` tel que `meanvarmob[i][0]` désigne la moyenne mobile des incréments logarithmiques pour le i-ème jour de cotations. `meanvarmob[i][0]` renvoie quant à lui la variance mobile du i-ème jour de cotation. On a dans ce cas par rapport au paramètre `int mois` : $\forall i \in \{0, 1, ..., 21 * mois - 1\}$, `meanvarmob[i][0]=meanvarmob[i][1]=0`. Les périodes glissantes sont décentrées à gauche en temps en chaque points. La méthode `getMobParam(KN<R2>VecteurdeGain)` permet ainsi par exemple de retourner le paramètre (en nombre de jours) des périodes glissantes utilisées pour calculer le vecteur `VecteurdeGain`. Grâce à cette méthode, la classe `Outils` permet d'imprimer des noms de fichiers automatiquement correspondants aux paramètres utilisés. Pour la conversion `jour/mois`, nous avons considéré une moyenne de 21 jours de cotations par mois. C'est ce qu'illustre la méthode `void doAll(Outils Objet)` qui permet l'impression des fichiers qui nous intéressent :

```
void doAll(Outils objet){
  R2 meanvar=objet.getMeanAndVar(objet.cotations);
  cout<<"mean and variance des increments "`;
  cout<<logarithmes ln(Si+1/Si) : "<<endl<<meanvar<<endl;
  KN<R2> meanvarmob3mois=objet.meanVarMob(3);
  KN<R2> meanvarmob6mois=objet.meanVarMob(6);
  KN<R2> meanvarmob9mois=objet.meanVarMob(9);
  KN<R2> meanvarmob12mois=objet.meanVarMob(12);
  const char* name;
  string moyfname="tendances";
  {stringstream bufferm;
  bufferm<<moyfname<<"-"<<objet.name;
  name= bufferm.str().c_str();}
  ofstream plot(name);
  R mu3;R mu6; R mu9; R mu12;
  for(int it=0;it<objet.N;it++){
    mu3=((meanvarmob3mois[it][0]+meanvarmob3mois[it][1]/2.));
    mu6=((meanvarmob6mois[it][0]+meanvarmob6mois[it][1]/2.));
    mu9=((meanvarmob9mois[it][0]+meanvarmob9mois[it][1]/2.));
    mu12=((meanvarmob12mois[it][0]+meanvarmob12mois[it][1]/2.));
    plot << it <<" "<< mu3 << " "<<mu6<<" "<<mu9<<" "<<mu12<<endl;  }
  const char* namevol;
  moyfname="volatilites";
  {stringstream bufferv;
  bufferv<<moyfname<<"-"<<objet.name;
  namevol= bufferv.str().c_str();}
  ofstream plot2(namevol);
  R sigma3;R sigma6;R sigma9;R sigma12;
  for(int it=0;it<objet.N;it++){
    sigma3=sqrt(meanvarmob3mois[it][1]); //62 jours
    sigma6=sqrt(meanvarmob6mois[it][1]); //125 jours
    sigma9=sqrt(meanvarmob9mois[it][1]); //188j
    sigma12=sqrt(meanvarmob12mois[it][1]); //251
    plot2<<it<<" "<<sigma3<<" "<<sigma6<<" "<<sigma9<<" "<<sigma12<<endl;}
  objet.printMeanAndVar(meanvarmob3mois,2); //imprimer cotations
  objet.printMeanAndVar(meanvarmob3mois,0);//et dates et indices
  objet.printMeanAndVar(meanvarmob3mois,1);
  objet.printMeanAndVar(meanvarmob6mois,0);
```

```
  objet.printMeanAndVar(meanvarmob6mois,1);
  objet.printMeanAndVar(meanvarmob9mois,0);
  objet.printMeanAndVar(meanvarmob9mois,1);
  objet.printMeanAndVar(meanvarmob12mois,0);
  objet.printMeanAndVar(meanvarmob12mois,1);
}
```

L'appel de la méthode `doAll` permet l'impression de nombreuses moyennes mobiles, que nous avons dû enlever ici, mais qui sont dans le code présent à l'adresse http ://starbrood.free.fr/ProjetFinance/ . Notons que, si l'objet `eurus` de type `Outils` existe, `eurus.printMeanAndVar(meanvarmob7mois,k)` crée automatiquement le fichier `moymob7-eurus.dat` ou `varmob7-eurus.dat`, grâce au nombre de 0 présents dans `((meanvarmob7mois[i][k=0])_i)` (moyennes empiriques des incréments logarithmiques), et `((meanvarmob7mois[i][k=1])_i)` (variances empiriques) – le cas k=2 permettant d'imprimer le vecteur de dates (`KN<R> string`) avec les indices correspondants. Les fichiers générés seront détaillés avec la deuxième partie du projet lors des Simulations de cours d'actifs.

```
// PARTIE I ----------------- PARTIE I --------------
Outils eurus("eurus.dat",2650);
doAll(eurus);
Outils cac("cac.dat",2281);
doAll(cac);
Outils rcf("rcf.dat",2281);
doAll(rcf);

R2 usR2=eurus.getMeanAndVar(eurus.cotations);
R2 cacR2=cac.getMeanAndVar(cac.cotations);
R2 rcfR2=rcf.getMeanAndVar(rcf.cotations);
cout<<"mean and var des increments logarithmiques eur-us : "<<usR2<<endl;
cout<<"mean and var des increments logarithmiques du cac  "<<cacR2<<endl;
cout<<"mean and var increments logarithmiques  de rcf : "<<rcfR2<<endl;
cout<<"on obtient pour mu et sigma"<<endl;
cout<<"mean and var eur-us : "<<(usR2[0]+usR2[1]/2.)<<" "<<usR2[1]<<endl;
cout<<"mean and var cac : "<<(cacR2[0]+cacR2[1]/2.)<<" "<<cacR2[1]<<endl;
cout<<"mean and var rcf : "<<(rcfR2[0]+rcfR2[1]/2.)<<"  "<<rcfR2[1]<<endl;
```

1.3 Moyenne et Variance mobiles

Dans cette section, nous allons formaliser un peu le raisonnement et afficher les moyenne et variance sur les 3, 6, 9 et 12 mois précédents pour chaque pas de temps. Nous considérons, pour simplifier, qu'un mois contient 21 jours de cotations. On va donc faire jour après jour le calcul de la moyenne et de la variance empirique des incréments logarithmiques de l'actif sur les 3, 6, 9 ou 12 mois précédents. On note toujours $L(S)$ la série chronologique des incréments logarithmiques d'un actif S, i.e. $L(S)_k = \ln(S_{k+1}/S_k), 1 \leqslant k \leqslant N-1$ où N est le nombre de jours d'étude de S. Dans chaque cas, on définit $\tilde{\mathbb{E}}^i[L(S)]$ et $\tilde{\mathbb{V}}^i[L(S)], i = 3, 6, 9, 12$ les séries des moyennes et variances empiriques sur i mois, explicitement :

$$\begin{aligned} \tilde{\mathbb{E}}^i[L(S)_k] &= \frac{\sum_{h=k-21i}^{k-1} L(S)_h}{21i} \qquad (21i+1 \leqslant k \leqslant N-1) \\ \tilde{\mathbb{E}}^i[L(S)_k] &= 0 \qquad\qquad (1 \leqslant k \leqslant 21i) \end{aligned} \tag{5}$$

$$\begin{aligned} \tilde{\mathbb{V}}^i[L(S)_k] &= \frac{\sum_{h=k-21i}^{k-1} (L(S)_h - \tilde{\mathbb{E}}^i[L(S)_k])^2}{21i} \qquad (21i+1 \leqslant k \leqslant N-1) \\ \tilde{\mathbb{V}}^i[L(S)_k] &= 0 \qquad (1 \leqslant k \leqslant 21i) \end{aligned} \tag{6}$$

Nous avons, de la même manière que précédemment,

$$\begin{cases} (\mu - \frac{\sigma^2}{2}) & = \tilde{\mathbb{E}}^i[L(S)] \\ \sigma^2 & = \tilde{\mathbb{V}}^i[L(S)] \end{cases}$$

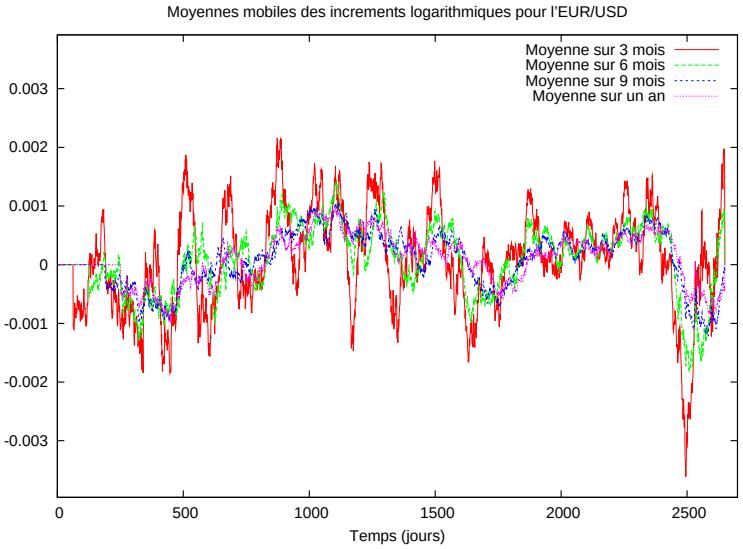
Ce qui donne :

