Université de Montréal

Système de transactions automatiques sur le marché des contrats à terme sur le taux d'intérêt

par

Julie Carreau

Département de mathématiques et de statistique
Faculté des arts et des sciences

Rapport de stage présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Finance mathématique et computationnelle

février 2001

© Julie Carreau, 2000

Université de Montréal
Faculté des études supérieures

Ce rapport de stage intitulé

Système de transactions automatiques sur le marché des contrats à terme sur le taux d'intérêt

présenté par

Julie Carreau

a été évalué par un jury composé des personnes suivantes :

René Garcia
(président-rapporteur)

Yoshua Bengio
(directeur de recherche)

Rapport de stage accepté le:

3 avril 2001

Le stage s'est déroulé à Hydro-Québec du mois de mai au mois d'août 2000, sous la supervision de Chakib Aabouche de la section d'ingégnierie financière de la Direction des Marchés Financiers.

L'objectif du stage est de développer un système pour transiger automatiquement dans le marché des contrats à terme sur le taux d'intérêt. Le système est basé sur des données intra-journalières à intervalle d'une heure. Il fournit donc la position à adopter sur le marché à chaque heure. La position est soit acheter un contrat, vendre un contrat ou rester neutre, c'est-à-dire ne pas effectuer de transactions. Le système de transactions proposé combine les règles d'analyse technique avec un réseau de neurones à une couche.

INTRODUCTION

Le contrat à terme sur le taux d'intérêt est un produit dérivé utilisé par Hydro-Québec pour se protéger contre certains risques financiers. Lors de l'émission d'obligations, Hydro-Québec s'engage à verser des coupons au taux d'intérêt sur le marché. Pour se prémunir contre les fluctuations du taux d'intérêt, Hydro-Québec contracte une entente (*swap*) avec un institut financier qui fixe le taux d'intérêt. Hydro-Québec paiera alors à l'institut financier le taux d'intérêt fixé par l'entente et ce dernier versera en retour aux détenteurs d'obligations le taux d'intérêt du marché. Le contrat à terme sur le taux d'intérêt entre en jeu lorsque les arbitragistes d'Hydro-Québec ont des "vues" sur les fluctuations du taux d'intérêt. Par exemple, s'ils prévoient une baisse du taux, alors Hydro-Québec est désavantagée par le *swap* et ils peuvent se protéger à l'aide de contrats à terme. Les techniques proposées pour transiger automatiquement sur le marché des contrats à terme sur le taux d'intérêt sont décrites brièvement dans les deux paragraphes qui suivent; il s'agit de règles d'analyse technique et de réseau de neurones.

Les règles d'analyse technique sont des outils graphiques ou mathématiques utilisés pour transiger dans le marché financier. Ces règles s'appliquent aux séries de prix d'instruments financiers; elles indiquent les tendances dans le mouvement des prix et fournissent donc la position à prendre dans le marché pour un instrument financier donné. Les règles d'analyse technique utilisées ici sont du type "mathématique" : elles résultent de calculs simples (par exemple, des moyennes mobiles) effectués sur les séries de prix. Ces règles donnent une série de signaux qui correspondent à une position à prendre dans le marché.

Le réseau de neurones est utilisé afin de combiner l'information véhiculée par les signaux des règles d'analyse technique, en espérant que la combination produite performe mieux que chacune des règles prises indivuduellement. Le réseau prend en entrées les signaux provenant des règles d'analyse technique. Chaque entrée est un vecteur contenant les positions recommandées par chacune des règles. Le réseau fournit en sortie un vecteur de probabilité : chaque élément du vecteur indique la probabilité qu'une position dans le marché soit optimale (au sens où elle rapporte le plus grand gain possible). Ce vecteur de probabilité est obtenu en appliquant la fonction *softmax* (cette fonction est décrite plus loin) à la somme pondérée des signaux des règles d'analyse technique. Les poids de la somme sont obtenus en maximisant l'espérance de gain produite par le réseau de neurones. Le réseau de neurones ne contient pas de couche cachée; il s'agit en fait de régression logistique.

Dans la section 1, les données ayant servi aux expériences sont décrites. Dans la section 2, l'utilisation des règles d'analyse technique est approfondie. Puis, dans la section 3, les détails du fonctionnement du réseau de neurones sont expliqués. Finalement, dans la section 4, les résultats de différentes simulations sont exposés.

1 : LES DONNÉES

Les données utilisées sont les séries de prix des contrats à terme sur le taux d'intérêt canadien (BAX). Ces données sont disponibles sur le site internet de la bourse de Montréal (*http://www.bdm.org/fr.stat.html*). Ce sont des données intra-journalières : elles proviennent des prix de toutes les transactions qui sont effectuées sur les BAX pendant une journée donnée à la bourse de Montréal. Les contrats à terme sur le taux d'intérêt ont quatre mois d'échéance possibles durant l'année pour lesquels il y a une certaine liquidité : mars (H), juin (M), septembre (U) et décembre (Z). On distingue donc les contrats à terme selon leur année et leur mois d'échéance. Par exemple, "BAXU97" désigne le contrat à terme sur le taux d'intérêt canadien ayant comme échéance le mois de septembre 1997.

Les données sont fournies en format texte et ont été chargées dans Access pour construire une base de données. Ensuite, les données ont été triées pour s'assurer qu'elles étaient bien classées par contrat, par date et par heure. Les données ont ensuite été divisées selon l'année et la date d'échéance du contrat : chaque table de la base de données contient les prix relatifs à un seul contrat à terme (par exemple, "BAXU97"). Finalement, les données ont été traitées pour que les prix soient donnés à intervalle de cinq minutes (originalement, l'intervalle était aléatoire, l'heure fournie est simplement l'heure de la transaction). Lorsqu'on veut savoir le prix disons à 8:05, le prix qui apparaît en fait dans la base de données est le prix de la dernière transaction la plus proche de 8:05. À cette étape, chaque table Access est retransformée en fichier texte pour pouvoir être facilement chargé dans Matlab (le logiciel utilisé pour la partie programmation). Chaque fichier texte se présente donc sous la forme "BAXM96.txt" et contient tous les prix du contrat à terme ayant comme échéance le mois de juin 1996 à intervalle de cinq minutes.

À Hydro-Québec, la gestion se fait sur une année et seuls les contrats à terme de l'année en cours sont considérés. Par exemple, pour l'année 1996, on s'intéresse aux prix des contrats à terme ayant leur échéance en 1996 et on s'intéresse uniquement aux prix de ces contrats pendant l'année 1996. Une série de prix est construite pour l'année en cours de la façon suivante: le contrat ayant l'échéance la plus courte est transigé (au début de l'année, il s'agit du contrat de mars), le contrat n'est plus transigé deux semaines (10 jours ouvrables) avant l'échéance, on transige alors le contrat suivant ayant l'échéance la plus courte. Dans l'ordre, le contrat de mars est transigé jusqu'à deux semaines avant l'échéance, ensuite les contrats de juin, de septembre et de décembre. Les transactions pour l'année en cours se terminent donc deux semaines avant l'échéance du contrat de décembre. On cesse de transiger un contrat deux semaines avant l'échéance car son prix a déjà convergé vers la valeur du contrat à terme à l'échéance.

Le peu de volume des contrats BAX ne justifient pas l'utilisation d'un intervalle de temps aussi petit que cinq minutes pour échantillonner la série de prix. Comme il y a peu de transactions (beaucoup moins qu'à intervalle de cinq minutes), l'estimation du prix aux cinq minutes est peu fiable (le même prix est répété pour plusieurs intervalles). Un intervalle d'une heure a donc été utilisé. Le tableau 1 contient quelques statistiques sur les données échantillonnées à intervalle d'une heure et le graphique 1 illustre chacune des séries pour les années 1993 à 1999.

La première figure montre une série de prix globalement croissante pour l'année 1993. Pour l'année 1994, la série décroît puis croît de nouveau mais il y a beaucoup de variance. Pour les années 1995-1996, la série de prix est globalement croissante, elle est relativement volatile bien que moins qu'en 1994. Les trois dernières années sont celles où on a observé le moins de variance; le niveau de la série est relativement stable, on n'observe donc pas de mouvement de croissance ou de décroissance.

2 : L'ANALYSE TECHNIQUE

Quatre types de règles d'analyse technique ont été utilisés : la règle des deux moyennes mobiles, la règle de la moyenne mobile avec filtre, la règle de la moyenne mobile exponentielle avec facteur d'oubli adaptatif et la règle du support-résistance.

