Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Prevendo a Variação Intraday do Ibovespa com Preditores Lineares

Autor:

Diogo Azevedo Barros

Orientador:

Prof. Luiz Calôba, D. Ing

Examinador:

Prof. Mariane Petraglia, P.hd

Examinador:

Prof. Edson Gonçalves, D.Sc

DEL

Setembro de 2009

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

DEDICATÓRIA

Dedico este trabalho a todos que me ajudaram antes, durante e após minha graduação. Devo muito à UFRJ e a todos que me ajudaram e espero retribuir da mesma maneira. Em especial dedico aos meus pais que me incentivaram sempre, meu irmão que sempre me ajudou e ao Quemel e ao Bomba, que me mostraram que nós mesmos criamos nossos obstáculos e somente nós podemos removê-los.

AGRADECIMENTO

Agradeço ao professor Calôba pela atenção e dedicação durante todo o trabalho. Agradeço à RiskControl por ter confiado no meu trabalho e sempre me ajudado nos estudos. Agradeço a todos os colegas da graduação por todos os anos de estudo e amizade. Agradeço ainda à UFRJ, por ter me dado um ensino de excelente qualidade que espero retribuir ao país.

RESUMO

Este trabalho propõe uma técnica para prever a pontuação de fechamento diária do Ibovespa assim como a tendência do seu gráfico diário utilizando redes neurais. Utilizamos dados de mercado diários do Ibovespa provenientes da Bloomberg e montamos a rede neural utilizando o módulo de redes neurais do Matlab. Definimos variáveis de entrada que mantém a significância das correlações durante o tempo e estudamos os dias que o modelo pode ser aplicado baseado em premissas de significância estabelecidas. Utilizamos uma série para testar a rede neural e simulamos estratégias de compra e venda do Ibovespa, obtendo retornos significativos.

Palavras-Chave: redes neurais, predição linear, Ibovespa, tendência, Matlab.

ABSTRACT

This paper proposes a technique to predict the Ibovespa daily closing quote as well as the tendency of its daily graphic using neural networks. We used Ibovespa daily market data from Bloomberg and analyzed the neural network using Matlab's neural network toolbox. We defined input variables that keep the correlations significance over time and analyzed the days which the model can be applied based on the same significance assumptions. We used a data set to test the neural network and simulated strategies of buying and selling the Ibovespa index, with significative returns.

Key-words: neural network, linear prediction, Ibovespa, tendency, closing quote.

SIGLAS

UFRJ – Universidade Federal do Rio de Janeiro

Ibovespa – Indice da Bolsa de Valores do Estado de São Paulo

NYSE – New York Stock Exchange

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Tema

A capacidade de prever eventos futuros sempre foi buscada ao longo da história, e na área econômica é objeto de estudo constante. A capacidade de prever o comportamento do principal índice de ações do Brasil (Ibovespa) é garantia de altos retornos financeiros e de redução dos riscos de investimento.

Este trabalho aplica técnicas de redes neurais para prever o fechamento do Ibovespa e seus extremos para o dia, seja de alta ou baixa e sua ordem de ocorrência. Estas informações permitem a montagem de estratégias de operações direcionais sobre o índice Ibovespa que reduzem os riscos nas aplicações financeiras e aumentam o retorno sobre os investimentos.

1.2 – Delimitação

Este estudo utiliza dados diários de fechamento de mercado do Ibovespa durante o período de Novembro de 2008 à Março de 2009 para montar e treinar uma rede neural que é simulada sobre outras janelas de dados. O modelo utilizado segue a premissa simplificadora que condiciona a predição do Ibovespa apenas à influência dos seus valores passados, não considerando a influência de outros fatores como o fechamento de outras bolsas no mundo (exceto bolsa de Nova Iorque), fatores político-monetários, fatores macroeconômicos externos, etc..

A simulação das operações no mercado utiliza por simplificação um produto financeiro fictício, o mercado à vista do índice Ibovespa. Este índice é negociado apenas na Bolsa de Mercadorias e Futuros (BM&F) sob a forma de um contrato futuro de índice, contudo para fins acadêmicos é possível inferir que os ganhos no mercado à vista fictício do Ibovespa geram ganhos no mercado futuro do índice negociado na

BM&F, tornando a estratégia de negociação do índice aplicável no mercado brasileiro e gerando retornos favoráveis a seus aplicadores.

1.3 – Justificativa

São negociados por ano uma média 20 milhões de contratos de futuro de Ibovespa na BM&F. A capacidade de prever o comportamento do Ibovespa permite que o investidor reduza seu risco nas aplicações e aumente seu retorno ao negociar contratos futuros de Ibovespa ou ações que possuem correlação alta com o índice, como os papéis da Vale e Petrobras.

1.4 – Objetivos

O objetivo deste trabalho é mostrar que a tendência do comportamento do índice de ações Ibovespa pode ser prevista utilizando-se um aproximador linear implementado por rede neural com função de ativação linear, através de um estudo cuidadoso sobre os dados de entrada desta rede.

Este trabalho mostra ainda que esta rede após ser treinada com uma janela de dados limitada no tempo, pode ser aplicada à outras janelas de dados e ainda manter retornos elevados sobre as aplicações financeiras resultantes das estratégias montadas à partir dos resultados da rede.

1.5 – Metodologia

Para garantir a qualidade e consistência dos resultados do trabalho, utilizamos dados de mercado provenientes da fonte de dados privados Bloomberg. Foram obtidos dados intradiários do índice Ibovespa com amostras a cada 30 minutos, em uma janela de 4 meses aproximadamente.

Foram utilizadas as ferramentas de análise estatística e de redes neurais do Matlab para realizar as análises dos dados. Para facilitar o entendimento dos resultados finais, foram gerados 2 programas com interface no Matlab utilizando seu módulo de

programação de interfaces (GUI). Estes programas mostram os resultados tratamentos realizados sobre os dados para a escolha dos dias operáveis do modelo.

1.6 – Descrição

No capítulo 2 são mostradas as etapas de tratamento dos dados de entrada da rede e o processo de escolha dos dias operáveis segundo as premissas do modelo. São ilustradas ainda as interfaces gráficas geradas no Matlab para visualização dos resultados que definiram a escolha dos dias operáveis pelo modelo.

No capítulo 3 é mostrada a montagem da rede neural e o seu treinamento utilizando os dias operáveis do modelo para reduzir o erro da rede. Ao final é possível verificar os resultados previstos contra os realizados na interface gráfica do Matlab.

No capítulo 4 é realizada uma nova simulação da rede sobre dados fora da amostra inicial. São montadas as estratégias de compra ou venda do Ibovespa utilizando os resultados previstos pela rede neural. Os resultados financeiros obtidos com as estratégias são aferidos e ilustrados para mostrar a vantagem de utilizar o modelo.

A conclusão mostra que a rede neural previu o comportamento do índice com precisão suficiente para gerar retornos favoráveis para operações com contratos à vista do Ibovespa.

Capítulo 2

Dados de Entrada

2.1 – Colhendo os dados

Para prever o comportamento do Ibovespa utilizando uma rede neural com função de ativação linear precisamos de uma série histórica de pontuação do índice suficientemente grande para que, após a eliminação dos dias não operáveis segundo as premissas do modelo, tivéssemos ainda uma quantidade de dias operáveis suficientes para realizar o treinamento da rede.

Utilizamos uma janela de 93 dias úteis entre o Outubro/08 e Março/09, colhendo a cotação do índice a cada 30 minutos para obtermos a pontuação de fechamento, a pontuação máxima e mínima do dia assim como a ordem em que a máxima e a mínima ocorreram. Se a pontuação máxima foi atingida após a mínima, significa que a bolsa teve uma tendência de alta, enquanto que o contrário resulta em uma tendência de baixa.

Figura 2.1.1- Gráfico intradiário do Ibovespa

Fonte: Bloomberg

Após a aferição da tendência diária sobre as amostras intradiárias, cada variável do modelo continha 93 amostras diárias entre os dias 31/Out/08 e 19/Mar/09. A partir destes dados de entrada, calculamos o retorno logarítimico de cada variável do modelo em relação à abertura do dia através da formula:

$$r_{\log} = \ln\left(\frac{V_f}{V_i}\right)$$

Logo, para calcular o retorno das variáveis de entrada do preditor em relação à abertura do dia:

$$r_{entrada} = \ln\left(\frac{V_f}{V_i}\right)$$

onde,

V_f = Abertura(d)

V_i = {Abertura(d-1); Mínima(d-1); Máxima(d-1); Fechamento(d-1)}

Para calcular o retorono das variáveis de saída do preditor em relação à abertura do dia:

$$r_{saída} = \ln\left(\frac{V_f}{V_i}\right)$$

onde,

V_f = {Mínima(d); Máxima(d); Fechamento(d)}

V_i = Abertura(d)

Às variáveis de entrada, ordem(d-1), e saída, ordem(d), foram atribuídos os valores 1 se a tendência for de alta (mínima→máxima) e -1 se a tendência for de baixa (máxima→mínima).