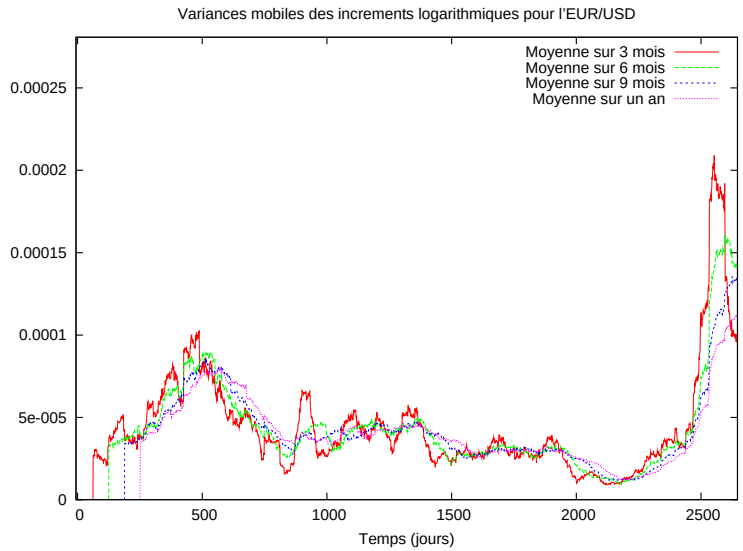
$$\begin{cases} \mu & = \tilde{\mathbb{E}}^i[L(S)] + \frac{\tilde{\mathbb{V}}^i[L(S)]}{2} \\ \sigma^2 & = \tilde{\mathbb{V}}^i[L(S)] \end{cases}$$

Et permet de calculer μ et σ (qui bien sûr dépendent de la période de glissement des données c'est-à-dire de i).

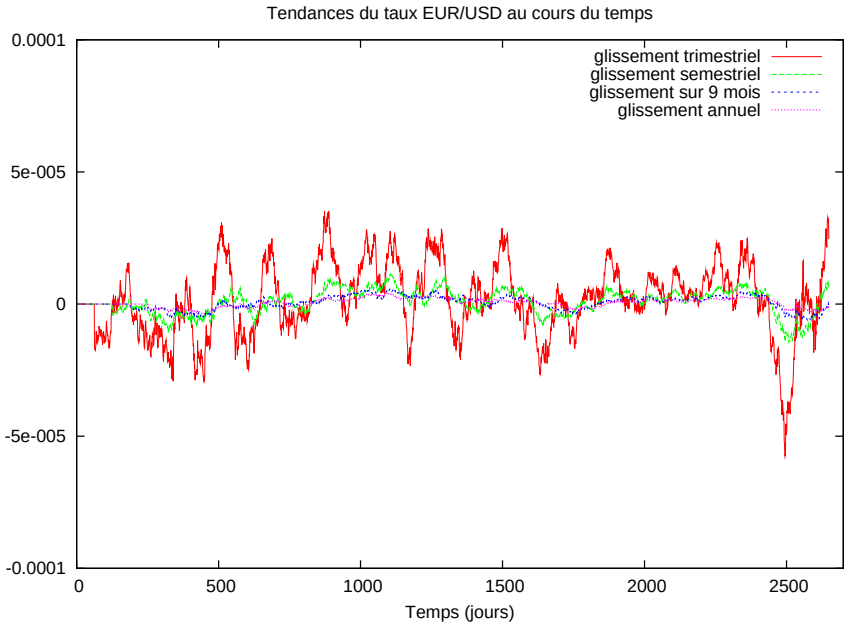
1.3.1 La parité Euro-Dollar

Nous présentons ci-dessous les moyennes et variances mobiles pour les incréments logarithmiques de la parité EUR/USD. Ces graphiques donnent surtout une idée du logarithme des rendements et de leur variance, mais nous nous en servons surtout pour le calcul numérique de μ et σ.

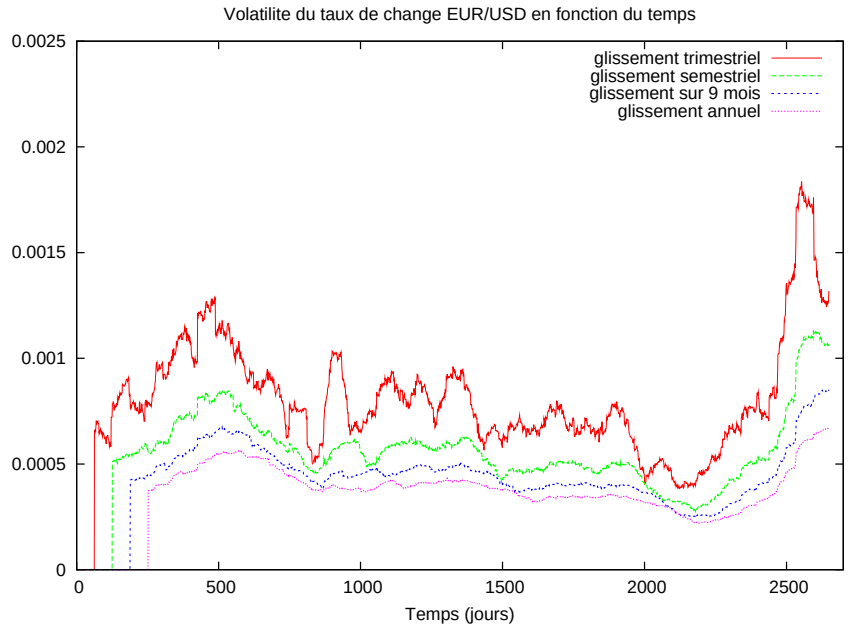




Pour les tendances et volatilités (μ et σ^2), nous utilisons les formules (5) et (6), qui dépendent du type de glissement des moyennes mobiles. Nous obtenons les résultats suivants pour la tendance :



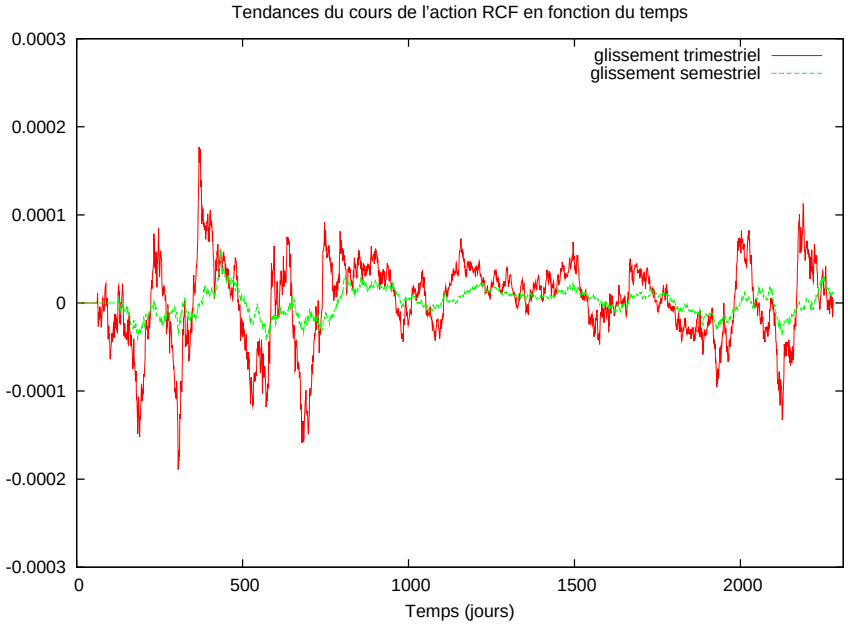
On observe que plus l'intervalle de temps qui sert au calcul de la tendance est grand, plus les variations sont faibles au cours du temps. Ceci est cohérent, car le rôle d'une moyene mobile est précisément de lisser les variations pour détecter la tendance. Pour les variances σ^2, nous obtenons pour les 4 cas les graphes suivants :