La moyenne mobile

Une moyenne mobile est une moyenne calculée sur une fenêtre de largeur fixe que l'on fait glissée sur l'axe du temps. La moyenne est alors comparée avec la valeur de la série qui suit immédiatement la fenêtre utilisée pour le calcul de la moyenne mobile.

La règle aux deux moyennes mobiles

Une règle aux deux moyennes mobiles utilise une moyenne mobile ayant une fenêtre longue et une moyenne mobile ayant une fenêtre courte. Un signal d'achat est enregistré lorsque la moyenne mobile courte croise la moyenne mobile longue vers le haut et inversement un signal

de vente est enregistré lorsque la moyenne mobile courte croise la moyenne mobile longue vers le bas. Cette règle ne donne pas de position neutre. Les paramètres libres de la règle aux deux moyennes mobiles sont les largeurs des fenêtres de chaque moyenne mobile.

La règle de la moyenne mobile avec filtre

La règle de la moyenne mobile avec filtre utilise un bande autour de la moyenne mobile qui est proportionnelle à l'écart-type de la série des prix. Lorsque la série est au-dessus de la moyenne mobile et que l'écart entre la moyenne mobile et la série est plus grand que la valeur de la bande, alors c'est un signal d'achat. Lorsque la série est au-dessous de la moyenne mobile et que l'écart entre la moyenne mobile et la série est plus grand que la valeur de la bande, alors c'est un signal de vente. Lorsque la distance entre la moyenne mobile et la série de prix est plus petite que la valeur de la bande, alors le signal est la position neutre. Les paramètres libre de cette règle sont la fenêtre de la moyenne mobile et la proportion de l'écart-type utilisée pour former la bande.

La moyenne mobile exponentielle

La valeur d'une moyenne mobile exponentielle au temps t est calculée en pondérant toutes les valeurs de la série jusqu'au temps t: $m_t = \alpha m_{t-1} + (1-\alpha)p_{t-1}$ où m est la moyenne mobile exponentielle, p est la série de prix et $\alpha \in [0,1]$ est le *facteur d'oubli*. Plus l'observation est distante dans le temps, et plus sa contribution au calcul de la moyenne mobile est petite. Le paramètre libre d'une moyenne mobile exponentielle est son facteur d'oubli.

La règle de la moyenne mobile exponentielle avec facteur d'oubli adaptatif

Le facteur d'oubli est dit *adaptatif* lorsqu'il varie en fonction de la capacité des valeurs passées de la série de prix à prédire la prochaine observation. La capacité de prédiction est estimée par la statistique r^2 de la régression linéaire. Plus r^2 tend vers 1 (plus le passé prédit bien le présent), plus α augmente (on oublie moins) et inversement. Le paramètre libre de cette règle est la largeur de la fenêtre sur laquelle on fait la régression linéaire.

La règle du support-résistance

La règle du support-résistance calcule le minimum et le maximum atteint par la série de prix sur une fenêtre de temps fixe. Le minimum détermine la ligne de support et le maximum la ligne de résistance. Lorsque le prix dépasse la ligne de résistance, on obtient un signal d'achat. Lorsque le prix est au-dessous de la ligne de support, on obtient un signal de vente. Lorsque le prix se situe entre les lignes de support et de résistance, le signal est la position neutre. Le paramètre de cette règle est la fenêtre sur laquelle le minimum et le maximum de la série de prix sont calculés. La figure 2 illustre le comportement de ces quatre règles d'analyse technique.

Optimisation des règles d'analyse technique

Les paramètres de chacune des règles d'analyse technique peuvent être optimisés. Le critère d'optimalité est le gain total obtenu par la règle. Lors de l'optimisation, plusieurs paramètres initiaux sont générés de façon aléatoire. À partir de chaque ensemble de paramètres initiaux, une descente locale est effectuée : la valeur du gain total obtenu avec les paramètres initiaux est comparée avec la valeur du gain total obtenu avec les paramètres voisins, les paramètres voisins donnant la valeur de gain la plus élevée sont retenus. La comparaison est recommencée à partir de ces nouveaux paramètres. La descente locale est terminée lorsqu'aucun des paramètres voisins ne donnent une valeur de gain total plus grande que les paramètres précédants. L'algorithme d'optimisation est conçu de façon à ne pas passer deux fois par les mêmes paramètres. Lorsque toutes les descentes locales sont terminées, les paramètres retenus sont ceux ayant donné le plus grand gain total parmi toutes les descentes locales. L'optimisation est difficile à cause de la présence de nombreux minima locaux.

3 : La régression logistique

Soit $p(a|x_t)$, la probabilité que l'action a soit optimale (au sens où elle rapporte le plus grand gain) à l'instant t étant donné le vecteur de signaux provenant des règles d'analyse technique x_t. Les actions possibles considérées sont : acheter un contrat, vendre un contrat et rester neutre. Ces actions sont représentées respectivement par $a = 1$, $a = -1$ et $a = 0$.

Deux choix ont été envisagés pour le vecteur x_t. D'abord, x_t contient autant d'éléments que de règles d'analyse technique sont considérées et les éléments de x_t appartiennent à l'ensemble $\{-1,0,1\}$. C'est-à-dire, les signaux des règles d'analyse technique donnent seulement la position à prendre dans le marché. Les paramètres de ces règles sont optimisés par rapport au critère de gain maximal. Ensuite, les éléments de x_t appartiennent à $\mathcal{R}$. Les règles d'analyse technique renvoient la position à prendre sur le marché (qui est en fait le signe du signal) et la force de cette position (l'amplitude du signal). Dans ce cas-ci, les paramètres des règles d'analyse technique ne sont pas optimisés. On génère plutôt aléatoirement un grand nombre de paramètres possibles pour chacune des règles. Le vecteur x_t est beaucoup plus long, il contient en fait les signaux de chacune des règles pour chacun des ensembles de paramètres générés.

Soit maintenant le vecteur de probabilité fourni par le réseau de neurone au temps t :

$$\hat{p}(x_t) = softmax(w * x_t + b)$$

Où w est la matrice de poids de dimension 3 x n, n est le nombre de signaux par unité de temps et b est le vecteur de biais de dimension 3 x 1. On peut introduire le vecteur de biais dans la matrice de poids en posant :

$$\tilde{w} = [b,w]$$
$$\tilde{x_t} = [1,x_t']'$$

Alors, on peut écrire :

$$\hat{p}(x_t) = softmax(\tilde{w} * \tilde{x_t})$$

La matrice $\tilde{w}$ est de dimension 3 x $(n + 1)$, la première colonne de cette matrice correspond alors au vecteur de biais b.

Chacun des éléments du vecteur $\hat{p}(x_t) = [\hat{p}(a_k|x_t)]_{k=1,2,3}$ est défini par :

$$\hat{p}(a_k|x_t) = \frac{\exp(\tilde{w_k} * \tilde{x_t})}{\sum_{j=1}^{3} \exp(\tilde{w_j} * \tilde{x_t})}$$

Où $k = 1,2,3$, $a_k \in \{-1,0,1\}$ et $\tilde{w_j}$ désigne la $j^{\text{ième}}$ ligne de la matrice $\tilde{w}$.

La matrice $\tilde{w}$ est obtenue en minimisant une fonction de coût. La fonction de coût choisie est l'espérance de la perte (c'est-à-dire moins le gain) occasionnée par la régression logistique :

$$\int\int L(a,\Delta p(t))p(a|x(t))dadt$$

Où $p(t)$ est le processus des prix des contrats à terme, $L(a.\Delta p(t)) = c|a| - a\Delta p(t)$ est la perte occasionnée par l'action a alors que le changement dans le prix est $\Delta p(t)$ et le coût de transaction est c et $p(a|x(t))$ est la probabilité que l'action a soit optimale. En pratique, $\tilde{w}$ est approximée par :

$$\hat{w} = \underset{w}{argmin} \sum_{t=2}^{T} \sum_{k=1}^{3} L(a_k,\Delta p_t)\hat{p}(a_k|x_t)$$

Où T est la longueur de la série de prix considérée. La minimisation de la fonction de coût est effectuée à l'aide de l'optimiseur de Matlab *fminunc* à laquelle une expression analytique pour le gradient est fournie.