2.2 – Tratando as variáveis

Após obter a série de retorno logaritmico em relação à abertura de cada variável do modelo, foi necessário realizar tratamentos sobre as séries para tentar reduzir influências externas ao Ibovespa e atenuar as séries

Para entender as influências externas sobre o Ibovespa consideramos, baseados em outros trabalhos, que o índice da bolsa de valores de Nova Iorque(Dow Jones Industrial Average), apresentava correlação alta com o Ibovespa. Devido ao tamanho das economias norte americana e brasileira, é de se esperar que o comportamento da bolsa americana influencia a bolsa brasileira, e não o contrário. Portanto, para reduzir esta influência sobre nossa série, colhemos da Bloomberg a volatilidade histórica dos índices Ibovespa e Dow Jones para a janela de análise do nosso modelo.

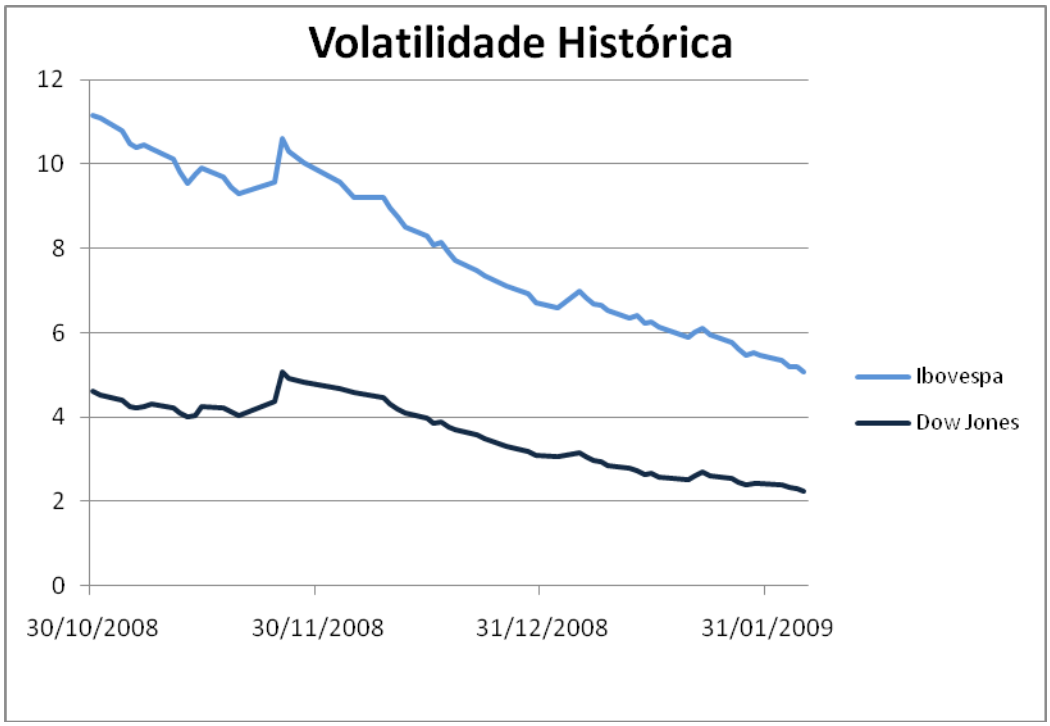

Figura 2.2.1- Volatilidade do Ibovespa x Dow Jones

Fonte: Bloomberg

Podemos observar que um súbito aumento na volatilidade do Dow Jones em Novembro causa um pico ainda maior de volatilidade no Ibovespa. Para eliminar esta influência externa da nossa série, que poderia prejudicar a estimativa da rede neural, excluímos os dias (24/11 à 28/11) que causavam o aumento da volatilidade do índice americano e, por consequência, o pico de volatilidade do Ibovespa.

A eliminação destes dias resultou no gráfico de volatilidades abaixo, que não contém mais picos de volatilidade na bolsa brasileira por influência da bolsa americana.

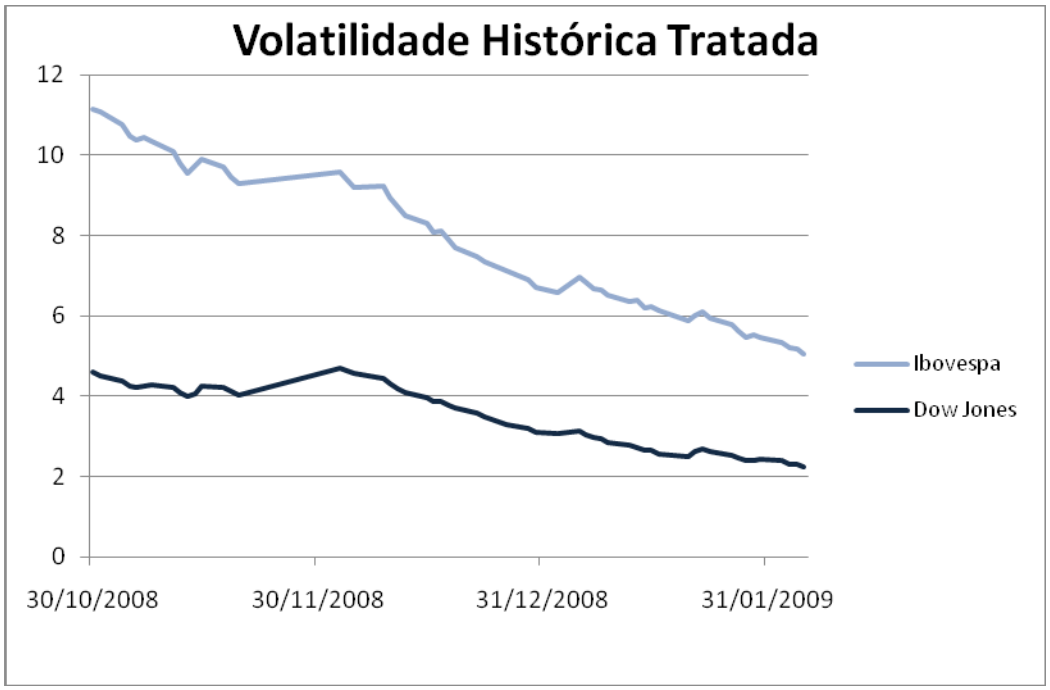

Figura 2.2.2- Volatilidade do Ibovespa x Dow Jones após eliminação dos dias de maior volatilidade da bolsa americana.

Fonte: Bloomberg

Após eliminar o período de maior influência da bolsa americana sobre a volatilidade do Ibovespa, foi possível obter a matriz de correlações entre as variáveis de entrada e saída abaixo:

Matriz de Correlações				
Entradas/Saídas	Máxima(d)	Mínima(d)	Fechamento (d)	Ordem(d)
Abertura(d-1)	-8%	4%	-6%	-19%
Fechamento(d-1)	34%	39%	31%	2%
Máxima(d-1)	-5%	14%	-4%	-18%
Mínima(d-1)	-5%	-2%	-9%	-23%
Ordem(d-1)	14%	12%	7%	-6%

Tabela 2.2.1- Matriz de correlações entre as variáveis de entrada e saída

Baseados na matriz de correlação, foi possível determinar quais os pares entrada/saída seriam escolhidos para o modelo, uma vez que as variáveis de entrada

explicarão o comportamento das variaveis de saída do nosso modelo. Escolhemos os pares que apresentavam correlação elevada, totalizando 10 pares entrada saída abaixo:

Entradas	Saída
Abertura(d-1)	Ordem(d)
Máxima(d-1)	
Ordem(d-1)	
Abertura(d-1)	Máxima(d)
Fechamento(d-1)	
Ordem(d-1)	
Fechamento(d-1)	Mínima(d)
Ordem(d-1)	
Fechamento(d-1)	Fechamento(d)
Ordem(d-1)	

Tabela 2.2.2 - Pares Entrada/Saída

Afim de estudar o comportamento das variáveis no tempo, *plotamos* gráficos que apresentam o produto entrada*saída ao longo do tempo, isto é uma espécie de "covariância instantânea" entre as variáveis. Utilizamos o algoritmo abaixo para para suavizar o sinal entrada*saída, limitando sua amplitude à uma variável K, determinada visualmente à partir de observações sobre os gráficos dos 10 pares entrada/saída.

$$S_{suavizado} = K * \tanh\left(\frac{S_{original}}{K}\right)$$

Abaixo é possível observar os gráficos originais e suavizados de cada par entrada*saída no tempo.