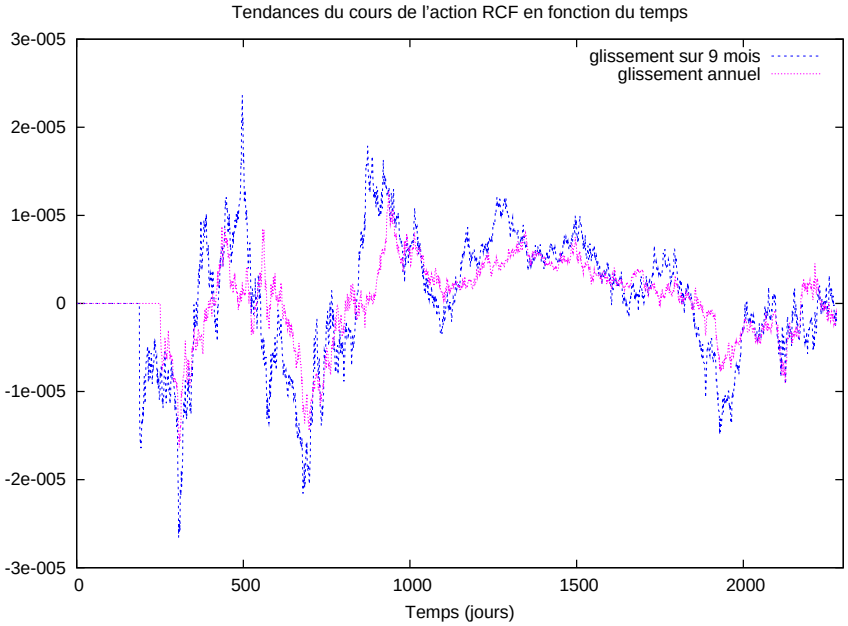


Sur ce graphique, on s'aperçoit que les courbes baissent à mesure que l'intervalle de glissement grandit. Ceci exprime le fait que la volatilité est plus faible autour de la tendance comme le laissait penser la figure précédente.

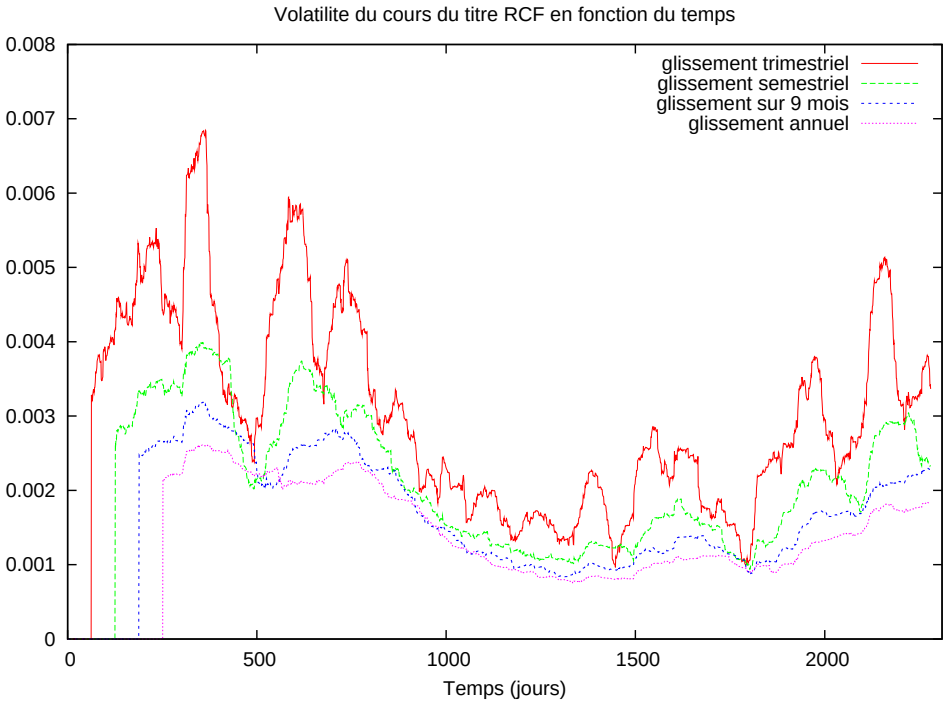
1.3.2 Le titre RCF

Nous présentons ci-dessous les résultats obtenus pour la tendance du cours de l'action Teleperformance, d'abord en glissements trimestriel et semestriel, puis en glissements sur 9 mois et 1 an, afin de pouvoir "zoomer" sur les deux courbes les plus lisses.



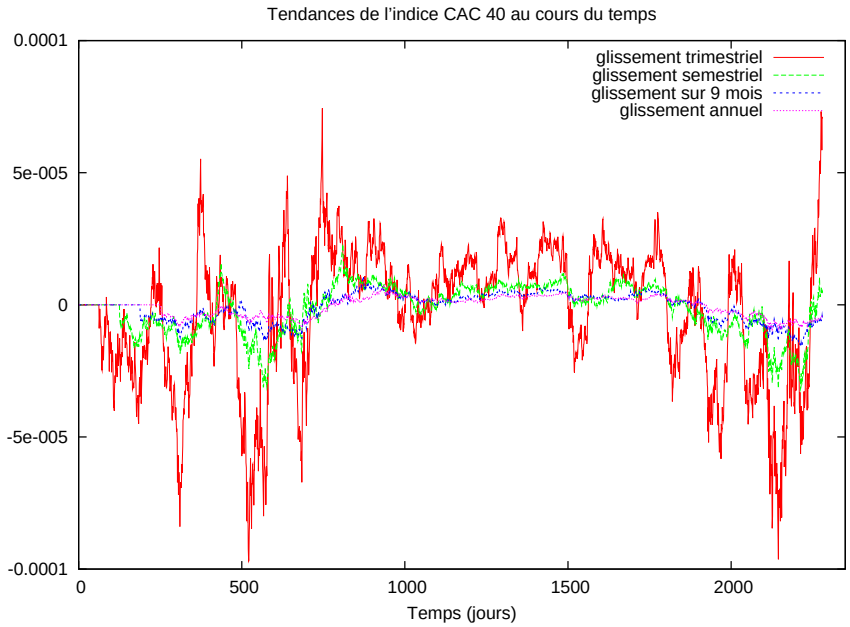


On voit qu'après un changement d'échelle, les tendances obtenues en glissement trimestriel ou annuel ont la même allure. Ci-dessous les résultats obtenus pour la volatilité σ^2.

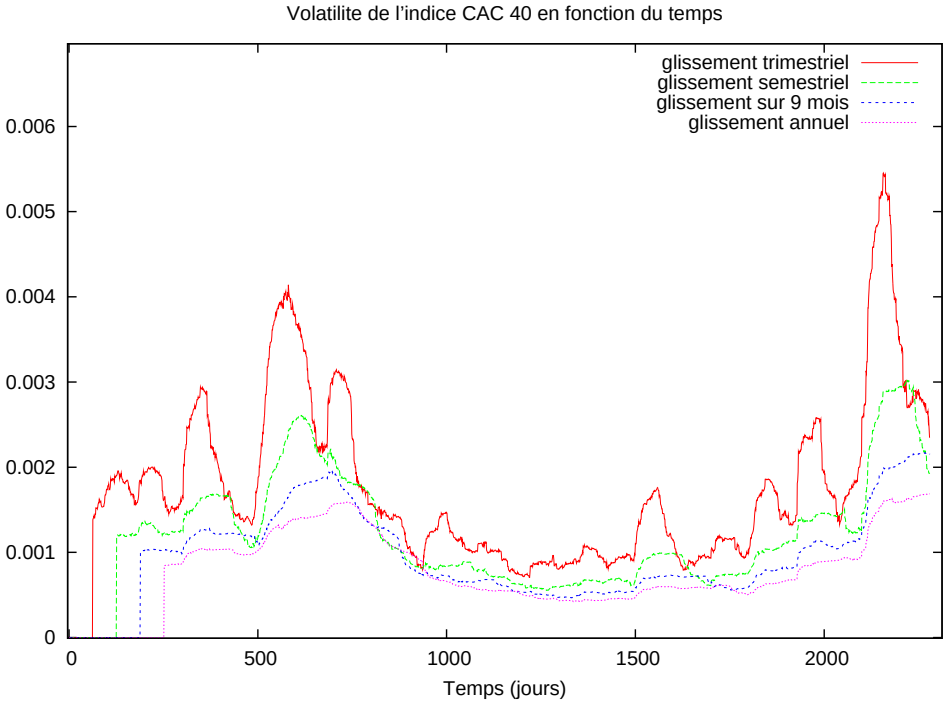


1.3.3 L'indice CAC 40

Les graphiques obtenus pour l'indice CAC 40 présentent les mêmes caractéristiques que ceux du titre RCF. Pour μ :



Et pour σ^2 :



On observe toujours un tassement des valeurs lorsque la période servant au calcul de la moyenne augmente.

2 Stratégies d'investissement

Nous nous intéressons maintenant à un actif financier, noté S_t, et considérons deux stratégies d'investissement : l'une consistant à acheter le titre et à le conserver jusqu'à la fin de la période sauf s'il passe sous un certain seuil, l'autre consistant à acheter une option d'achat sur le titre de maturité la fin de la période et de strike le prix initial.