Une fois la matrice de poids $\tilde{w}$ déterminée par minimisation de la fonction de coût, le signal au temps t fourni par le réseau de neurone est l'action a_i telle que $\hat{p}(a_i|x_t)$ est maximale, c'est-à-dire, la probabilité que l'action a_i soit optimale est la plus grande au temps t.

4 : EXPÉRIENCES

Deux types d'expériences ont été mises en oeuvre, ils correspondent aux deux choix du vecteur de signaux x_t. Dans les deux cas, on a supposé que le coût de transaction est zéro ($c = 0$).

Expérience 1

Le premier type d'expérience nécessite l'optimisation des paramètres de chaque règle d'analyse technique. Une fois les paramètres obtenus, on crée une matrice de signaux dont chaque ligne est le vecteur de signal d'une règle d'analyse technique pour la série de prix considérée. On fournit ensuite cette matrice de signaux au réseau de neurones pour lequel il faut déterminer la matrice de poids $\tilde{w}$ en minimisant l'espérance de la perte. Ensuite, on peut tester la performance du réseau de neurones en calculant le gain généré par les signaux résultant du réseau de neurones sur une nouvelle série de prix.

L'expérience va comme suit :

1. Les paramètres des règles d'analyse technique sont déterminés par optimisation sur la série de prix A.
2. Une matrice de signaux de dimension 4 x T est généré à partir des règles d'analyse technique pour la série A.
3. La matrice de signaux est donnée en entrée au réseau de neurones et la matrice de poids $\tilde{w}$ est obtenue par minimisation de la fonction de perte toujours sur la série A.
4. La performance du réseau de neurones est testée en calculant le gain obtenu par le réseau sur une nouvelle série de prix B.

Dans la première colonne du tableau 2, on retrouve les séries ayant servi à l'optimisation des paramètres et dans la deuxième colonne, on retrouve les séries ayant servi à tester les paramètres. Chaque série est décrite par les contrats qu'elle contient et les séries indiquées ont été formées pour chaque année (1993 à 1999). Les gains obtenus sur chacun de ensembles d'entraînement et ceux obtenus sur les ensembles de test correspondant avec l'utilisation des paramètres déterminés sur les ensembles d'entraînement sont présentés dans le tableau 3. Cette expérience correspond à une stratégie qui utilise les contrats de l'année en cours dont l'échéance est dépassée pour optimiser les paramètres des règles d'analyse technique et ceux du réseau de neurones. Ensuite, ces paramètres sont utilisés pour créer de nouveaux signaux de règles d'analyse technique. Ceux-ci sont fournis en entrées au réseau de neurones et ce dernier donne en sortie la position qu'il recommande sur le marché. À chaque heure, le système calcule les signaux des règles d'analyse technique, ensuite il peut obtenir la réponse du réseau de neurones quant à la position à prendre sur la marché. Lorsqu'un contrat échoit, le système remet à jour ses paramètres en recommançant le processus d'optimisation sur la série de prix incluant les contrats de l'année en cours qui sont déjà échus.

Dans le tableau 3, la série A (l'ensemble d'entraînement) représente la série sur laquelle l'optimisation des règles d'analyse technique et la minimisation de la fonction de coût du réseau de neurones sont faites. La série B (l'ensemble de test) est la série sur laquelle les paramètres trouvés à partir de la série A sont testés : on calcule le gain obtenu sur la série B avec les règles d'analyse technique et le réseau de neurones et leurs paramètres déterminés à l'aide de la série A. La colonne 1 représente le gain obtenu avec la règle de la moyenne mobile exponentielle avec facteur d'oubli adaptatif, la colonne 2 représente le gain obtenu avec la règle aux deux moyennes mobiles, la colonne 3, celui obtenu avec la règle de la moyenne mobile avec filtre et la colonne 4 celui obtenu avec la règle du support-résistance. La colonne RN représente le gain du

réseau de neurones et la colonne B & H le gain d'une stratégie de "buy & hold". Une stratégie de "buy & hold" consiste à acheter le contrat à terme au début de la période considérée et à le garder jusqu'à deux semaines avant l'échéance. Ensuite, le contrat suivant est acheté et on procède de cette façon jusqu'à la fin de la période d'intérêt. Le gain est comparable entre éléments d'une même ligne : le nombre de données varie d'une ligne à l'autre (avec la taille de la série considérée) et un gain plus grand sur deux lignes différentes n'indique pas nécessairement une règle plus performante.

Les performances du réseau de neurones sur les séries de test ne sont pas très convaincantes : il bat la stratégie de "buy & hold", mais peu souvent : 16 fois sur 21 sur les séries d'entraînement et 6 fois sur 21 sur les séries test. De plus, le réseau n'est pas systématiquement meilleur que les règles d'analyse technique : le réseau obtient un gain supérieur à ceux des règles d'analyse technique 8 fois sur 42. Ces dernières ne donnent pas des performances beaucoup plus intéressantes sur les séries de test : les résultats ne permettent pas de conclure de la supériorité d'une règle par rapport à la stratégie de "buy & hold". Pour certaines séries (d'entraînement et de test), le réseau de neurones donne exactement la même performance que la première règle d'analyse technique, c'est-à-dire la règle de moyenne mobile exponentielle avec facteur d'oubli adaptatif.

Le tableau 4 contient les performances moyennes annuelles de chacune des règles d'analyse technique, du réseau de neurones et de la stratégie de "buy & hold". La dernière ligne du tableau inclut la moyenne des performances annuelles pour chacune des stratégies. En moyenne, le réseau performe mieux que la stratégie de "buy & hold" mais, il y a des années où cette stratégie donne un gain plus élevé que le gain obtenu par le réseau de neurones. La troisième règle d'analyse technique, la moyenne mobile avec filtre, donne un gain moyen sur toutes les années plus élevé que celui du réseau de neurones. La règle d'analyse technique qui performe le moins bien dans la majorité des cas est la règle du support-résistance.

Les paramètres obtenus par optimisation lors de l'étape 1 sont présentés pour chaque série d'entraînement dans le tableau 5. Certains paramètres ne semblent pas très appropriés. Par exemple, dans le cas de la règle du support-résistance (règle 4), l'intervalle sur lequel le maximum et le minimum sont calculés peut être aussi petit que deux périodes. Dans ce cas-ci, cela signifie qu'on se base sur les deux dernières heures pour déterminer la position dans le marché, ce qui n'est pas très réaliste. Le même problème est observé pour la règle de la moyenne mobile exponentielle à facteur d'oubli adaptatif (la règle 1). Le nombre de périodes "optimal" est parfois aussi petit que deux : cela signifie que la régression qui détermine le pouvoir prédictif du passé sur la série de prix est effectuée sur les deux périodes précédentes. Cela provient probablement de la difficulté d'optimiser les règles d'analyse technique. Outre le risque d'obtenir un gain très élevé sur la série d'entraînement par rapport à la série de test, on fait face à de nombreux minima locaux qui rendent l'optimisation très difficile (on risque de ne pas trouver de minimum global). On peut également remarquer que les paramètres optimisés sont très variables d'une série d'entraînement à l'autre.

Expérience 2

Le deuxième type d'expérience ne fait pas l'optimisation des paramètres. Un ensemble de paramètres est généré aléatoirement pour chaque règle d'analyse technique. Les règles utilisées sont les mêmes que lors de l'expérience 1 à l'exception de la règle du support-résistance qui a été exclue. La raison en est la difficulté d'utiliser cette règle de façon complètement automatique et conséquemment les maigres résultats obtenus par cette règle. Une matrice de signaux est construite à partir des signaux produits par chaque règle et pour chaque valeur des paramètres générés. Le signal émis par une règle d'analyse technique est modifié : au lieu d'être compris dans l'ensemble $\{-1,0,1\}$ le signal est dans $\mathcal{R}$, ce qui donne en plus comme information la force du signal. Ensuite, on procède comme dans la première expérience : la matrice de signaux est fournie au réseau de neurones et la matrice de poids $\tilde{w}$ est optimisée par rapport à la fonction

de coût (espérance de la perte). Finalement, on obtient les signaux produits par le réseau de neurones.

L'expérience se déroule comme suit :

1. Un ensemble de paramètres aléatoires est généré pour chaque règle d'analyse technique.
2. Chaque règle produit une série de signaux pour chacun des paramètres générés précédemment qui correspond à une série de prix A.
3. L'ensemble de ces signaux forme une matrice qui est fournie en entrée au réseau de neurones. On peut alors obtenir la matrice $\tilde{w}$ en minimisant la fonction de coût sur la série A.
4. La performance du réseau de neurones et des règles d'analyse technique est testée sur la série B.