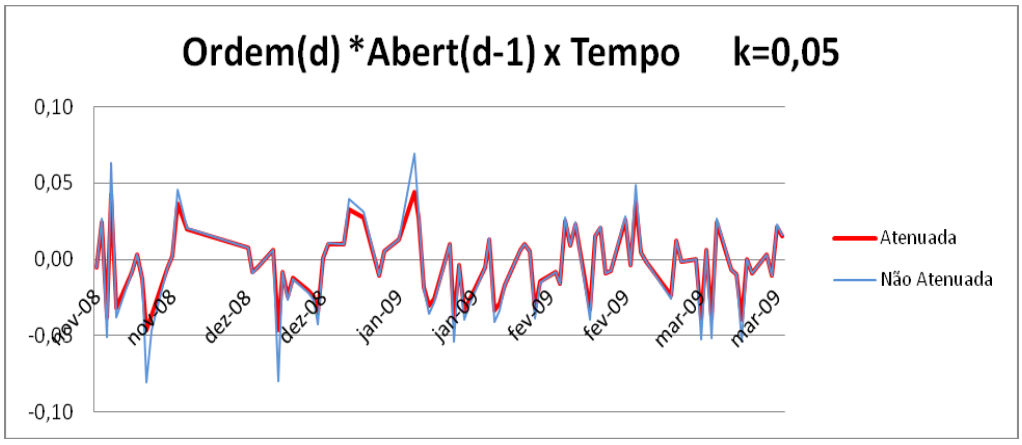

Figura 2.2.3- Gráfico Ordem(d)*Abertura(d-1) x Tempo, k = 0,05

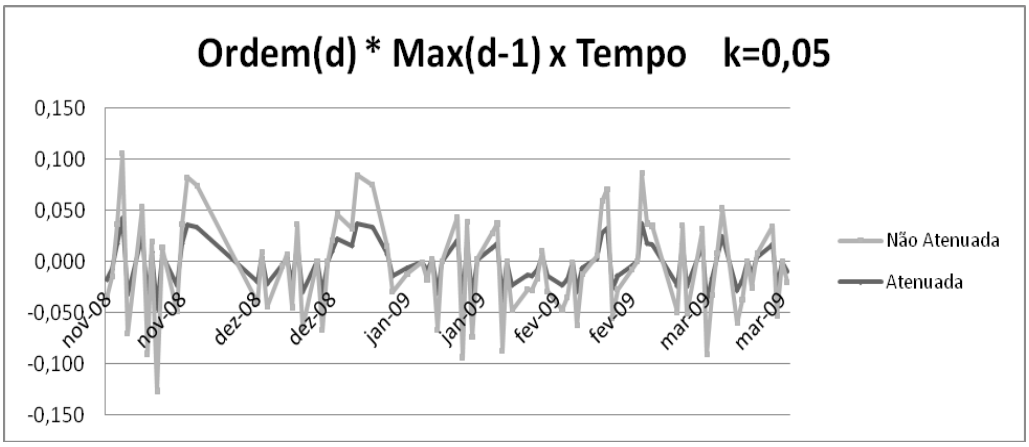

Figura 2.2.4- Gráfico Ordem(d)*Máxima(d-1) x Tempo, k=0,05

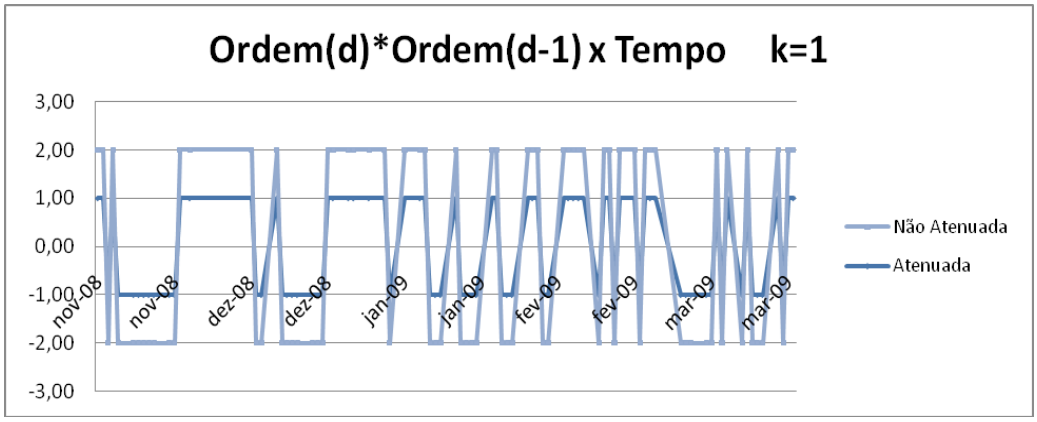

Figura 2.2.5- Gráfico Ordem(d)*Ordem(d-1) x Tempo, k=1

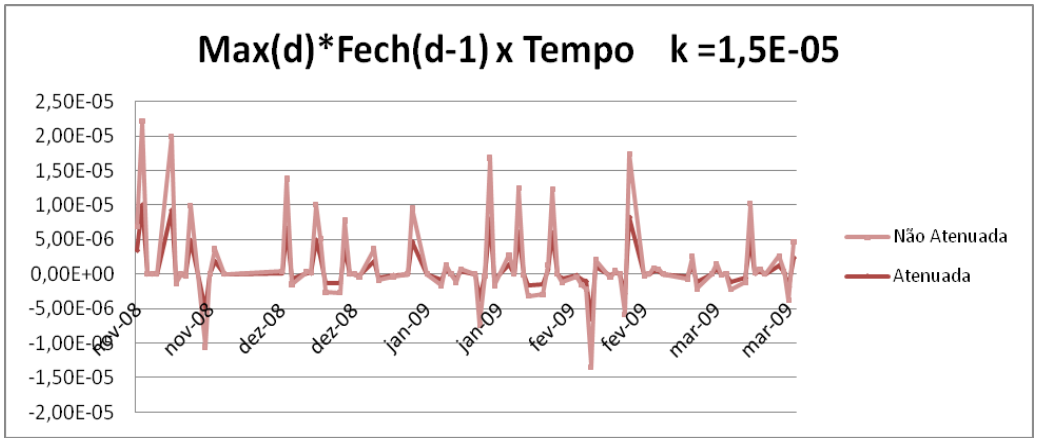

Figura 2.2.6-Gráfico Máxima(d)*Fechamento(d-1) x Tempo, k=1,5E-05

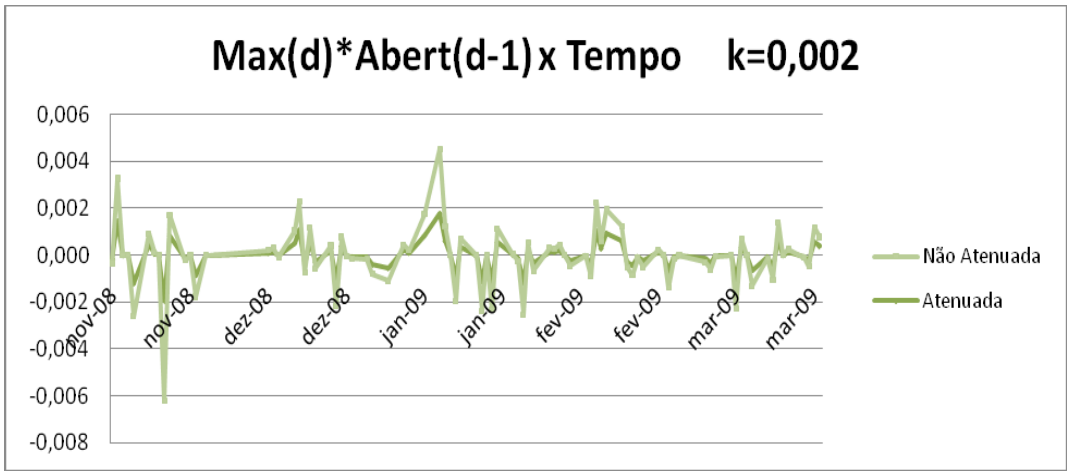

Figura 2.2.7- Máxima(d)*Abertura(d-1) x Tempo, k=0,002

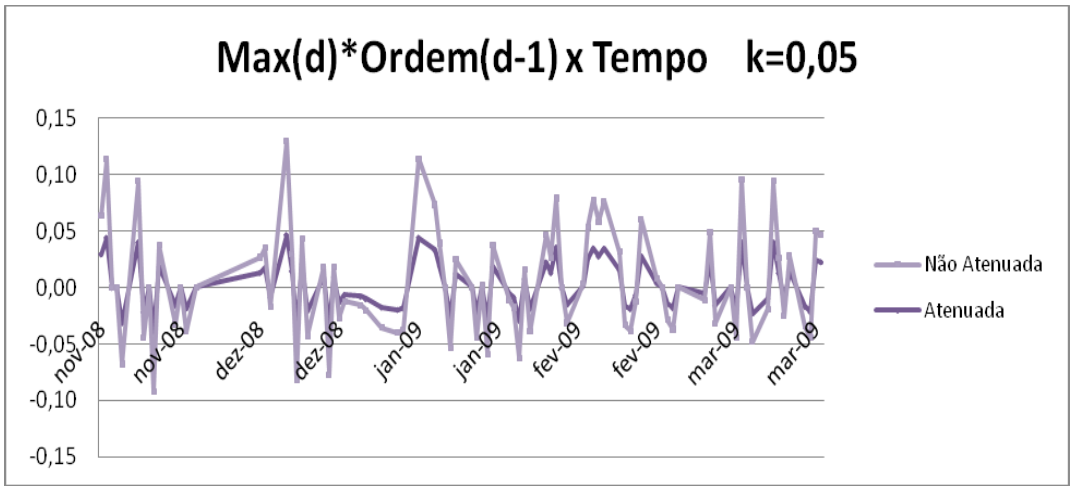

Figura 2.2.8- Máxima(d)*Ordem(d-1) x Tempo, k=0,05

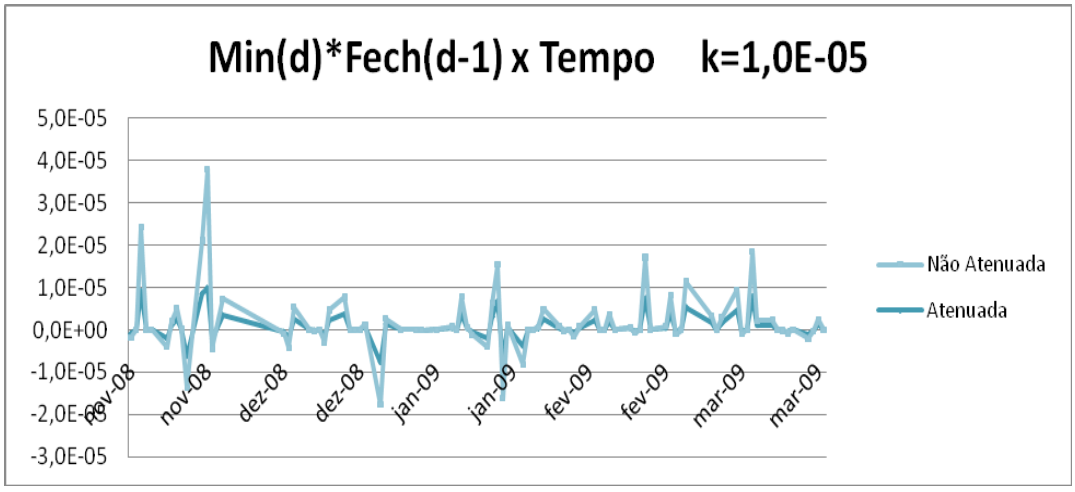

Figura 2.2.9- Mínima(d)*Fechamento(d-1) x Tempo, k=1,0E-05

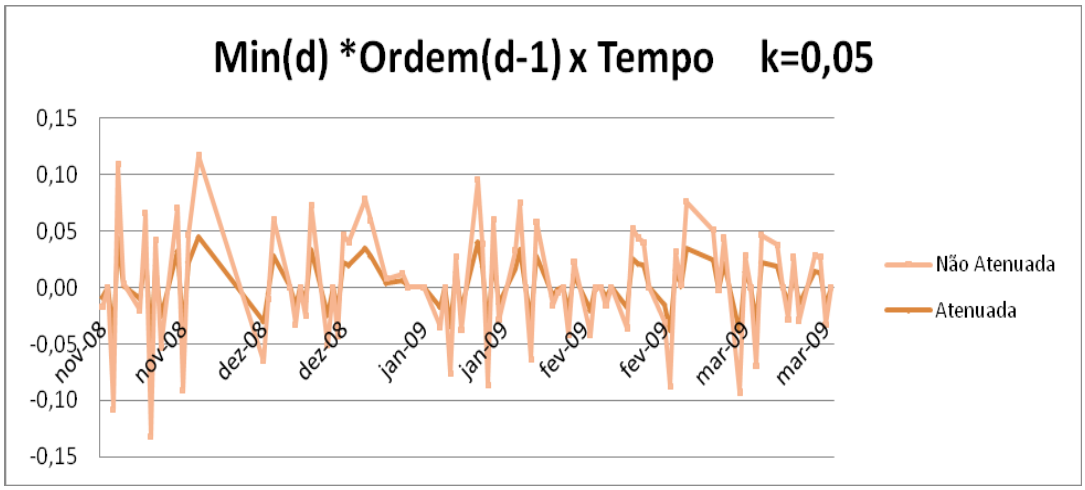

Figura 2.2.10- Mínima(d)*Ordem(d-1) x Tempo, k=0,05

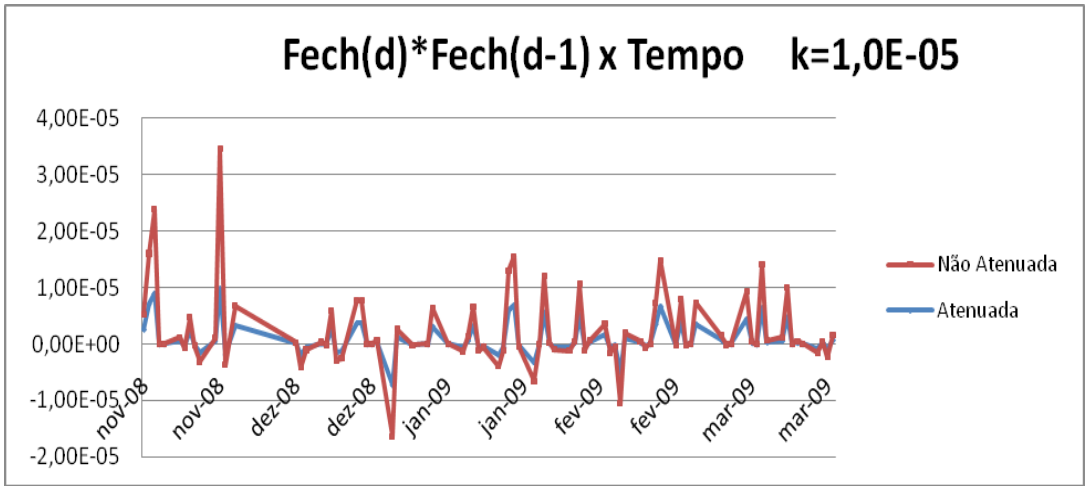

Figura 2.2.11- Fechamento(d)*Fechamento(d-1) x Tempo, k=1,0E-05