2.1 Modélisation

2.1.1 Le schéma d'Euler-Maruyama implicite

Nous supposons que l'actif est régi par la dynamique (2). Une bonne manière de l'approcher est le schéma d'Euler-Maruyama implicite. nous allons donc discrétiser la période en N points, et nous choisirons ici $N = 252$ car nous travaillons sur une période d'un an soit environ 252 jours ouvrés. Nous considérons ensuite la chaîne de Markov $(Y_n, 0 \leqslant n \leqslant N)$ définie par $Y_0 = S_0$ et :

$$Y_{n+1} = Y_n + \mu Y_{n+1} + \sigma Y_n \Delta W_n \qquad (0 \leqslant n \leqslant N-1) \tag{7}$$

W_t est un mouvement Brownien Standard, et $\Delta W_n = W_{n+1} - W_n$ suit donc une loi $\mathcal{N}(0,1)$. Le schéma d'Euler-Maruyama est fortement et faiblement consistant, et converge fortement à l'ordre 0.5 et faiblement à l'ordre 1, on peut donc travailler convenablement.

2.1.2 Formule de Cox-Muller

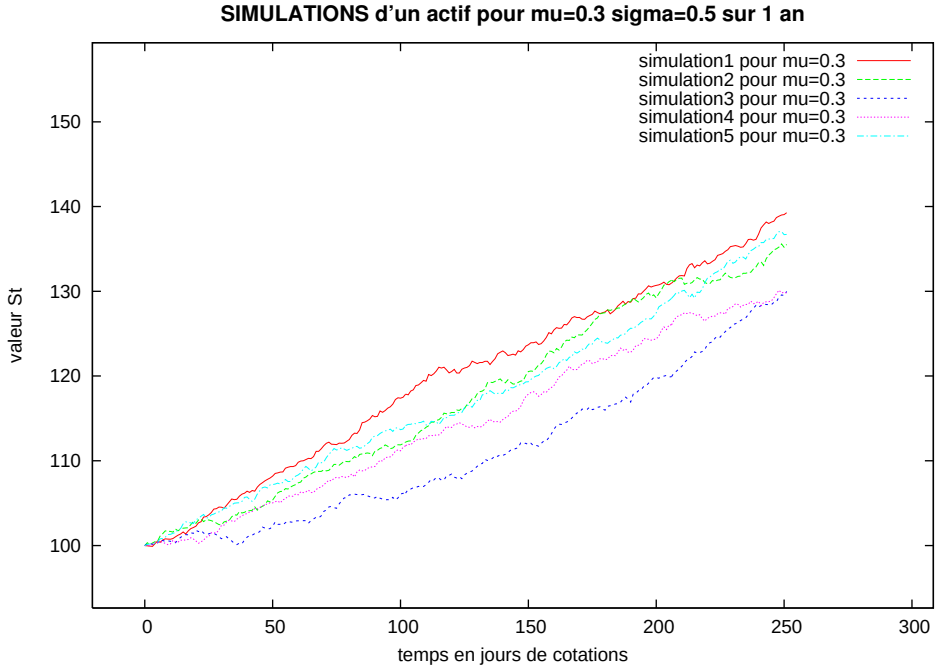
On utilisera, pour simuler une variable aléatoire de loi normale centrée réduite, la formule de Cox-Muller :

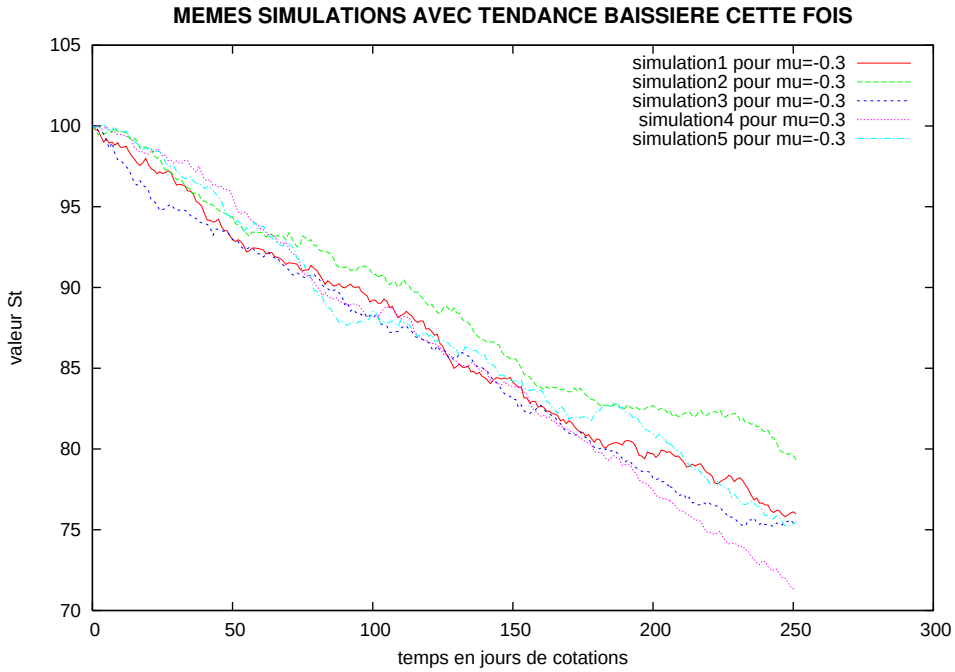
$$\Delta W_n = \sqrt{-2log(Z_1^n)}\cos(2\pi Z_2^n),$$

Où Z_1 et Z_2 sont deux nombres aléatoires générés à chaque étape de temps par la machine.

2.1.3 Simulations

Nous présentons ci-dessous plusieurs réalisations des simulations du cours de l'actif, pour $\mu = 0.3$ et $\mu = -0.3$.





2.2 Programmation

2.2.1 Cadre de travail

Nous allons donc simuler un grand nombre de réalisations de cours de notre actif S_t sur des périodes d'un an que nous considèrerons contenir 252 jours. On cherche donc à :

→ Simuler le cours d'un actif de paramètre μ et σ. (cf écriture de S_t en processus d'Itô).

→ Calculer le gain des stratégies $P1$ et $P2$ selon le cours final de S_t pour $P1$ (achat de call). Le $Gain_{P2}$ dépend de toute la filtration induite par toutes les variations de S_t durant les 252 jours de cotations de l'année. Rappelons que si l'actif passe sous le seuil $S_0 - C_0$, la perte est égale à $C0$ (la perte est égale à $S_0 - S_\tau$ ou $\tau = \{inft_i > 0, S_{t_i} \leq S_0 - C_0\}$. avec t_i un des N points de discrétisation.

→ Calculer et stocker $Gain_{P_1}$ et $Gain_{P_2}$ pour un nombre élevé de simulations : M.

→ Calculer les moyennes et variances empiriques des Gains à μ et σ fixés.

→ Effectuer ces calculs pour de nombreuses valeurs de μ telle qu'on puisse faire une représentation des moyennes et des variances des Gains en fonction de μ.

Pour une meilleure compréhension du code à venir, voici la liste des vecteurs et fichiers manipulés qui permettent des représentations graphiques sous Gnuplot. Pour ne pas encombrer notre répertoire de fichiers textes, les moyennes mobiles sur 3,6,9,12 mois seront enregistrées en colonnes dans un même fichier. Le tableau ci-dessous résume les fichiers que notre programme crée. Ces fichiers étant nombreux nous allons réduire leurs nombre pour la partie II.

Nom de fichier	Format	généré par défaut	Partie
table-eurus.dat	$[[jour_i][date][courseuros - dollar]$	code désactivé	1
moyMobX-eurus.dat	$[[jour_i][\sum ln(\frac{S_{i+1}}{S_i}) * \frac{1}{N}][VarEmpirique]$	code désactivé	1
tendances-eurus.dat	$[[jour_i][\mu_t^3]_t\ [\mu_t^6]_t\ [\mu_t^9]_t\ [\mu_t^{12}]_t$	oui	1
tendances-rcf.dat	$[[jour_i][\mu_t^3]_t\ [\mu_t^6]_t\ [\mu_t^9]_t\ [\mu_t^{12}]_t$	oui	1
tendances-cac.dat	$[[jour_i][\mu_t^3]_t\ [\mu_t^6]_t\ [\mu_t^9]_t\ [\mu_t^{12}]_t$	oui	1
volatilites-eurus.dat	$[[jour_t][\sigma_t^3]_t\ [\sigma_t^6]_t\ [\sigma_t^9]_t\ [\sigma_t^{12}]_t$	oui	1
volatilites-rcf.dat	$[[jour_t][\sigma_t^3]_t\ [\sigma_t^6]_t\ [\sigma_t^9]_t\ [\sigma_t^{12}]_t$	oui	1
volatilites-cac.dat	$[[jour_t][\sigma_t^3]_t\ [\sigma_t^6]_t\ [\sigma_t^9]_t\ [\sigma_t^{12}]_t$	oui	1
SimulationGainP1P2.dat	$[[Simulation_i][S_T^i]_i\ [Gain_{P1}^i]_i\ [Gain_{P2}^i]_i$	oui	2
MeanGainMuP1P2.dat	$[\mu_j]_j[Gain_{P1j}]_j\ [Gain_{P2j}]_j$	oui	2
VarGainMuP1P2.dat	$[\mu_j]_j\ [VarGain_{P1j}]_j\ [VarGain_{P2j}]_j$	oui	2
Alire !.txt	cf chapitre "Résumé des Résultats"	oui	1 et 2

$[[jour_i] \rightarrow$ désigne les indices des jours i.e $= 0, 1, ..., N-1$
$[Simulation_i \rightarrow$ désigne les indices des 250 simulations réalisées dans notre cas.
$\mu_{t\,t}^3\ \ \mu_t$ approximé sur 3 mois grâces aux moyennes et variances empiriques mobiles.
$\sigma_{t\,t}^3 \rightarrow$ variance calculée à partir des variances empiriques des incréments logarithmiques.
$\mu_{jj} \rightarrow$ vecteur faisant varier μ de -0.5 à 0.5 par sauts de pas h=0.02.
$Gain_{P1jj} \rightarrow$ gains moyens calculés avec 250 simulations pour les différentes valeurs de μ.
$[VarEmpirique] \rightarrow$ désigne la variance empirique des incréments $(\frac{S_{i+1}}{S_i})_{i=0,..,N}$
$moyMobX - eurus.dat \rightarrow$ désigne le vecteur des moyennes mobiles sur 1 de période glissante de X mois.