Les séries d'entraînement et de test sont les mêmes que celles utilisées lors de la première expérience. La stratégie correspondant à cette expérience est semblable à celle de la première expérience. Les contrats déjà échus sont utilisés comme série d'entraînement pour d'abord former une matrice de signaux (étapes 1 et 2 ci-haut) et déterminer les poids du réseau ($\tilde{w}$) par optimisation (étape 3). Les paramètres du système sont alors déterminés, il faut ensuite prédire une position optimale dans le marché. Pour ce faire, un vecteur de signaux est obtenue pour la nouvelle période en évaluant d'abord les règles d'analyse technique pour chacun des paramètres générés et le réseau peut produire son propre signal ce qui sera donc la position optimale prédite par le système pour la nouvelle période.

Les résultats de l'expérience 2 sont résumés dans le tableau 6. Dans la première colonne du tableau, tout comme pour le tableau 3, un "A" devant le nom de la série désigne une série d'entraînement et un "B" une série de test. Les quatre colonnes suivantes sont des statistiques sur les gains obtenus par les règles d'analyse technique : le gain maximum, le gain minimum, le gain moyen et l'écart-type sur le gain. Les deux dernières colonnes contiennent les gains obtenus par le réseau de neurones et la stratégie de "buy & hold", dénotés respectivement par "RN" et "B&H". Le réseau de neurones performe mieux que la stratégie de "buy & hold" 12 fois sur 21 sur les séries d'entraînement et 8 fois sur 21 sur les séries de test. Ces résultats sont similaires à ceux obtenus avec la première expérience. Ici, le réseau permforme mieux que le "buy & hold" deux fois de moins sur les séries d'entraînement et deux fois de plus sur les séries de test.

Dans le tableau 7, on peut comparer les gains annuels moyens pour le réseau de neurones dans les expériences 1 et 2 sur les séries de test versus le "buy & hold". Les années sont inscrites dans la première colonne, les résultats pour les réseaux de neurones de l'expérience 1 et 2 sont inscrits dans les deux colonnes suivantes ("RN1" et "RN2" respectivement) et dans la dernière colonne, on trouve les résultats pour le "buy & hold" ("B&H"). Les résultats sont variables, aucune stratégie n'est systématiquement meilleure qu'une autre. Toutefois, en faisant la moyenne sur toutes les années (dernière ligne du tableau 7), on constate que le réseau de neurones de l'expérience 1 performe mieux, en moyenne, que celui de l'expérience 2 et que le "buy & hold". Également, le réseau de neurones de l'expérience 2 performe moins bien en moyenne que le "buy & hold".

Conclusion

L'objectif du stage était de développer un système pour transiger automatiquement sur le marché des contrats à terme sur le taux d'intérêt. La technique employée combinait les règles d'analyse technique au moyen d'un réseau de neurones à une couche. La réussite de ce système dépend en partie de la structure sous-jacente de la série de prix considérée. Le système testé ici ne performe pas de façon fiable sur le marché des contrats à terme. Les règles d'analyse technique sont des techniques utilisées couramment pour transiger sur les marchés financiers. Cependant, la méthode proposée ne semble pas capable de combiner les règles d'analyse technique de façon efficace.

Il faudrait éventuellement utiliser plus de règles d'analyse technique différentes; celles qui ont été retenues ne sont pas nécessairement les plus performantes. On pourrait aussi explorer d'autres façons de combiner les signaux provenant des règles d'analyse technique.

	Moyenne	Écart-type	Maximum	Minimum
BAX 1993	94.6598	0.7147	95.86	92.97
BAX 1994	94.0433	1.1253	96.36	91.91
BAX 1995	92.8104	0.7697	94.23	91.06
BAX 1996	95.2709	0.7637	97.07	94.25
BAX 1997	96.2547	0.3106	96.90	95.66
BAX 1998	94.8799	0.2280	95.39	93.70
BAX 1999	94.9901	0.2319	95.56	94.44

TAB. 1 –. *Statistiques sur les BAX*

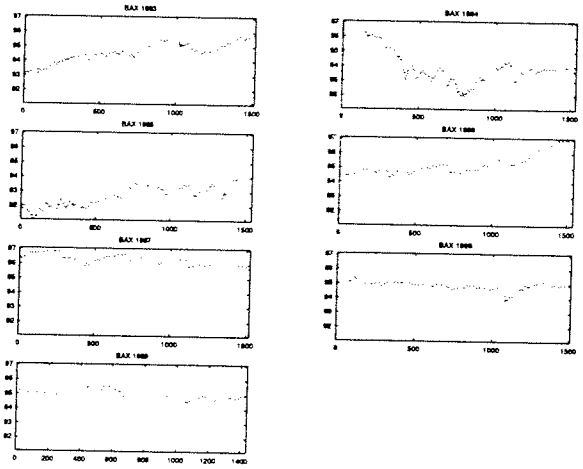

FIG. 1 –. *Contrats BAX de 1993 à 1999*

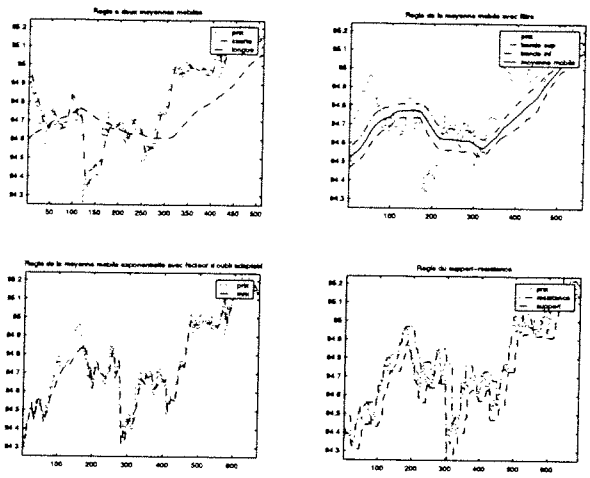

FIG. 2 –. *Illustration des quatre règles d'analyse technique utilisées*

A : entraînement	B : test
BAXH	BAXM
BAXH & BAXM	BAXU
BAXH & BAXM & BAXU	BAXZ

TAB. 2 –. *Séries utilisées pour l'entraînement et le test par année*

Série	1	2	3	4	RN	B & H
A : BAXH93	0.29	1.03	0.51	0.15	0.29	0.91
B : BAXM93	0.88	-0.34	0.42	-0.39	0.88	1.06
A : BAXH93 & BAXM93	1.10	1.30	1.03	-0.18	1.80	1.97
B : BAXU93	0.03	0.63	0.58	-0.70	0.43	0.77
A : BAXH93 & BAXM93 & BAXU93	0.74	2.20	1.95	-0.70	2.32	2.74
B : BAXZ93	0.32	0.58	0.16	-0.09	0.32	0.86
A : BAXH94	0.47	0.61	0.62	-0.01	0.47	-0.11
B : BAXM94	2.53	1.15	1.04	0.35	2.53	-1.91
A : BAXH94 & BAXM94	3.06	3.06	2.93	0.92	3.34	-2.02
B : BAXU94	1.55	1.53	1.84	0.88	0.65	1.35
A : BAXH94 & BAXM94 & BAXU94	5.55	5.17	5.36	1.80	5.29	-0.67
B : BAXZ94	0.64	-0.06	0.24	0.56	0.94	0.560
A : BAXH95	0.87	0.79	1.01	0.44	0.81	0.51
B : BAXM95	0.16	-0.08	0.94	-0.38	0.12	0.28
A : BAXH95 & BAXM95	2.25	1.75	2.60	0.58	2.23	0.79
B : BAXU95	0.36	0.76	1.62	0.11	1.08	0.18
A : BAXH95 & BAXM95 & BAXU95	3.67	0.77	5.09	1.17	3.85	0.97
B : BAXZ95	0.79	-2.73	1.66	0.78	0.77	0.79
A : BAXH96	0.84	0.48	0.76	-0.07	0.86	0.46
B : BAXM96	-0.02	0.58	0.39	0.34	0.20	0.32
A : BAXH96 & BAXM96	1.07	1.13	0.89	0.35	1.07	0.78
B : BAXU96	0.66	0.48	0.74	0.49	0.86	1.06
A : BAXH96 & BAXM96 & BAXU96	1.87	1.59	1.53	0.84	1.51	1.8
B : BAXZ96	1.32	1.20	0.29	-0.07	0.96	1.88
A : BAXH97	0.19	0.59	0.23	0.06	0.47	0.35
B : BAXM97	-0.01	0.73	0.21	-0.05	0.49	0.27
A : BAXH97 & BAXM97	0.80	1.64	1.53	0.01	0.80	0.62
B : BAXU97	0.20	0.66	-0.01	-0.01	0.20	0.10
A : BAXH97 & BAXM97 & BAXU97	0.57	1.99	1.23	0.15	0.57	0.72
B : BAXZ97	-0.57	-0.39	-0.22	0.03	-0.57	-0.13
A : BAXH98	0.96	-0.10	0.94	0.28	0.92	0.02
B : BAXM98	-0.03	-0.03	0.43	-0.02	0.01	0.15
A : BAXH98 & BAXM98	1.16	0.64	1.20	0.19	0.90	0.17
B : BAXU98	1.01	1.07	0.87	0.32	0.89	-0.87
A : BAXH98 & BAXM98 & BAXU98	2.18	1.34	2.02	0.63	2.08	-0.70
B : BAXZ98	0.00	0.08	0.17	0.14	0.12	1.02
A : BAXH99	0.31	0.43	0.31	-0.04	0.31	-0.31
B : BAXM99	-0.08	0	0.15	0.00	-0.01	0.44
A : BAXH99 & BAXM99	0.91	0.97	1.03	0.07	0.91	0.13
B : BAXU99	0.07	-0.37	-0.03	0.17	0.07	0.17
A : BAXH99 & BAXM99 & BAXU99	1.22	1.28	1.44	0.31	1.02	0.30
B : BAXZ99	-0.29	0.07	-0.22	-0.07	-0.19	0.35