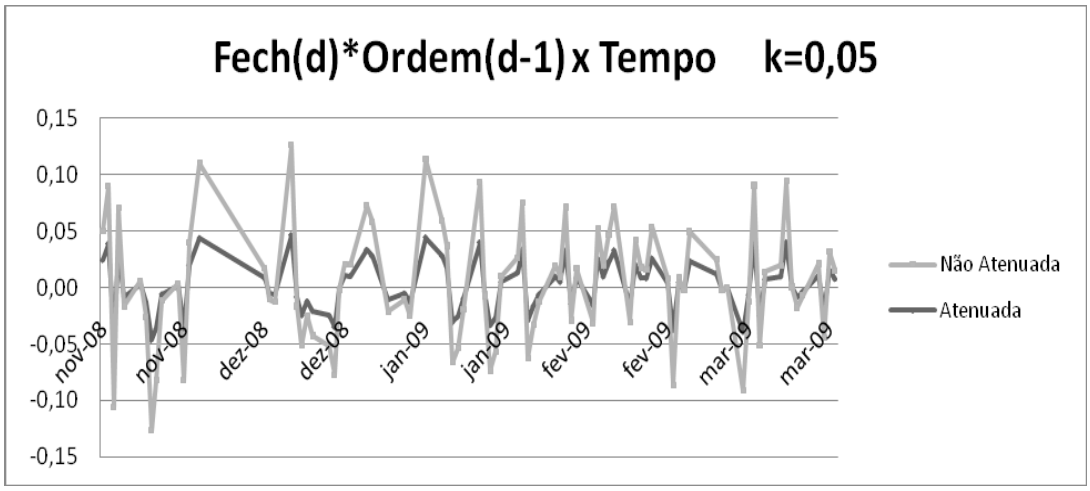

Figura 2.2.12- Fechamento(d)*Ordem(d-1) x Tempo, k=0,05

2.3 – Escolhendo a janela de análise

Para analisar quais os dias serão utilizados no modelo vamos aduçar o produto entrada*saída x tempo através de média móvel com diferentes janelas, para entender qual a melhor janela de dados devemos usar para considerar os dias operáveis do modelo.

Para calcular a média móvel para todas os pares entrada/saída utilizamos o algoritmo:

$$A_i = \frac{\sum_{i}^{i+N-1} S_i}{N}$$

Onde,

A_i = Média móvel do produto entrada*saída

N = Número de dias da janela móvel

S_i = Produto entrada*saída

Para todas os 10 pares entrada/saída calculamos a média móvel para 3, 5 e 7 dias e *plotamos* para analisar qual a janela mais estável a ser utilizada. Abaixo segue um exemplo dos gráficos de média móvel para as 3 janelas do par Ordem(d)*Abertura(d-1) x Tempo.