Pour le lecteur interessé,la commande de traitement de nos fichiers sous *Gnuplot* fonctionne ainsi. Pour tracer en 2D l'évolution des volatilités du Cac40 σ^{6mois} sur une période de 9 ans. *La volatilité σ^{3mois} sera initialisée à 0 pour les 6 premiers mois pour des raisons évoquées précédemments).*

```
plot "volatilites-cac.dat" u 1:3 title "`volatilite mobile" w p
```

2.2.2 Structure et Code C++ des Simulations : la classe Simulation

```
class Simulation{
    public :
        int M; //nombre de simulations
        int N; //nbre de jour (i.e échéance)
        R h; R C0;R K;
        R mu;R sigma;

        //vecteur des gains  P1 et P2 (pour chaque réalisation)
        KN<R> gainP1;KN<R> gainP2;
        //vecteur des S_T issus des M simulations
        KN<R> valfinals;

        //gain Moyen à mu et sigma fixé calculé à partir de M simulations
        //i.e à partir des M coordonnées de gainP1 et gainP2
        R gainMoyenP1;R gainMoyenP2;
        //idem pour variances empiriques des Gains
        R gainVarianceP1;R gainVarianceP2;

        //S0 commun à toutes nos simulations
        static const R S0=100.;

    //constructeur de la classe. M2 : nombre de simulations
    //N2 simulation sur N jours de cotations. mu,C0 et K en parametres
     Simulation(int M2,int N2,R mu2,R sigma2,R C0,R K2);

     //pour générer juste une simulation du cours St sur N jours
      // et l'imprimer dans le fichier de nom "fname"
     void plotSimulation(char* fname);

     //apres appel constructeur gainP1, gainP2 et valfinals sont remplis.
     //setGain() permet de calculer et stocker gainMoyenPi et gainVariancePi
     void setGains();

     //permet de générer SimulationsGainsP1P2.dat
     void plot();

     //les fichiers MeanGainMuP1P2 et VarGainMuP1P2 sont générés
     //dans le main à l'aide des méthodes ci-dessus
};
```

Les choix de programmations sont similaires à ceux de la partie I. La classe Simulation permet de réaliser M simulations de l'évolution d'un actif durant N jours. On choisit de stocker dans la classe uniquement les Gains à échéance de stratégies P1 et P2 dans les vecteurs `KN<R> GainP1(M) et KN<R> GainP2 \verb` ainsi que les valeurs finales S_T du sous jacent simulé numériquement. (stockées dans `KN<R> valsfinals(M)`). Puis les Gains moyens sont calculés et stockés uniquement après l'appel de la méthode `GainMoyenP1 et GainMoyenP2`. Le constructeur de cette classe calcule directement une simulation du sous-jacent grâce au schéma de Euler-Maruyama implicite qui nécessite de calculer des réalisations de lois normales centrées. Ces réalisations sont renvoyées par la méthode `R simulationNormale() \end{verbatim}` dont le code est donné dans le chapitre suivant. Voici le code du constructeur :

```
Simulation::Simulation(int M2,int N2,R mu2,R sigma2,R C02,R K2):M(M2),N(N2){
        mu=mu2;sigma=sigma2;C0=C02;K=K2;
        //variable qui permet de savoir si on passé le seuil So-Co
        R indiceSeuilC0=-1.;
        R Y; // solution courante qui évolue jusqu'au temp final

        //vecteurs auxiliaires qui seront stockés dans gainP1,gainP2,valsfinals
        KN<R> gainP1t(M);KN<R> gainP2t(M);KN<R> valfinalst(M);

      //boucle sur les différentes simulations
        for(int k=0;k<M;k++){
           Y=S0;int compteur=-1;

           //boucle sur le temps pour calcul de Y (sous-jacent)
           for(int i=0;i<N;i++){

              //cf description chapitre méthodes complémentaires
              R normale=simulationNormale();

              //schéma d'euler maruyama implicite
              R valeur=(1.+sigma*normale/((R)(N-1.)))/(1.-mu/((R)(N-1.)));
              Y=valeur*Y;

              //test pour calcul de gainP2 d'atteinte du seuil
              if(Y<S0-C0 && compteur==-1){
                  indiceSeuilC0=Y;
                  compteur=1; //pour retenir quel jour le cap a été franchi
              }
           }
          valfinalst[k]=Y;
          gainP1t[k]=max(Y-K,0.)-C0; //calcul GainP1
          if(indiceSeuilC0!=-1.){ //calcul perteP2 si seuil atteint
          gainP2t[k]=-(S0-indiceSeuilC0);
          }else{
          gainP2t[k]=Y-S0; //Gain ou Perte si tenu jusqu'à échéance
          }
       }
       //transfert dans les variables locales de la classe
       gainP1=gainP1t;
       gainP2=gainP2t;
       valfinals=valfinalst;
}
```

Les vecteurs `KN<R> GainP1t(M),KN<R> GainP2t(M),KN<R> valfinalst(M)` sont présents en tant qu'auxiliaire. Lors de la déclaration des vecteurs de type KN il faut préciser la taille, ce qui ne peut pas se faire dans le constructeur,ni dans le squelette quand bien même la taille de ces vecteur dépend du paramètre M, variable locale de la classe. Par contre la classe KN permet l'instanciation par copie. La complexité du constructeur est de l'ordre de N*M, c'est pourquoi la partie des calculs qui consistent à calculer les variables `GainMoyenP1 et GainMoyenP2` se fera par le simple appel de la fonction `objet.setGains() \verb` où `objet` est une instanciation de la classe `Simulation \verb`.

```
 void Simulation::setGains(){
      gainMoyenP1=0.;gainMoyenP2=0.;R aux1=0.;R aux2=0.;
      for(int i=0;i<M;i++){
              gainMoyenP1+=gainP1[i];
              gainMoyenP2+=gainP2[i];}
      gainMoyenP1=gainMoyenP1/((R)(M));
      gainMoyenP2=gainMoyenP2/((R)(M));


      for(int i=0;i<M;i++){
              aux1+=pow(gainP1[i]-gainMoyenP1,2);
              aux2+=pow(gainP2[i]-gainMoyenP2,2);}
      aux1=aux1/((R)(M));aux2=aux2/((R)(M));
      gainVarianceP1=aux1;gainVarianceP2=aux2;
}
 void Simulation::plot(){
        ofstream writer1("SimulationgainsP1P2.dat");
        for(int i=0;i<M;i++){
           writer1<<i<<" "<<valfinals[i]<<" "<<gainP1[i]<<" "<<gainP2[i]<<endl;
      }
}
```