TAB. 3 –. *Gains obtenus sur les séries d'entraînement et de test par chacune des règles pour chacune des années dans l'expérience 1*

	1	2	3	4	RN	B&H
1993 :	0.4100	0.2900	0.3867	-0.3933	0.5433	0.8967
1994 :	1.5733	0.8733	1.0400	0.5967	1.3733	0
1995 :	0.4367	-0.6833	1.4067	0.1700	0.6567	0.4167
1996 :	0.6533	0.7533	0.4733	0.2533	0.6733	1.0867
1997 :	-0.1267	0.3333	-0.0067	-0.0100	0.0400	0.0800
1998 :	0.3267	0.3733	0.4900	0.1467	0.3400	0.1000
1999 :	-0.1000	-0.1000	-0.0333	0.0333	-0.0433	0.3200
moyenne :	0.4533	0.2628	0.5367	0.1138	0.5119	0.4143

TAB. 4 –. *Gain moyen obtenu par chaque règle sur les ensembles de test par année*

	1	2		3		4
BAXH93 :	45	168	21	191	2.4	91
BAXH93 & BAXM93 :	42	122	71	7	0.2	63
BAXH93 & BAXM93 & BAXU93:	95	197	94	8	1.1	59
BAXH94 :	9	56	19	66	0.4	274
BAXH94 & BAXM94 :	103	112	1	31	0.6	5
BAXH94 & BAXM94 & BAXU94:	107	38	2	38	0.8	5
BAXH95 :	287	188	53	22	1.2	12
BAXH95 & BAXM95 :	10	31	2	17	0.3	89
BAXH95 & BAXM95 & BAXU95:	7	175	43	15	0.1	9
BAXH96 :	47	206	89	5	0.5	2
BAXH96 & BAXM96 :	42	201	95	5	0.9	4
BAXH96 & BAXM96 & BAXU96:	65	38	13	6	0.7	4
BAXH97 :	2	131	57	15	1.8	54
BAXH97 & BAXM97 :	35	46	38	79	1.2	81
BAXH97 & BAXM97 & BAXU97:	36	52	31	92	0.6	262
BAXH98 :	129	151	34	16	0.1	12
BAXH98 & BAXM98 :	54	80	2	16	1.6	3
BAXH98 & BAXM98 & BAXU98:	115	49	2	12	0.8	5
BAXH99 :	2	78	60	106	0.5	109
BAXH99 & BAXM99 :	74	199	15	54	0.8	45
BAXH99 & BAXM99 & BAXU99:	73	54	13	23	0.9	24

TAB. 5 –. *Paramètres de chaque règle obtenus après optimisation sur les ensembles d'entraînement*

Série	Max	Min	$\hat{\mu}$	$\hat{\sigma}$	RN	B&H
A : BAXH93	1.07	-1.39	-0.0551	0.4742	0.03	0.91
B : BAXM93	1.52	-1.88	0.1408	0.6412	1.08	1.06
A : BAXH93 & BAXM93	2.28	-2.10	-0.1431	0.7425	1.12	1.97
B : BAXU93	2.05	-2.23	0.5264	0.5616	1.71	0.77
A : BAXH93 & BAXM93 & BAXU93	2.38	-4.56	-0.2750	0.9200	-0.36	2.74
B : BAXZ93	1.66	-0.88	0.5822	0.3718	0.40	0.86
A : BAXH94	0.69	-0.89	0.1462	0.3216	0.55	-0.11
B : BAXM94	3.13	-1.91	1.5167	0.7238	1.95	-1.91
A : BAXH94 & BAXM94	3.74	-1.26	1.7943	0.8509	2.9	-2.02
B : BAXU94	3.15	-1.73	1.2220	0.7899	0.29	1.35
A : BAXH94 & BAXM94 & BAXU94	6.81	-3.13	3.0860	1.5413	3.03	-0.67
B : BAXZ94	1.78	-1.88	0.0305	0.6788	-0.28	0.56
A : BAXH95	1.71	-2.49	-0.2849	0.6783	0.17	0.51
B : BAXM95	1.74	-1.94	-0.1045	0.7729	0.02	0.28
A : BAXH95 & BAXM95	3.29	-3.93	-0.0724	1.3517	1.81	0.79
B : BAXU95	2.08	-1.96	0.3267	0.7884	1.34	0.18
A : BAXH95 & BAXM95 & BAXU95	5.53	-4.61	0.3744	1.8533	0.89	0.97
B : BAXZ95	2.77	-3.23	-0.3357	1.2225	-0.97	0.79
A : BAXH96	1.02	-0.48	0.2075	0.2227	0.50	0.46
B : BAXM96	0.90	-1.56	-0.3184	0.5633	-1.10	0.32
A : BAXH96 & BAXM96	1.27	-1.63	-0.1314	0.6095	-0.37	0.78
B : BAXU96	1.20	-0.20	0.6875	0.1983	0.88	1.06
A : BAXH96 & BAXM96 & BAXU96	2.35	-1.2100	0.5966	0.6409	0.83	1.84
B : BAXZ96	1.92	-0.52	1.1407	0.4217	1.46	1.88
A : BAXH97	0.61	-0.65	0.0284	0.2107	-0.15	0.35
B : BAXM97	1.23	-0.73	0.3106	0.5277	0.63	0.27
A : BAXH97 & BAXM97	1.96	-0.68	0.6139	0.6419	0.98	0.62
B : BAXU97	1.00	-0.67	0.1540	0.2409	-0.02	0.10
A : BAXH97 & BAXM97 & BAXU97	2.61	-1.21	0.2867	0.6773	0.09	0.72
B : BAXZ97	0.49	-0.97	-0.2021	0.2041	-0.09	-0.13
A : BAXH98	1.26	-1.12	0.0570	0.4980	0.30	0.02
B : BAXM98	0.77	-0.49	0.1426	0.2173	0.21	0.15
A : BAXH98 & BAXM98	1.52	-1.56	0.0476	0.6305	0.60	0.17
B : BAXU98	1.43	-2.19	0.3848	0.8139	0.85	-0.87
A : BAXH98 & BAXM98 & BAXU98	2.56	-3.08	0.2114	1.2341	1.15	-0.70
B : BAXZ98	1.40	-1.54	-0.1426	0.3637	-1.00	1.02
A : BAXH99	0.49	-0.41	0.0399	0.2356	0.31	-0.31
B : BAXM99	1.32	-0.60	0.3367	0.3429	0.22	0.44
A : BAXH99 & BAXM99	1.33	-0.24	0.5178	0.2999	0.93	0.13
B : BAXU99	1.17	-1.01	-0.1012	0.3335	-0.11	0.17
A : BAXH99 & BAXM99 & BAXU99	2.36	-0.78	0.6863	0.4609	1.16	0.30
B : BAXZ99	0.73	-1.39	-0.1264	0.3094	0.15	0.35