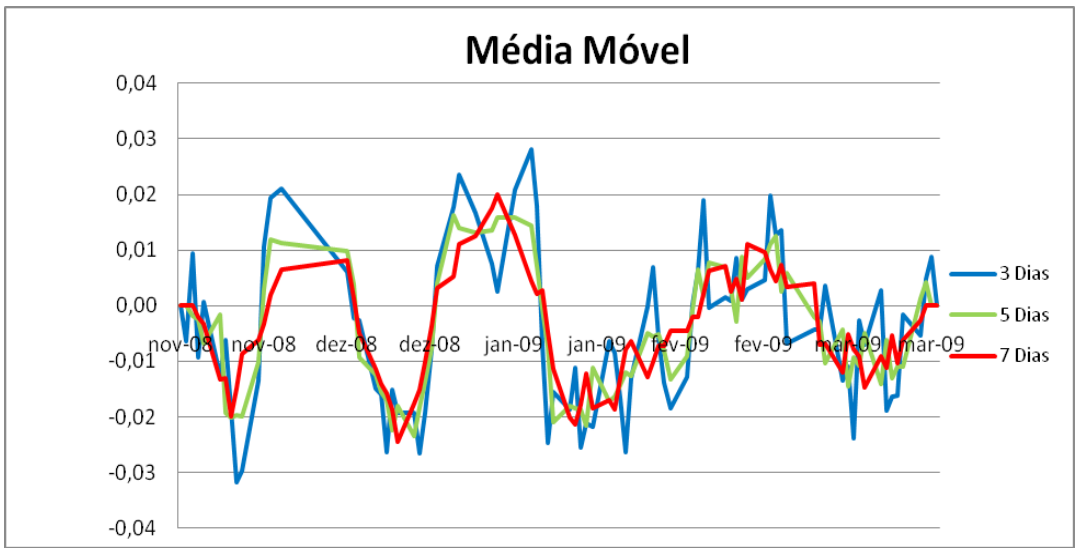

Figura 2.3.1- Média móvel do par Ordem(d)*Abertura(d-1) x Tempo, com janelas de 3, 5 e 7 dias.

O programa "*Plota* Média Movel" desenvolvido no Matlab permite *plotar* a média móvel de todos os pares entrada/saída com qualquer janela de média móvel desejada. O código completo deste programa se encontra no Apêndice A deste documento.

Utilizando a interface do programa criado no Matlab, *plotamos* a média móvel para 3, 5 e 7 dias do mesmo par entrada/saída *plotado* acima, Ordem(d)*Abertura(d-1) x Tempo, com as mesmas cores de legenda. O par desejado é selecionado com a *checkbox* e a janela é digitada na janela ao lado do par. Para gerar o gráfico basta clicar no botão Gera Gráfico, que irá *plotar* todos os gráficos na mesma tela até que o botão Limpa seja clicado, limpando todos os gráficos da tela.

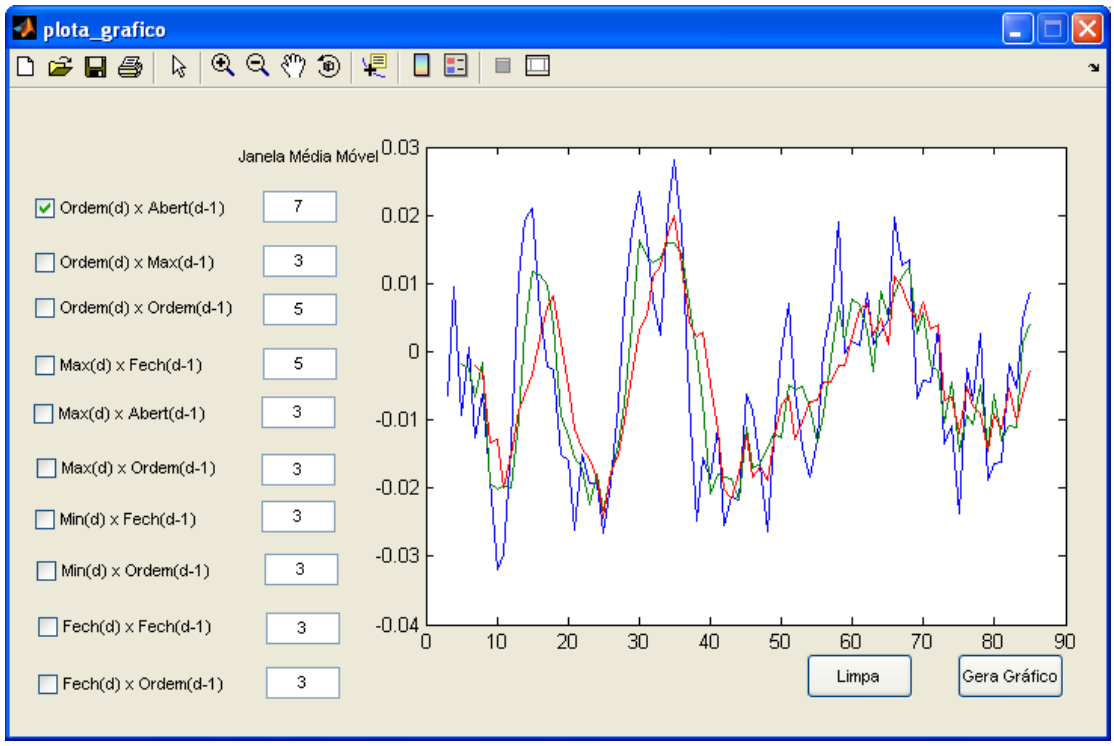

Figura 2.3.2 – Média móvel do par Ordem(d)*Abertura(d-1) para as janelas de 3, 5 e 7 dias.

Podemos perceber que o resultado da média móvel *plotado* no Matlab para as diferentes janelas (Fig.2.3.2) é o mesmo que o obtido no Excel (Fig.2.3.1). O programa "*Plota* Média Móvel" permite que o usuário escolha e visualize, de maneira mais flexível e amigável, os gráficos de média móvel com diferentes janelas para todos os pares entrada/saída. Diversos gráficos podem se sobrepor na tela facilitando a sua comparação.

Após analisar os gráficos de média móvel para as janelas de 3, 5 e 7 dias de todos os pares de entrada, escolhemos utilizar a janela móvel de 7 dias para determinar quais os dias serão escolhidos como operáveis pelo nosso modelo. Escolhemos esta janela por apresentar menor volatilidade em relação às outras duas, o que já era esperado, e por ainda transimitir alguma variação da "covariância".

2.4 – Determinação dos dias operáveis

Para determinar quais os dias utilizaríamos para operar no nosso modelo, assumimos as seguintes premissas:

1. Evitaríamos dias de variação grande na pontuação do Ibovespa, pois a alta volatilidade iria não seria captada pelo nosso modelo de estimação linear.
2. Evitaríamos dias onde a correlação entre os pares de entrada estivesse próxima de zero, pois isto não indicaria nenhuma tendência para o Ibovespa.

Após estabelecer estas premissas, utilizamos as séries dos pares entrada x saída atenuados, ilustrados no item 2.2. Sobre estas séries de pares entrada x saída atenuadas foi calculada a média móvel com janela de 7 dias, definida no item 2.3, para cada data da série. Calculamos ainda o desvio padrão de cada série de média móvel de 7 dias dos pares entrada saída.

Média Móvel Janela 7 Dias		
Saídas	Entradas	Desvio Padrão
Ordem(d) x	Abertura(d-1)	9,85E-03
	Máxima(d-1)	7,53E-03
	Ordem(d-1)	4,67E-01
Máxima(d) x	Fechamento(d-1)	8,02E-07
	Abertura(d-1)	2,13E-04
	Ordem(d-1)	7,49E-03
Mínima(d) x	Fechamento(d-1)	9,22E-07
	Ordem(d-1)	5,22E-03
Fechamento(d) x	Fechamento(d-1)	8,66E-07
	Ordem(d-1)	8,33E-03

Tabela 2.4.1 – Desvio padrão das séries de média móvel

Afim de identificar variações grandes na pontuação do Ibovespa, calculamos quanto cada ponto da série de média móvel se desviava da média dos dois pontos

anteriores. Com isso obtivemos para cada par entrada x saída, a série de desvios em relação à média dos dois dias anteriores.