2.2.3 Méthode Main et Utilisation des objets *Simulation*

```
int main(void)  {

  srand(time(NULL));
  //int a=getsize("eurus.dat");  //cf méthodes complémentaires
  //int b=getsize("cac.dat"); 2281 //calcul le nombre de jour max de
  //int c=getsize("RNO.dat"); //cotations présent dans les fichiers

  // PARTIE I
  Outils eurus("eurus.dat",2650);
  doAll(eurus);
  Outils cac("cac.dat",2281);
  doAll(cac);
  Outils rcf("rcf.dat",2281);
  doAll(rcf);
  R2 usR2=eurus.getMeanAndVar(eurus.cotations);
  R2 cacR2=cac.getMeanAndVar(cac.cotations);
  R2 rcfR2=rcf.getMeanAndVar(rcf.cotations);


  // PARTIE II ---------------- PARTIE II ----------------
  Simulation simule(500,252,0.1,0.5,20.96203,100.);

  /* pour simuler plusieurs simulations avec les mêmes parametres
  simule.plotSimulation("simulation03-1.dat");
  simule.plotSimulation("simulation03-2.dat");
  simule.plotSimulation("simulation03-3.dat");
  simule.plotSimulation("simulation03-4.dat");
  simule.plotSimulation("simulation03-5.dat"); */

  simule.plot();
  simule.setGains();
  //tentative de différenciation par schéma fini infructueuse
  //puisque réalisations indépendantes aléatoires
  //1./0.02 : pas de discrétisation de l espace de test pour mu
  KN<R> P1((int)(1./0.02));KN<R> P2((int)(1./0.02));
 // KN<R> P1v((int)(1./0.02));KN<R> P2v((int)(1./0.02));
 // KN<R> P1mderiv((int)(1./0.02));KN<R> P2mderiv((int)(1./0.02));
  int compteur=0;
  R C0=20.96203; //valeur du call calculé

//50 réalisations pour mu variant de 250 simulations sur 252 jours.
  for(R i=-0.5;i<=0.5;i=i+0.02){
        Simulation simule2(250,252,i,0.5,C0,100.);
        simule2.setGains(); //calcul gainMOyenPi
        P1[compteur]=simule2.gainMoyenP1;
        P2[compteur]=simule2.gainMoyenP2;

    /*utilisé pour la tentative de différenciation
    //    P1v[compteur]=simule2.gainVarianceP1;
    //    P2v[compteur]=simule2.gainVarianceP2;
   R pas=0.0001;
   Simulation simuleh(250,252,i+pas,0.5,20.96203,100.);
   simuleh.setGains();
   P1mderiv[compteur]=(simuleh.gainMoyenP1-P1[compteur])/pas;
   P2mderiv[compteur]=(simuleh.gainMoyenP2-P2[compteur])/pas;*/
        compteur++;
 }
```

```
 ofstream writera("MeanGainMuP1P2.dat");
 ofstream writerc("VarGainMUP1P2.dat");

 /* pour différentier le gain moyen...
 ofstream writere("P1derivMU.dat");
 ofstream writerf("P2derivMU.dat");
 */
 compteur=0;
 for(R i=-0.5;i<=0.5;i=i+0.02){
         writera<<i<<" "<<P1[compteur]<<" "<<P2[compteur]<<endl;
         writerc<<i<<" "<<P1v[compteur]<<" "<<P2v[compteur]<<endl;
   /* pour imprimer dérivées des gains par rapport à mu
         writere<<i<<" "<<P1mderiv[compteur]<<endl;
         writerf<<i<<" "<<P2mderiv[compteur]<<endl;
  */
         compteur++;
}
// Buffer stringstream qui sera affiché dans la console et imprimé
```

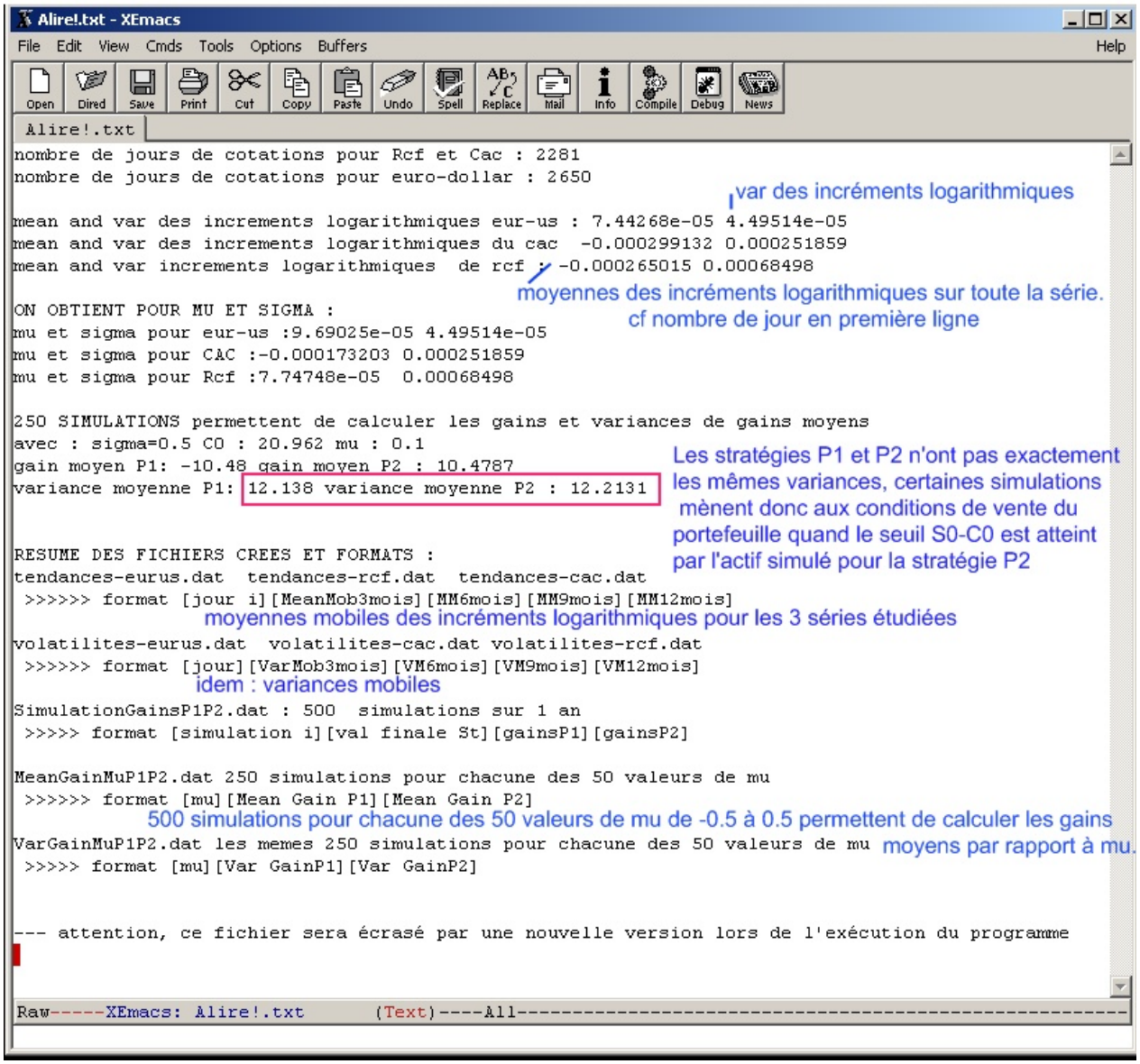
```
stringstream cout2;
cout2<<"nombre de jours de cotations pour Rcf et Cac : 2281"<<endl;
cout2<<"nombre de jours de cotations pour euro-dollar : 2650"<<endl<<endl;
cout2<<"mean and var des increments logarithmiques eur-us : "<<usR2<<endl;
cout2<<"mean and var des increments logarithmiques du cac  "<<cacR2<<endl;
cout2<<"mean and var increments logarithmiques  de rcf : "<<rcfR2<<endl<<endl;
cout2<<"ON OBTIENT POUR MU ET SIGMA :"<<endl;
cout2<<"mu et sigma pour eur-us :"<<(usR2[0]+usR2[1]/2.)<<" "<<usR2[1]<<endl;
cout2<<"mu et sigma pour CAC :"<<(cacR2[0]+cacR2[1]/2.)<<" "<<cacR2[1]<<endl;
cout2<<"mu et sigma pour Rcf :"<<(rcfR2[0]+rcfR2[1]/2.)<<"  "<<rcfR2[1]<<endl<<endl;


cout2<<"250 SIMULATIONS permettent de calculer les gains"<<endl;
cout2<<" et variances de gains moyens "<<endl;
cout2<<"avec : sigma="<<0.5<<" C0 : "<<C0<<" mu : "<<0.1<<endl;
cout2<<"gain moyen P1: "<<simule.gainMoyenP1<<" gain moyen P2 : ";
cout2<<simule.gainMoyenP2<<endl;
cout2<<"variance moyenne P1: "<<simule.gainVarianceP1;
cout2<<" variance moyenne P2 : "<<simule.gainVarianceP2<<endl<<endl<<endl;
cout<<cout2.str()<<endl;

cout2<<"RESUME DES FICHIERS CREES ET FORMATS : "<<endl;
cout2<<"tendances-eurus.dat  tendances-rcf.dat  tendances-cac.dat"<<endl;
cout2<<">>> format [jour i][MeanMob3mois][MM6mois][MM9mois][MM12mois]"<<endl<<endl;
cout2<<"volatilites-eurus.dat  volatilites-cac.dat volatilites-rcf.dat"<<endl;
cout2<<">>> format [jour][VarMob3mois][VM6mois][VM9mois][VM12mois]"<<endl<<endl;
cout2<<"SimulationGainsP1P2.dat : 500  simulations sur 1 an"<<endl;
cout2<<">>> format [simulation i][val finale St][gainsP1][gainsP2]"<<endl<<endl;
cout2<<"MeanGainMuP1P2.dat 250 simulations pour chacune des 50 valeurs de mu"<<endl;
cout2<<">>> format [mu][Mean Gain P1][Mean Gain P2]"<<endl<<endl;
cout2<<"VarGainMuP1P2.dat les memes 250 simulations";
cout2<<"pour chacune des 50 valeurs de mu"<<endl;
cout2<<">>> format [mu][Var GainP1][Var GainP2]"<<endl<<endl<<endl;
cout2<<"--- attention, ce fichier sera écrasé par une nouvelle version";
cout2<<" lors de l'exécution du programme"<<endl;

  ofstream write("Alire!.txt");
  write<<cout2.str();
  return 0;
}
```