TAB. 6 –. *Gain obtenu sur les séries d'entraînement et de test pour l'expérience 2*

Année	RN1	RN2	B&H
1993 :	0.5433	1.0633	0.8967
1994 :	1.3733	0.6533	0
1995 :	0.6567	0.1300	0.4167
1996 :	0.6733	0.4133	1.0867
1997 :	0.0400	0.1733	0.0800
1998 :	0.3400	0.0200	0.1000
1999 :	-0.0433	0.0867	0.3200
moyenne :	0.5119	0.3629	0.4143

TAB. 7 –. *Gain moyen annuel du réseau de neurones versus le "buy & hold" sur les séries de test pour l'expérience 1 et 2*

CODE : MATLAB

Voici le code ayant servi à manipuler les données. Les contrats sont placés dans un tableau de cellules, ce qui permet de calculer les signaux des règles d'analyse technique pour chaque contrat puis de les assembler par "roll-over".

```
function y=contrats(nb_contrats,varargin)
% Crée un tableau de cellules contenant nb_contrats allant de premiereAnnee à
% derniereAnnee (ces arguments sont fournis dans 'varargin'}

nv=length(varargin);

if nv ==1
   premiereAnnee=varargin{1};
   nb_annees=1;
else
   derniereAnnee=varargin{2};
   premiereAnnee=varargin{1};
   nb_annees=derniereAnnee-premiereAnnee+1;
end

repertoire='BAX/BAX';
suffixe='_60.dat';

y=cell(nb_contrats,nb_annees);

switch nb_contrats
case{1}
   tipe='H';
case{2}
   tipe=char('H','M');
case{3}
   tipe=char('H','M','U');
otherwise
   tipe=char('H','M','U','Z');
end

for j=1:nb_annees
   for i=1:nb_contrats
      nom=strcat(repertoire,tipe(i),num2str(premiereAnnee+j-1),suffixe);
      y{i,j}=fscanf(fopen(nom),'%f');
   end
```

```
end
```

```
function serie=con2se(contrats,frequence)
% convertit des données présentées sous forme de tableau de cellules
% en une série chronologique
% 'frequence' est la fréquence d'échantillonage des données

% Données supplémentaires servant aux calculs des règles d'analyse
% technique permettant que les signaux commencent au début de l'année
%  (non disponible pour l'année 1992) :
% pour la fréquence de 5 min, on a laissé 4740 données
supMax=4740;
 % données supplémentaires pour la fréquence qui nous intéresse
supFrequence=supMax/(frequence/5);

[nb_contrats nb_annees]=size(contrats);
sep=zeros(nb_contrats+1,nb_annees);

for j=1:nb_annees
   for i=1:nb_contrats
      sep(i+1,j)=length(contrats{i,j});
   end
end

sep(1,find(sep(2,:)>supFrequence))=...
     supFrequence*ones(1,length(find(sep(2,:)>supFrequence)));

serie=[];
for j=1:nb_annees
   for i=1:nb_contrats
      serie=[serie;contrats{i,j}(sep(i,j)+1:end)];
   end
end
```

Voici le code permettant de calculer les signaux des règles d'analyse technique. Les signaux sont dans $\{-1,0,1\}$. Pour obtenir les signaux dans $\mathcal{R}$, il suffit de faire une petite modification indiquée en commentaire dans le code.

```
function z=kama(m,serie)
% moyenne mobile exponentielle avec facteur d oubli adaptatif
n=length(serie);
if m<=1, error('m <=1'); end
f=30;
s=2;
if n==0 | n<m, z=[]; return, end
y=zeros(n,1);
r2=zeros(n,1);
pente=zeros(n,1);
fastest=2/(f+1);
slowest=2/(s+1);
y(1:m)=serie(1:m);
```

```
for i=m+1:n
   % régression linéaire
   [r2(i),b(i)]=regress(i-m:i-1,serie(i-m:i-1));
   sc=(r2(i)*(fastest-slowest)+slowest)^2;
   y(i)=y(i-1)+sc*(serie(i-1)-y(i-1));
end

% calcul du signal
z=nan*zeros(n,1);
z(m+1:end)=sign(serie(m+1:end)-y(m+1:end));
% modification :
% z(m+1:end)=serie(m+1:end)-y(m+1:end);
```

```
function z=maMa3(param,serie)
% règle à deux moyennes mobiles.
% on suppose que la moyenne calculée de (x+1:x+l) sera comparée
% au prix en x+l+1
% i.e. on compare la moyenne avec l'observation qui suit les
% observations ayant servi au calcul de la moyenne
l=param(1);
c=param(2);
n=length(serie);

xl=0;
for i=1:l
   xl=xl+serie(i:n-l+i);
end
xl=xl/l;

xc=0;
for i=l-c+1:l
   xc=xc+serie(i:n-l+i);
end
xc=xc/c;

z=nan*zeros(n,1);
z(l+1:end)=sign(xc[1:end-1]-xl[1:end-1]);
% modification :
% z(l+1:end)=xc[1:end-1]-xl[1:end-1];
```

```
function z=mmFi(param,serie)
% moyenne mobile de période p avec un filtre f_t=x*écart-type(serie(t-p+1:t))
% règle de trading : position longue lorsque prix > mm+f
%                    fermeture de la position longue lorsque prix < mm
%                    position courte lorsque prix < mm -f
%                    fermeture de la position courte lorsque prix > mm
p=param(1);
x=param(2);
```

```
n=length(serie);
y=zeros(n,1);
f=zeros(n,1);
z=nan*zeros(n,1);
y(1:p)=serie(1:p);

for i=p+1:n
   y(i)=mean(serie(i-p:i-1));
   f(i)=x*std(serie(i-p:i-1));
   if y(i)+f(i)<serie(i)
      z(i)=1;
   elseif y(i)-f(i)>serie(i)
      z(i)=-1;
   elseif i>1 & z(i-1)==1 & serie(i)<y(i)
      z(i)=0;
   elseif i>1 & z(i-1)==-1 & serie(i)>y(i)
      z(i)=0;
   elseif i>1
      z(i)=z(i-1);
   end

end

% modification :
% for i=p:n
%    y(i)=mean(serie(i-p+1:i));
%    f(i)=x*std(serie(i-p+1:i));
%    if y(i)+f(i)<serie(i) | y(i)-f(i)>serie(i)
%       z(i)=serie(i)-(y(i)+f(i));
%    elseif i>1 & z(i-1)==1 & serie(i)<y(i)
%       z(i)=0;
%    elseif i>1 & z(i-1)==-1 & serie(i)>y(i)
%       z(i)=0;
%    elseif i>1
%       z(i)=z(i-1);
%    end
% end
```

```
function z=rdt_sr(w,serie)
% règle du support-résistance
% w : fenêtre dans laquelle est calculé le niveau de support et de résistance
n=length(serie);
A=serie(1:n-w)';
for i=2:w
   A=[A;serie(i:n-w+i-1)'];
end

maxSerie=max(A);
minSerie=min(A);

positionLongue=serie(w+1:n)>maxSerie';
```

```
positionCourte=serie(w+1:n)<minSerie';

% Pour calculer les signaux :
z=[nan*zeros(w,1);positionLongue-positionCourte];  %colonne
```

```
function gain=buyHold(contrats,frequence)
% calcule le gain pour une strategie de buy&hold

supMax=4740; % pour la fréquence de 5 min, on a laissé 4740 données
supFrequence=supMax/(frequence/5);
[nb_contrats nb_annees]=size(contrats);
sep=zeros(nb_contrats+1,nb_annees);

for j=1:nb_annees
   for i=1:nb_contrats
      sep(i+1,j)=length(contrats{i,j});
   end
end

sep(1,find(sep(2,:)>supFrequence))=...
      supFrequence*ones(1,length(find(sep(2,:)>supFrequence)));

gain=0;
for j=1:nb_annees
   for i=1:nb_contrats
gain=gain+contrats{i,j}(end)-contrats{i,j}(sep(i,j)+1);
   end
end
```