Desvio em relação à média		
Saídas	Entradas	Desvio Padrão
Ordem(d) x	Abertura(d-1)	5,74E-03
	Máxima(d-1)	4,94E-03
	Ordem(d-1)	2,65E-01
Máxima(d) x	Fechamento(d-1)	5,92E-07
	Abertura(d-1)	1,43E-04
	Ordem(d-1)	4,77E-03
Mínima(d) x	Fechamento(d-1)	6,50E-07
	Ordem(d-1)	4,41E-03
Fechamento(d) x	Fechamento(d-1)	7,04E-07
	Ordem(d-1)	5,86E-03

Tabela 2.4.2 – Desvio padrão das series de desvio à média

Afim de seguir as premissas estabelecidas neste item, definimos que para evitar grandes variações no Ibovespa, bastaria elminar os dias onde a média móvel variou acima de determinado valor de desvio padrão. Escolhemos 1.2 desvios padrão de variação como limite máximo para considerarmos um dia operável pelo modelo.Para evitar valores de correlação próximos à zero, definimos que o a média mínima aceitável para as correlações deveria ser de 0.1 desvios padrão.

Para um determinado dia ser considerado operável em uma série definimos que ele deveria obedecer às duas restrições estabelecidas nas premissas, logo cada valor da série deveria estar acima de 0.1 desvios padrão e abaixo de 1.2 desvios padrão da sua série. Para um determinado dia ser operável pelo modelo, utilizamos uma restrição ainda maior, este dia deveria ser operável para todos as outras séries. Com isso restringimos o nosso modelo a operar somente em dias nos quais todas as séries estão obedecendo às premissas estabelecidas neste item.

Abaixo mostramos uma tabela resumindo os resultados de limite mínimo e máximo para cada série assim como o percentual de dias operáveis em cada série. Mostramos ainda o resultado final com o percentual dos dias operáveis do modelo, considerando que todas as séries também são operáveis nestes dias.

Percentual de dias operáveis				
Saída	Entrada	Desvio Máximo	Média Mínima	Dias Operáveis
Ordem(d) x	Abertura(d-1)	1,18E-02	5,74E-04	97%
	Máxima(d-1)	9,04E-03	4,94E-04	86%
	Ordem(d-1)	5,60E-01	2,65E-02	100%
Máxima(d) x	Fechamento(d-1)	9,63E-07	5,92E-08	84%
	Abertura(d-1)	2,55E-04	1,43E-05	88%
	Ordem(d-1)	8,98E-03	4,77E-04	92%
Mínima(d) x	Fechamento(d-1)	1,11E-06	6,50E-08	88%
	Ordem(d-1)	6,26E-03	4,41E-04	81%
Fechamento(d) x	Fechamento(d-1)	1,04E-06	7,04E-08	82%
	Ordem(d-1)	1,00E-02	5,86E-04	86%
		Total de Dias Operáveis		32%

Tabela 2.4.3 – Percentual de dias operáveis por série e total do modelo

Aplicando a restrição de utilizarmos somente dias onde todas as séries são operáveis, chegamos ao resultado de 32% de dias operáveis na nossa janela de dados utilizada.

Um segundo programa no Matlab chamado "Negocia", *plota* o gráfico da série entrada x saída e os dias operáveis sobre o gráfico de cada série. Para *plotar* os dias operáveis traçamos retas verticais, onde cada reta representa um dia operável na série. É possível ainda *plotar* o gráfico do Ibovespa para a janela utilizada no nosso trabalho e analisar quais os dias foram indicados como operáveis pelo nosso modelo.

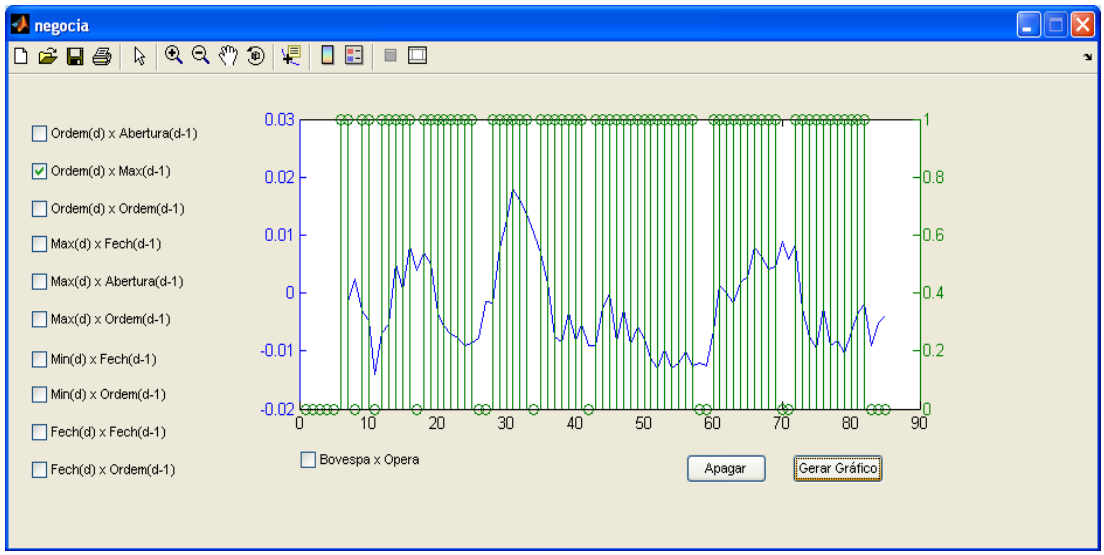

Figura 2.4.1 – Interface do programa "Negocia" mostrando os dias operáveis para todos os pares entrada x saída

Abaixo mostramos o resultado final do estudo de dias operáveis, *plotando* o gráfico do Ibovespa e os dias operáveis do modelo. Podemos perceber que os dias operáveis em casa série são quase a maioria, enquanto que os dias operáveis pelo modelo todo são apenas 32%, o que é um resultado significante, dado que 1/3 dos dias selecionados possui um comportamento suficiente para ser modelado linearmente.

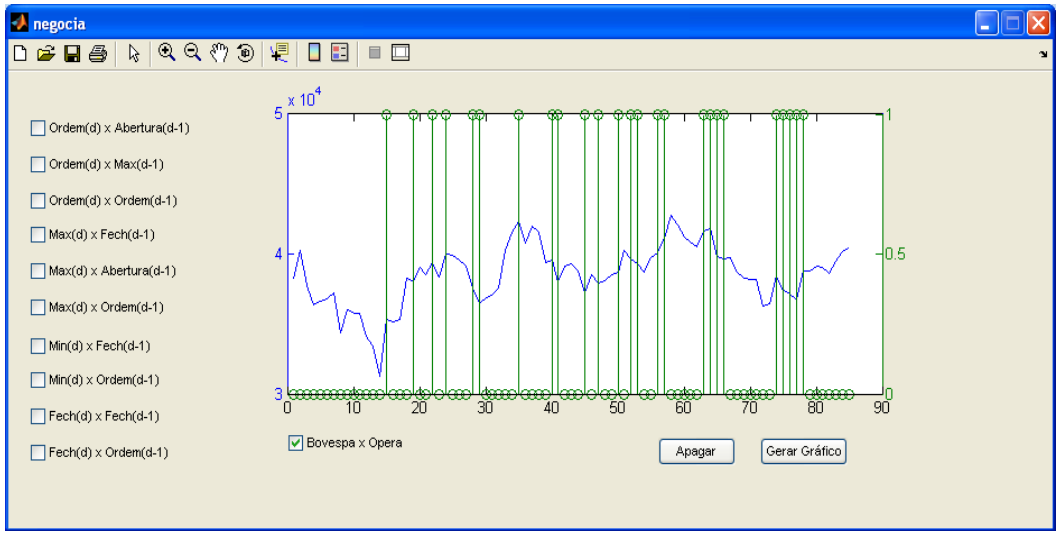

Figura 2.4.2 – Interface do programa "Negocia" mostrando os dias operáveis do modelo sobre o gráfico do Ibovespa no período.

Este programa possui funcionamento parecido com o anterior, "*Plota* Média Móvel", onde apenas as séries entrada x saída selecionadas pelas *checkboxes* são *plota*das na janela. Os gráficos são gerados apenas ao clicar no botão Gera Gráfico e são apagados após clicar no botão Apagar. O código deste programa encontra-se no Apêndice A deste trabalho.

Capítulo 3

Redes Neurais

3.1 – Preparando a rede

Redes neurais são poderosas ferramentas na modelagem de sistemas não lineares mas neste trabalho utilizamos redes com função de ativação linear, afim de capturar o comportamento linear dos dados após o tratamento realizado no capítulo 2. Adotamos esta estratégia porque diversos trabalhos anteriores mostraram que devido ao sistema ser muito ruidoso e rapidamente variável no tempo, não é possível capturar sua não linearidade de forma eficiente.