2.2.4 Résumé des Résultats : le fichier $Alire!.txt$ généré en fin de programme



2.3 Analyse des sorties

2.3.1 Espérance de gain

Dans le cas de la stratégie par achat d'option (stratégie P1), l'investisseur choisit d'acheter une option d'achat européenne sur l'actif S_t, de prix d'exercice S_0, et d'échance $T = 1$. On rappelle que, par la formule de Black et Scholes, le prix en 0 d'une telle option d'achat sur le sous-jacent S_t supposé régi par la dynamique (2) est donné par :

$$C_0(S_0, K, r, T, \sigma) = S_0\mathcal{N}(d_1) - Ke^{-rT}\mathcal{N}(d_2) \tag{8}$$

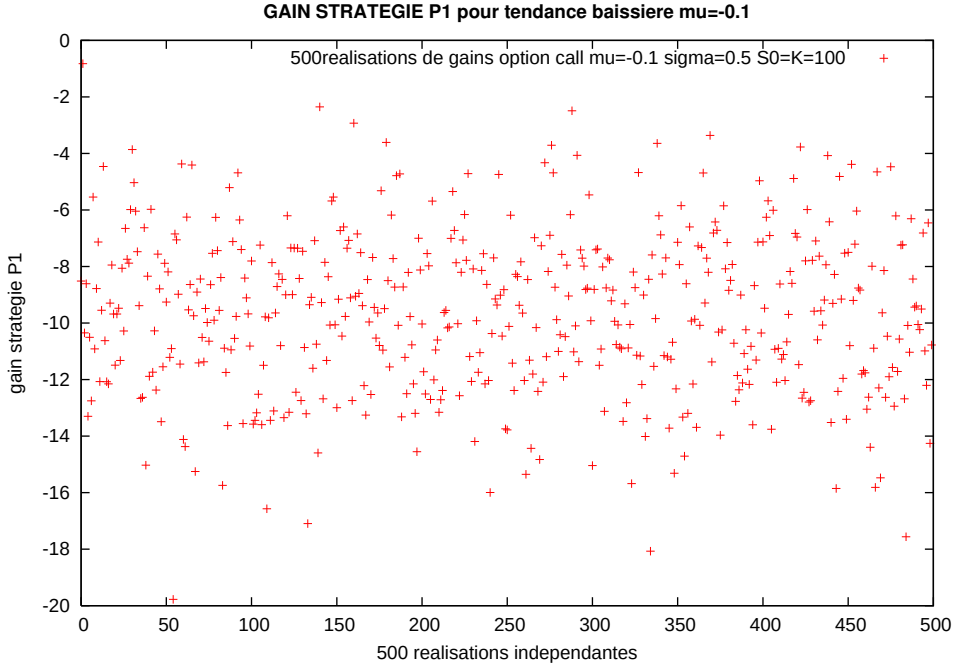
Où

$$d_1 = \frac{1}{\sigma\sqrt{T}}\left(\ln(\frac{S_0}{K}) + (r + \frac{\sigma^2}{2})T\right)$$
$$d_2 = d_1 - \sigma\sqrt{T}$$

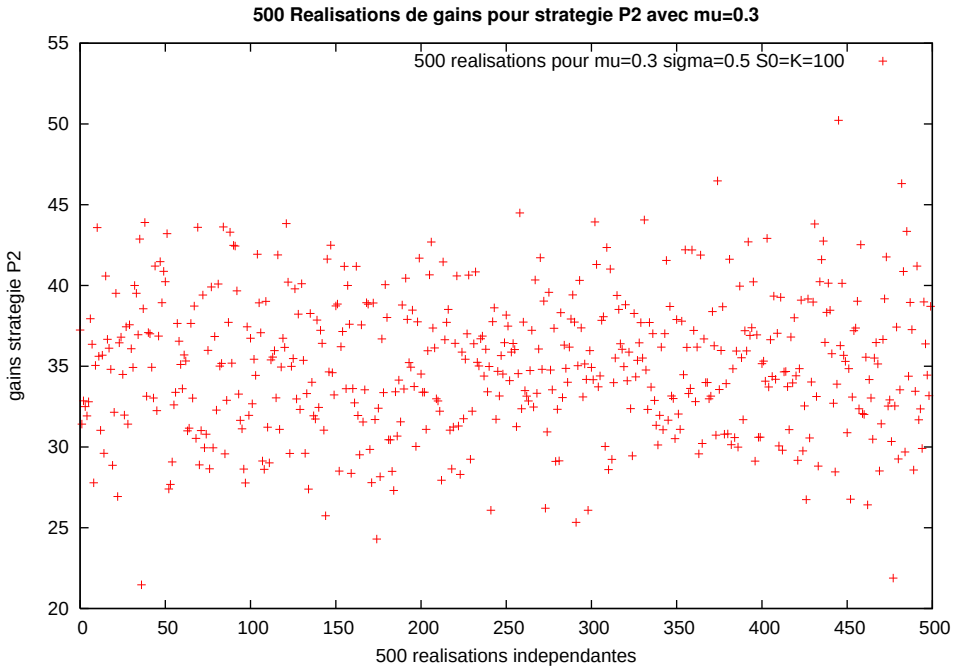
Et $\mathcal{N}(\cdot)$ est la fonction de répartition d'une loi normale centrée réduite. Avec les valeurs $r = 0.03,\ \sigma = 0.5,\ S_0 = K = 100$, le prix devient :

$$\begin{aligned} C_0 &= 100(\mathcal{N}(d_1) - e^{-0.03}\mathcal{N}(d_2)) \\ d_1 &= 0,31 \\ d_2 &= -0,19 \end{aligned} \tag{9}$$

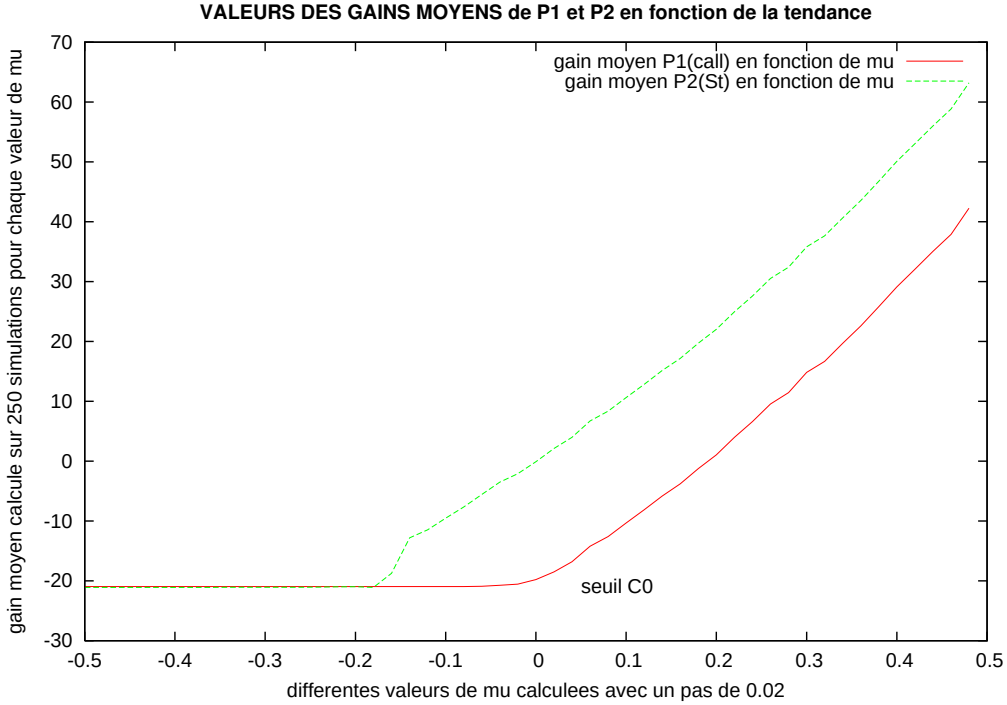
Et après calcul, on trouve $C_0 = 20,96203$. Le gain réalisé par une telle stratégie (éventuellement négatif), qui dépend naturellement de μ est donc $(S_T - S_0)^+ - C_0$. Pour l'estimer, nous faisons 500 simulations de la chaîne de Markov du processus discrétisé par le schéma d'Euler-Maruyama implicite, calculons le gain réalisé pour chaque réalisation puis faisons la moyenne de ces gains. En étudiant le gain pour chaque réalisation, du processus, nous obtenons un nuage de points très dispersés, ce qui confirme que les différents résultats sont totalement décorrélés. Par exemple, pour $\mu = -0.1$ (marché faiblement baissier),