Voici le code permettant de manipuler les règles d'analyse technique.

```
% La fonction contratsMultiples renvoit le gain d'une règle d'analyse
% technique tandis que contratsMultiples_signal renvoit le signal de la
% règle mais les calculs sont essentiellement les mêmes.

% function gain_tot=contratsMultiples(param,contrats,frequence,regle,cout)
function signal=contratsMultiples_signal(param,contrats,frequence,regle,cout)

% les contrats doivent être rangés dans un tableau de cellules 'contrats'
% dont le nombre de lignes est le nombre de contrats et le nombre de colonnes
% le nombre d'annees
% 'règle' est le nom de la fonction faisant les calculs techniques;
% 'param' contient les différents paramètres nécessaires pour la
% règle
% 'cout' est le cout de transaction
% 'frequence' est la fréquence d'échantillonage des données

% Données supplémentaires servant aux calculs des règles d'analyse
% technique permettant que les signaux commencent au début de
% l'année (non disponible pour l'année 1992) :
% pour la fréquence de 5 min, on a laissé 4740 données :
```

```
supMax=4740;
% données supplémentaires pour la fréquence qui nous intéresse
supFrequence=4740/(frequence/5);

[nb_contrats nb_annees]=size(contrats);


z=cell(nb_contrats,nb_annees);
sep=zeros(nb_contrats+1,nb_annees);

for j=1:nb_annees
   for i=1:nb_contrats
      sep(i+1,j)=length(contrats{i,j});
      if ~isempty(contrats{i,j})
         z{i,j}=feval(regle,param,contrats{i,j}); %vecteur colonne
      end
   end
end

sep(1,find(sep(2,:)>supFrequence))=...
     supFrequence*ones(1,length(find(sep(2,:)>supFrequence)
));

serie=[];
signal=[];

for j=1:nb_annees
   for i=1:nb_contrats
      signal=[signal;z{i,j}(sep(i,j)+1:end)];
      serie=[serie;contrats{i,j}(sep(i,j)+1:end)];
   end
end

gain=signal(1:end-1).*diff(serie(1:end));
gain_tot=sum(gain)-cout*trade;



function [gainMax, paramMax]=optimContratsMultiples_v2(contrats,frequence,...
                                                regle,cout)
% optimisation des paramètres de 'regle', i.e. une règle d'analyse technique
% l'optimisation est faite sur sum(delta(prix)) pondérée par la position
% prise sur le marché.
% ici on crée aléatoirement les paramètres initiaux et
% on fait le tabou search
[nb_contrats nb_annees]=size(contrats);
serie=con2se(contrats,frequence);
n=length(serie);
nSup=395/(frequence/60);
maxIt=20;
nb_essai=20;
iMax=0;
```

```
% initialisation des valeurs de départ pour chaque essai
param=[];
switch regle
case{'mmFi'}
   nb_param=2;
   param=[round(100*rand(1,nb_essai));round(20*rand(1,nb_essai))];
   pas_init=ones(nb_param,2);
   borneSup=[nSup+1;inf];
   borneInf=zeros(nb_param,1);
case{'maMa3'}
   nb_param=2;
   for i=1:nb_param
      param=[round(100*i*rand(1,nb_essai));param];
   end
   pas_init=ones(nb_param,2);
   borneSup=(nSup+1)*ones(nb_param,1);
   borneInf=zeros(nb_param,1);
case{'rdt_sr','kama'}
   nb_param=1;
   param=round(100*rand(1,nb_essai));
   pas_init=ones(nb_param,2);
   borneSup=nSup;
   borneInf=1;
otherwise
   error('regle non permise');
end


if nargin ==3
   cout=0;
elseif nargin<3 | nargin>4
   error('nombre d arguments incongru');
end

% valeurs du maximum local et de ses paramètres trouvées après
% chaque essai et chaque itération
gainMaxIt=-inf*ones(1,nb_essai);
paramMaxIt=nan*zeros(nb_param,nb_essai);
gainMax=-Inf;
paramMax=nan*zeros(nb_param,1);

% points déjà visités par l'algorithme
tabou=zeros(nb_param,nb_essai*maxIt);

% compteur pour le tableau 'tabou'
tab=1;


for k=1:nb_essai
   pas=pas_init;
   param0=param(:,k);
   if strcmp(regle,'maMa3')
      if param0(1)<param0(2)
         temp=param0(1);
```

```
      param0(1)=param0(2);
      param0(2)=temp;
   elseif param0(1)==param0(2)
      param0(1)=param0(1)+1;
   end
   if param0(2)<=borneInf(2)
      param0(2)=borneInf(2)+1;
   end
end
% vérification si le paramètre a déjà été visité
test=[];
for i=1:nb_param
   test=[test;isempty(find(param0(i)==tabou(i,:)))];
end
if ~isempty(find(test==1)) & param0<borneSup & param0>borneInf
   tabou(:,tab)=param0;
   tab=tab+1;
   % évaluation de la fonction aux paramètre param0
   if strcmp(regle,'mmFi')
      gain0=contratsMultiples([param0(1),param0(2)/10],...
                  contrats,frequence,regle,cout);
   else
      gain0=contratsMultiples(param0,contrats,frequence,...
                  regle,cout);
   end
   iMaxOld=0;

   for j=1:maxIt
      if strcmp(regle,'maMa3')
         borneSup(2)=param0(1);
         borneInf(1)=param0(2);
      end
      gain=nan*ones(nb_param*2,1);
      param_voisins=param0*ones(1,nb_param*2);
      % Évaluation de la fonction aux paramètres voisins
      for i=1:nb_param
         if param0(i)+pas(i,1) <borneSup(i) & (iMaxOld==0 | iMax==i+i-1)
            param_voisins(i,i+i-1)=param0(i)+pas(i,1);
            if strcmp(regle,'mmFi')
               gain(i+i-1)=contratsMultiples([param_voisins(1,i+i-1),...
                              param_voisins(2,i+i-1)/10],...
                               contrats,frequence,regle,cout);
            else
               gain(i+i-1)=contratsMultiples(param_voisins(:,i+i-1),...
                                 contrats,frequence,regle,cout);
            end
         elseif ceil(pas(i,1)/2)~=pas(i) & param0(1)+ceil(pas(i,1)/2)<...
                                                      borneSup(i)
            pas(i,1)=ceil(pas(i,1)/2);
            param_voisins(i,i+i-1)=param0(i)+pas(i,1);
            if strcmp(regle,'mmFi')
               gain(i+i-1)=contratsMultiples([param_voisins(1,i+i-1),...
                     param_voisins(2,i+i-1)/10],...
                      contrats,frequence,regle,cout);
```

```
        else
           gain(i+i-1)=contratsMultiples(param_voisins(:,i+i-1),...
                 contrats,frequence,regle,cout);
        end
     end

     if param0(i)-pas(i,2)>borneInf(i) & (iMaxOld==0 | iMax==i+i)
        param_voisins(i,i+i)=param0(i)-pas(i,2);
        if strcmp(regle,'mmFi')
           gain(i+i)=contratsMultiples([param_voisins(1,i+i),...
                 param_voisins(2,i+i)/10],...
                  contrats,frequence,regle,cout);
        else
           gain(i+i)=contratsMultiples(param_voisins(:,i+i),...
                   contrats,frequence,regle,cout);
        end
     elseif ceil(pas(i,2)/2)~=pas(i,2) & param0(i)-ceil(pas(i,2)/2)...
                                                  >borneInf(i)
        pas(i,2)=ceil(pas(i,2)/2);
        param_voisins(i,i+i)=param0(i)-pas(i,2);
        if strcmp(regle,'mmFi')
           gain(i+i)=contratsMultiples([param_voisins(1,i+i),...
               param_voisins(2,i+i)/10],...
                contrats,frequence,regle,cout);
        else
           gain(i+i)=contratsMultiples(param_voisins(:,i+i),...
                 contrats,frequence,regle,cout);
        end
     end
  end

  % Détermination du voisin optimal
  [gainMax,iMax]=max(gain);

  %Ajout des nouveaux points visités dans tabou
  for i=1:nb_param*2
     tabou(:,tab)=param_voisins(:,i);
     tab=tab+1;
  end

  if gainMax >gain0
     paramMax=param_voisins(:,iMax);
     verif=[];
     for i=1:nb_param
        verif=[verif;isempty(find(paramMax(i)==...
                 tabou(i,1:tab-2*nb_param-1)))];
     end
     if isempty(find(verif==1))
        break
     end
     gain0=gainMax;
                 param0=paramMax;
     iMaxOld=iMax;
         elseif isnan(gainMax)
```

```
            gainMax=gain0;
            paramMax=param0;
            break
         elseif 2*pas(:,1) < borneSup/2
            pas=2*pas;
            gainMax=gain0;
            paramMax=param0;
         else
            gainMax=gain0;
            paramMax=param0;
            break
         end
      end %boucle for j
      gainMaxIt(k)=gainMax;
      paramMaxIt(:,k)=paramMax;
   end %if isempty
end %boucle for k

[gainMax iMax]=max(gainMaxIt);
if strcmp(regle,'mmFi')
   paramMax=[paramMaxIt(1,iMax),paramMaxIt(2,iMax)/10];
else
   paramMax=paramMaxIt(:,iMax);
end
```

```
function signal=signaux(param_cell,contrats,frequence,cout,regle)
% calcul les signaux des règles sur les contrats
% param_cell doit être un tableau de cellules de dimensions nx1 où
%  n est le nombre de règle d analyse technique considéré
if nargin==3
   cout=0;
   regle={'kama','maMa3','mmFi','rdt_sr'};
elseif nargin==4
   regle={'kama','maMa3','mmFi','rdt_sr'};
end
nb_param_cell=length(param_cell);
z=cell(nb_param_cell,1);
depart=ones(nb_param_cell,1);


for i=1:nb_param_cell
   z{i}=contratsMultiples_signal(param_cell{i},contrats,...
                frequence,char(regle(i)),cout);
   indice=find(~isnan(z{i}));
   depart(i)=indice(1);
end
depart_final=max(depart);

signal=[];
for i=1:nb_param_cell
   signal=[signal;z{i}'];
```

```
end
```