Para cada variável de saída desejada montamos uma rede com ativação linear, considerando as variáveis de entrada mais significantes para cada saída desejada, conforme mostrado na tabela 2.2.2 deste trabalho. Abaixo segue o modelo de rede neural utilizado, neste exemplo com apenas 2 entradas:

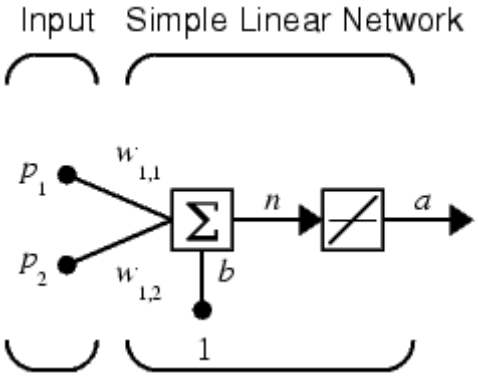

Figura 3.1.1 – Modelo de rede neural com ativação linear

Onde,

$$a = w_{1,1}p_1 + w_{1,2}p_2 + b$$

Podemos perceber que a variável de saída *a* depende linearmente do somatório das entradas ponderadas pelos seus respectivos pesos *w*. Estes pesos são calculados através do treinamento da rede que será explicado posteriormente. Abaixo ilustramos o modelo da rede neural linear para cada saída estudada neste trabalho (ver tabela 2.2.2).

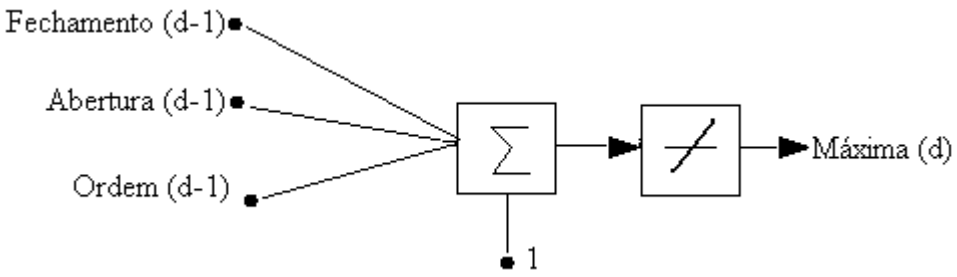

Figura 3.1.2 – Rede neural linear para estimar a variável Máxima(d)

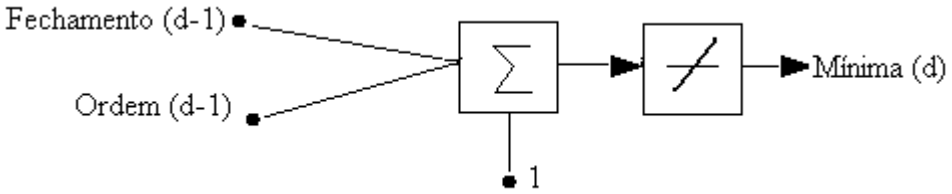

Figura 3.1.3 – Rede neural linear para estimar a variável Mínima(d)

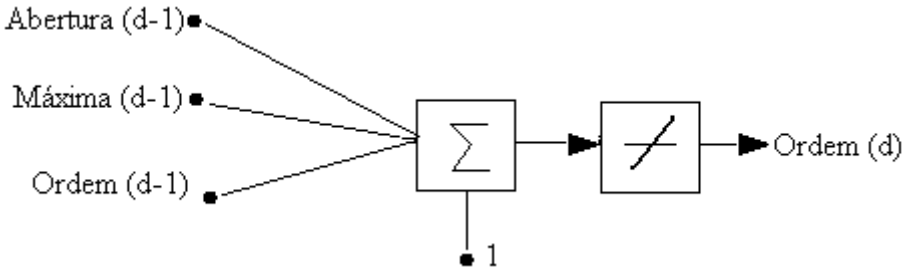

Figura 3.1.4 – Rede neural linear para estimar a variável Ordem(d)

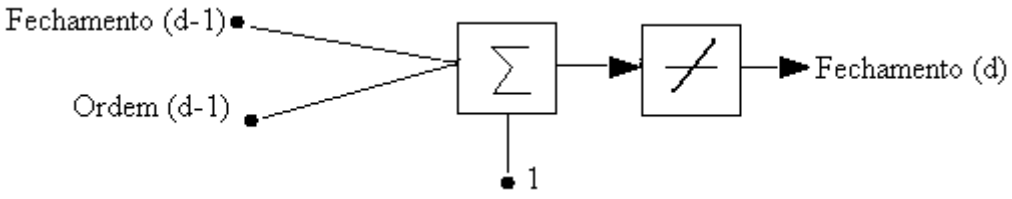

Figura 3.1.5 – Rede neural linear para estimar a variável Fechamento(d)

Antes de utilizar as variáveis nos modelos acima, normalizamos as séries utilizando a fórmula abaixo:

$$X_{Norm} = \frac{X - \overline{S}}{\sigma_S}$$

Onde,

$\overline{S}$: Média da série da váriável a ser normalizada

σ_S : Desvio padrão da série da variável a ser normalizada

Para criar as redes acima, utilizei o Toolbox de redes neurais do Matlab. Para criar uma rede neural chamada "***net***" com função de ativação linear, utilizei a seguinte função:

$$Net = newlin(P, T, ID, LR)$$

Onde,

P : Matriz RxQ, com R elementos de entrada

T : Matriz SxU, com S elementos de saída

ID : *Atraso do vetor de entrada, por default =0*

LR : Taxa de aprendizado, por default=0.01

Utilizamos os valores default de ID e LR, o vetor das entradas transposto como P e T=1.

3.2 – Treinando a rede

Para realizar o treinamento das redes, utilizamos apenas os dias operáveis para todas as variáveis ao mesmo tempo, limitando o nosso número de dias a 32% do total, como mostrado na tabela 2.4.3. Com isso a nossa rede foi treinada com valores de entrada e saída apenas dos dias que estavam dentro das premissas estabelecidas do modelo, evitando um treinamento para valores errados da rede.

Utilizando ainda o Toolbox de Redes Neurais do Matlab para tratar as redes, simulamos cada rede com a seguinte função:

$$TrainedNet = train(net, P, T, P_i, A_i)$$

Onde,

net : Rede criada anteriormente para cada variável de saída

P : Matriz de entradas

T : Vetor de saída com resultados esperados

P_i : Atraso das entradas inicial (default=0)

A_i : Atraso das camadas inicial (default=0)

Utilizamos Pi e Ai com os valores default, a matriz P e o vetor T apenas dos dias operáveis pelo modelo, ou seja, que possuiam o dia operável por todas as variáveis de entrada ao mesmo tempo.

3.3 – Resultados

Após montar e treinar as redes com a janela de dados de referência, utilizamos todos os dados de entrada para simular a resposta da rede ao prever o comportamento dos valores do dia baseado no treinamento realizado.

Para simular a rede utilizamos o Toolbox de redes neurais do Matlab e utilizamos a seguinte função:

$$Estimated = sim(net, P, P_i, A_i, T)$$

Onde,

net : Rede treinada anteriormente para cada variável de saída

P : Matriz de entradas

T : Vetor de saída com resultados esperados

P_i : Atraso das entradas inicial (default=0)

A_i : Atraso das camadas inicial (default=0)

A seguir comparamos resultado obtido com cada rede e o resultado esperado, mostrando que a aproximação linear é bastante eficiente apesar de haver um atraso na identificação de mudança de tendência, clara nos gráficos mostrados, que estão com os resultados normalizados.

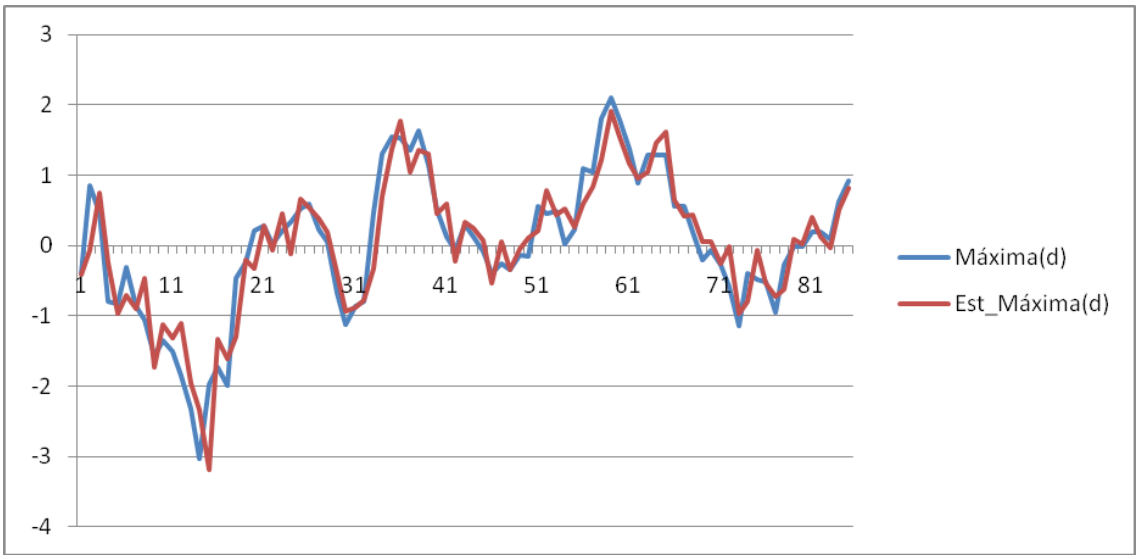

Figura 3.3.1 – Gráfico do resultado estimado x ocorrido da variável Máxima(d)