La stratégie P2 consiste quant à elle à garder le titre, et à ne le vendre que si le cours tombe sous le seuil $S_0 - C_0$. Le gain réalisé en T dans ce cas est $(S_T - S_O)\mathbb{I}_{\{S_t \geqslant S_0 - C_0, 0 \leqslant t \leqslant T\}}$. Ce dernier dépend également de μ, et pour l'estimer nous faisons à nouveau 500 réalisations du gain en T puis en calculons la moyenne. Les 500 réalisations sont à nouveau parfaitement décorrélées, et pour $\mu = 0.3$, par exemple, toutes les réalisations présentent un gain positif :

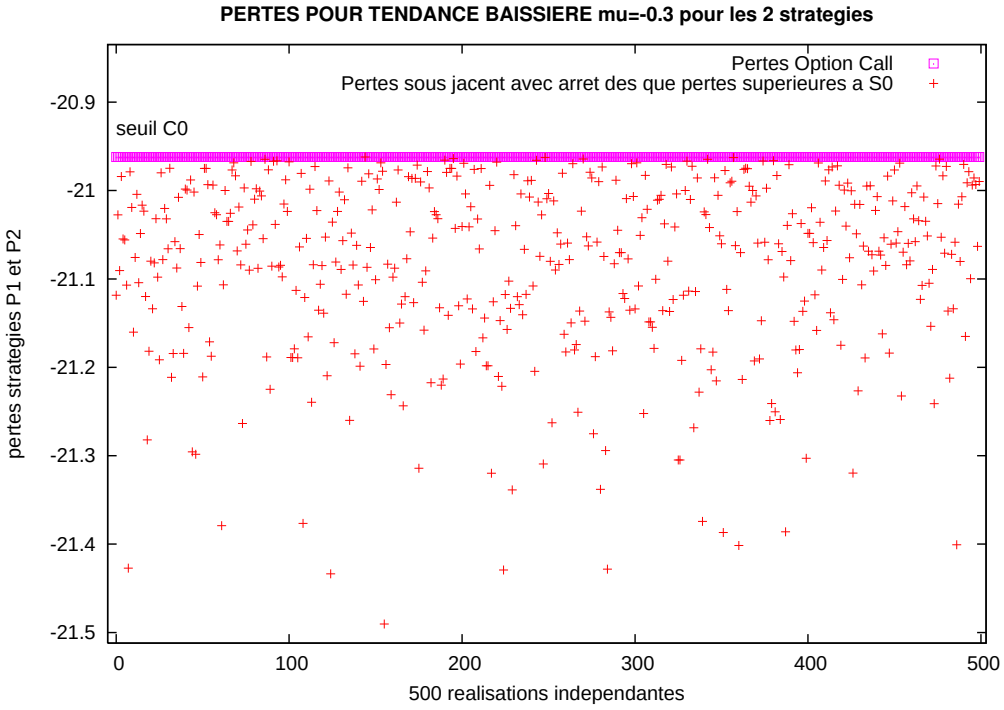


Finalement, nous présentons dans la figure suivante les gains moyens réalisés en fonction de μ, pour les stratégies P1 et P2.



On observe que pour des valeurs de μ négatives et suffisamment grandes en valeur absolue, les gains moyens associés aux deux stratégies sont grossièrement les mêmes : ils représentent la perte du montant C_0, qui est le niveau

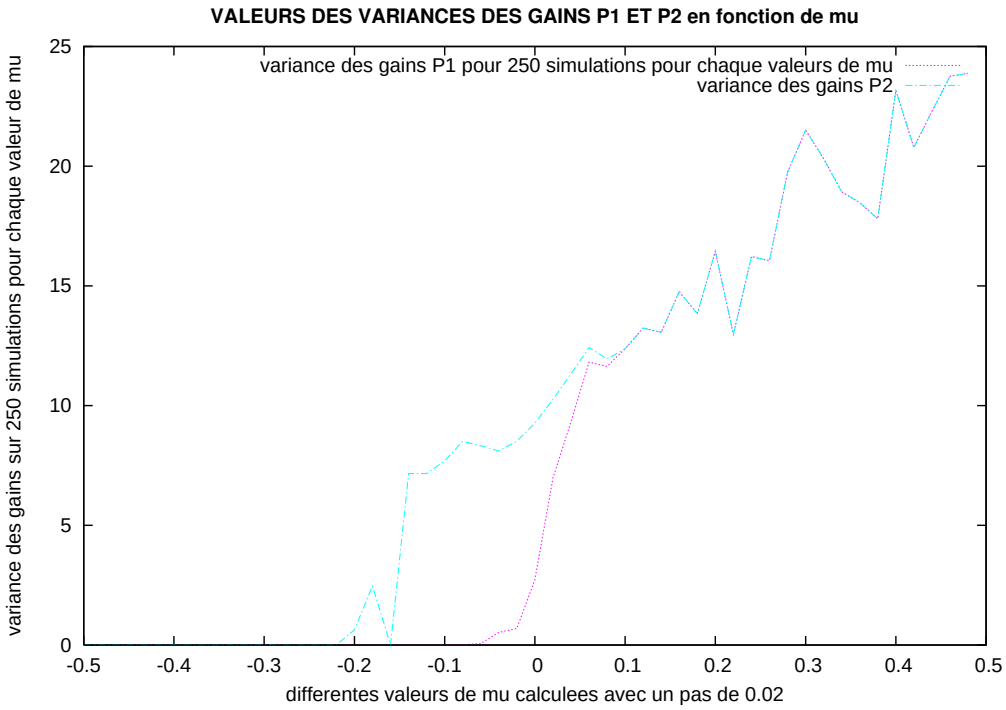
de perte maximal accepté par les deux stratégies. Lorsque la tendance μ augmente, le marché est de plus en plus haussier, et les gains moyens des deux stratégies s'en ressentent : dès que la valeur estimée de S_T dépasse $S_0 - C_0$, le gain moyen de la stratégie P2 augmente (mais pas linéairement, car même si $S_T \geqslant S_0 - C_0$, il peut exister une date où ce seuil est franchi à la baisse, dans ce cas le titre est vendu et la perte est de $-C_0$). Dès que le marché est suffisamment haussier et que la valeur estimée de S_T est supérieure à celle de S_0 (ce qui correspond à $\mu \geqslant 0$ par construction du processus), le call, dont le pay-off est linéairement fonction de S_T s'il n'est pas nul, devient payant, et le gain moyen procuré par la stratégie P1 commence à augmenter. Pour des valeurs suffisamment grandes de μ, le marché est très haussier et le cours du titre ne passe plus jamais le seuil $S_0 - C_0$ à la baisse. À partir de ces valeurs les évolutions en fonction de μ (à la hausse) des gains moyens des stratégies P1 et P2 sont parallèles. Finalement, à la vue de ce graphique, on s'aperçoit qu'on a plus à gagner à acheter le titre et à surveiller sa position plutôt qu'à acheter le call de strike S_0, dont la prime est élevée puisque la volatilité est élevée. Supposer que l'on exercera un call de strike S_0 est – à un facteur d'actualisation près, égal à $e^{-0.03}$ ici – équivalent à acheter aujourd'hui le titre pour le revendre, sauf que l'on ne commence à gagner (ou plutôt à rembourser ses pertes) que lorsque $S_T \geqslant S_0$ (contre $S_T \geqslant S_0 - C_0$ pour la stratégie P2). La stratégie P2 est donc a priori la meilleure. L'intérêt de la stratégie P1 réside cependant dans le fait de ne jamais dépasser le seuil de perte C_0. En effet, comme l'illustre la figure suivante montrant 500 réalisations de gain dans un marché fortement baissier ($\mu = -0.3$), le fait de vendre le titre dès que son cours tombe sous le seuil prévu ne protège pas contre les pertes de la dernière journée où l'on possède le titre. Si le cours s'effondre en une journée, le détenteur de la stratégie P1 vendra le titre à la clôture avec l'assurance que la perte engendrée sera supérieure ou égale à C_0, pouvant même être très supérieure si la baisse a été forte :



Avec cette tendance, toutes les projections sur le call prédisent qu'il ne sera pas payant, et engendrera donc une perte d'exactement C_0. La stratégie P2, elle, finira toujours par la vente du titre à perte, et l'on voit que de nombreuses réalisations du gain donnent une perte en résultant supérieure au prix du call, de parfois 40 ou 50 centimes, ce qui n'est pas négligeable si le nombre de titres à acheter est important.

2.3.2 Variance des gains

La variance des gains est calculée comme toute variance empirique, et en la traçant en fonction de μ, nous obtenons les résultats suivants.



On voit qu'à partir de valeurs de μ telles que le call est payant et que le cours du titre ne descend jamais au-dessous de $S_0 - C_0$, les variances des deux stratégies sont les mêmes, puisque les deux gains dépendent dans ce cas pareillement de l'évolution du cours, c'est-à-dire linéairement avec un facteur 1.