Voici le code ayant servi aux calculs du réseau de neurones.

```
function y=L(a,contrats,frequence,t,cout)
% fonction de perte pour les signaux de trading sur "contrats" au temps t
% "contrats" est un tableau de cellules tel que fournit par la fonction
% contrats.m du répertoire (parent d:\Julie)
% a représente la position à prendre;
% les valeurs permises pour a sont {1,0,-1}
% a peut être un vecteur colonne,
% dans ce cas-là t doit être un vecteur de même longueur
% y est de la même dimension que a
serie=con2se(contrats,frequence);
if nargin==3
   cout=0;
end
c=2;
std_diff=std(diff(serie));
y=cout*abs(a)-a.*((serie(t)-serie(t-1))*ones(1,size(a,2)));
```

```
function p=classifier(w,x)
% calcule la probabilité que chaque action soit optimale (sortie su réseau)
% w est une matrice de poids 3*(n+1) incluant les biais
% w(k,:)=[w_{k,0},w_{k,1},w_{k,2},...,w_{k,n}], k=1,2,3
% x représente les signaux provenenant des règles d analyse technique
% size(x,1) : nombre de signaux
% size(x,2) : longueur de la série sur laquelle s applique les signaux
p=softmax(w*[ones(1,size(x,2));x]);
```

```
function a=softmax(n)
if nargin < 1, error('Not enough arguments.'); end
%expn=exp(n);
%a=expn/sum(expn,1);

[s,q]=size(n);
expn=exp(n);
denom=1./sum(expn,1);
a=expn.*denom(ones(1,s),:);
```

```
function [y,g]=fperte_2(w,x,contrats,frequence,cout)
% fonction de perte à minimiser pour identifier la matrice de poids w
% le nombre de colonnes de x doit être le même que la longueur de la
% série obtenue à partir de 'contrats', et les lignes de x
% représente les différents signaux
```

```
% calcule la fonction objective et le gradient
if nargin==3
   cout=0;
end
T=size(x,2);
if T==1
x=[ones(size(x)),x];
T=2;
end
t=2:T;
a=[ones(T-1,1),zeros(T-1,1),-ones(T-1,1)];
perte=L(a,contrats,frequence,t,cout);
proba=classifier(w,x(:,2:T));
A=proba.*perte';
y=sum(sum(A));

%calcul du gradient :
if nargout>1
   x_mod=[ones(1,size(x,2));x];
   g=nan*ones(3,size(x_mod,1));
   B=(proba.^2).*perte';
   for j=1:size(x_mod,1)
      C=(ones(3,1)*x_mod(j,2:T)).*(A-ones(3,1)*sum(B));
      g(:,j)=sum(C')';
   end
end
```

```
function gain=combine(p,contrats,frequence,cout)
% on veut évaluer la performance de la classification de signaux obtenue;
% calcule le gain obtenu par le réseau de neurones
% p=classifier(w,x); w est la matrice de poids et x celle des signaux
% on suppose que la longueur de p est le même que
% celle de la série obtenue de 'contrats'
serie=con2se(contrats,frequence);
if nargin==2
   cout=0;
end
[a j]=max(p);
z=zeros(length(serie),1);
z(find(j==1))=1;
z(find(j==2))=0;
z(find(j==3))=-1;

trade=length(find(diff(z)));
% ajout de la première transaction
if z(1)~=0
   trade=trade+1;
end
% s'il y a au moins une transaction, alors on clot la dernière transaction
if trade~=0 & z(end)~=0
   trade=trade+1;
```

```
end
gain=sum(z(1:end-1).*diff(serie))-cout*trade;
```

Voici un exemple de code employé pour faire les simulations dans le cas où les paramètres des règles d'analyse technique sont obtenus par optimisation.

```
t0=clock;
w=cell(7,3);
param_test=cell(7,3);
gainIn=cell(7,3,3);
gainOut=cell(7,3,3);
suivant=cell(1,1);
supFrequence=4740/12;

for i=0:6
for j=1:3
% ensemble d'entraînement :
y=contrats(j,93+i);

% ensemble de test :
s=contrats(j+1,93+i);
suivant{1}=s{j+1}(length(s{j})-supFrequence+1:end);

% Optimisation des règles d'analyse technique :
[gain_kama param_kama]=optimContratsMultiples_v2(y,60,'kama',0);
[gain_maMa3 param_maMa3]=optimContratsMultiples_v2(y,60,'maMa3',0);
[gain_mmFi param_mmFi]=optimContratsMultiples_v2(y,60,'mmFi',0);
[gain_sr param_sr]=optimContratsMultiples_v2(y,60,'rdt_sr',0);
param_test{i+1,j}=creerParam(param_kama,param_maMa3,param_mmFi,param_sr);

% Calcul des signaux des règles d'anlyse technique
% pour les paramètres optimisés :
signal=signaux(param_test{i+1,j},y,60);

% optimisation : determination des w :
options=optimset('GradObj','on');
[w{i+1,j},fval,exitflag,ouput,grad]=fminunc('fperte_2',zeros(3,5),options,signal,y,60,0);

% Transformation des tableaux de cellules en vecteurs :
serie=con2se(y,60);
suivant_serie=con2se(suivant,60);

% Calcul du gain dse règles d'analyse technique in-sample :
gain_technique=zeros(4,1);
for k=1:4
gain_technique(k)=sum(signal(k,1:end-1)'.*diff(serie(1:end)));
end

% Calcul du gain du réseau de neurones in-sample :
p=classifier(w{i+1,j},signal);
gain=combine(p,y,60,0);

% Calcul du gain du ''buy and hold'' in-sample :
gain_BH=buyHold(y,60);
```

```
gainIn{i+1,j,1}=gain_technique;
gainIn{i+1,j,2}=gain;
gainIn{i+1,j,3}=gain_BH;

% Gain réalisé hors-echantillon :
% Signaux d'analyse technique hors-échantillon :
signal_suivant=signaux(param_test{i+1,j},suivant,60);

% Calcul du gain dse règles d'analyse technique :
gain_technique_suivant=zeros(4,1);
for k=1:4
gain_technique_suivant(k)=sum(signal_suivant(k,1:end-1)'.*diff(suivant_serie(1:end)));
end

% Calcul du gain du réseau de neurones :
p_suivant=classifier(w{i+1,j},signal_suivant);
gain_suivant=combine(p_suivant,suivant,60,0);

% Gain fait par le buy & hold
gain_BH_suivant=buyHold(suivant,60);

gainOut{i+1,j,1}=gain_technique_suivant;
gainOut{i+1,j,2}=gain_suivant;
gainOut{i+1,j,3}=gain_BH_suivant;

end
end

temps=etime(clock,t0);

save mesResultatsVieux gainIn gainOut param_test w temps
```