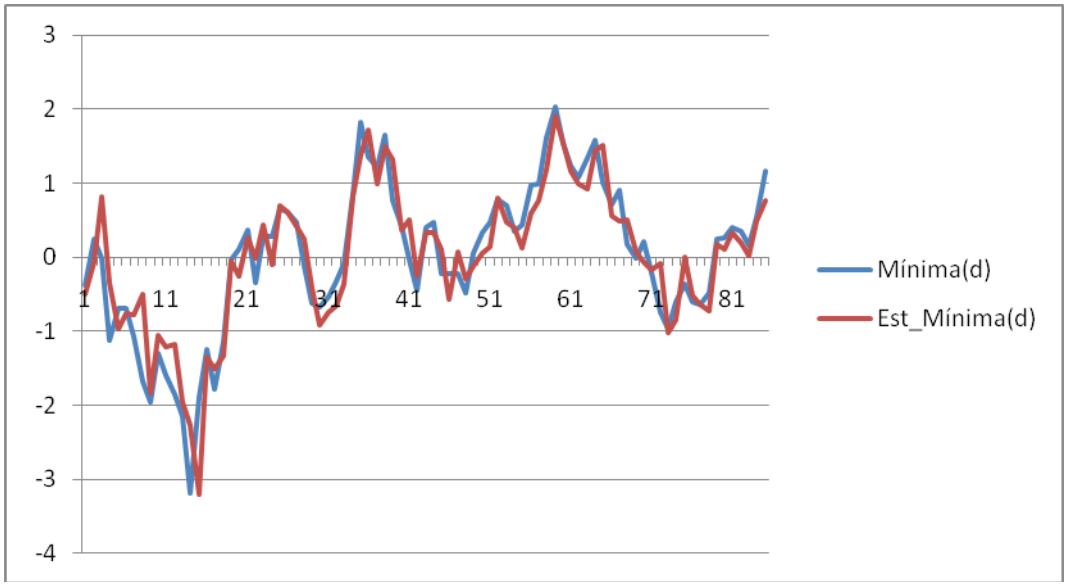

Figura 3.3.2 – Gráfico do resultado estimado x ocorrido da variável Mínima(d)

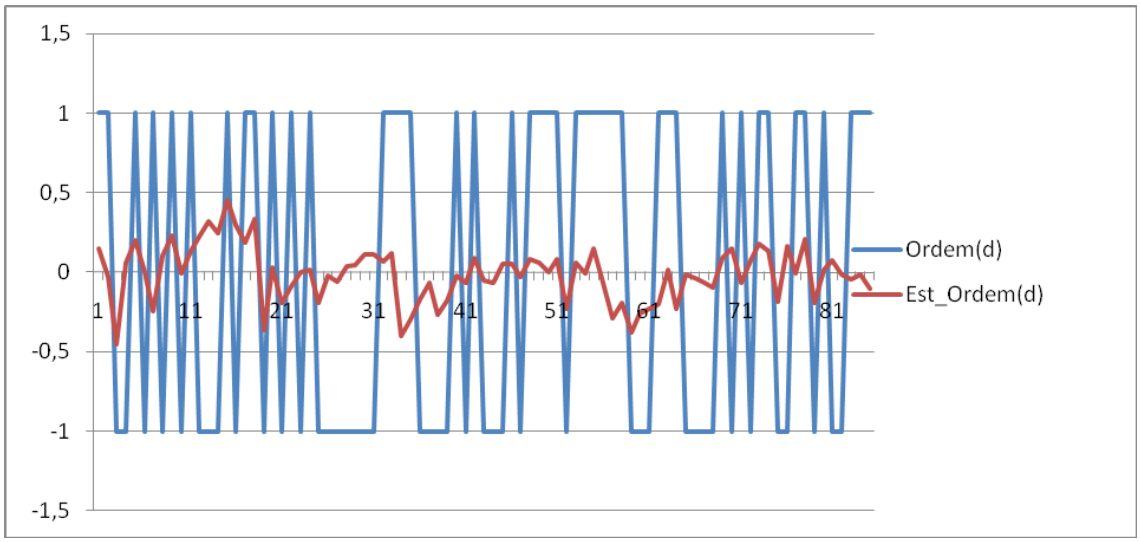

Figura 3.3.3 – Gráfico do resultado estimado x ocorrido da variável Ordem(d)

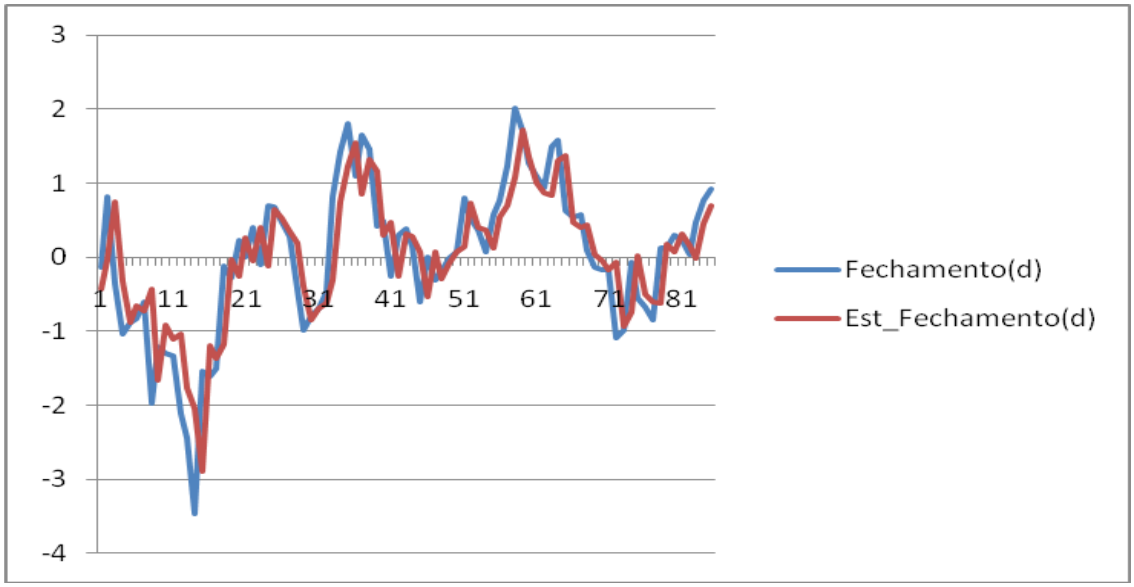

Figura 3.3.4 – Gráfico do resultado estimado x ocorrido da variável Fechamento(d)

A estimação da ordem de acontecimento da máxima e mínima do dia apresentou resultados bastante divergentes devido ao modelo de predição linear. Contudo as simulações das variáveis de Máxima, Mínima e Fechamento do dia se comportaram bem com uma função de ativação linear e obtiveram previsão próxima do ocorrido.

Capítulo 4

Simulações e Conclusões

4.1 - Operações e lucros acumulados

Após obtermos os resultados das simulações dos preditores lineares, simulamos operações no mercado financeiro que permitem utilizar os resultados estimados como referência para operar no mercado. Utilizando a estimação do fechamento do Ibovespa para o dia, podemos identificar se o indice vai ter tendência de alta ou baixa no dia.

$$Tendência(d) = Est\_Fech(d) - Abertura(d)$$

se Tendência(d) < 0 => Prevemos uma tendência de baixa para o índice
se Tendência(d) > 0 => Prevemos uma tendência de alta para o índice

Após identificarmos esta tendência escolhemos entre duas estratégias para operar no mercado, estratégia comprada ou vendida no índice.

Se Tendência(d) for de baixa => Estratégia vendida
Se Tendência(d) for de baixa => Estratégia comprada

Os lucros de cada estratégia são calculados da seguinte maneira:

$$LucroEstrategiaComprada = Fech(d) - Abertura(d)$$

$$LucroEstrategiaVendida = Abertura(d) - Fech(d)$$

Realizamos algumas simplificações para tornar os resultados mais claros e diretos de obtermos, mas que de forma alguma prejudicam o resultado final, pois não assumem nenhuma premissa irreal no mercado. Para apurar os resultados diários assumimos que:

- ✓ O investidor executa o modelo assim que o mercado abre, utilizando a pontuação de abertura como referência.
- ✓ A decisão de compra ou venda é feita assim que o mercado abre e o preço de compra ou venda do índice é o mesmo que o de abertura, assumimos que ainda não ocorreu nenhuma operação que afetasse o índice entre a abertura e a operação.
- ✓ Não estamos negociando um produto específico do mercado brasileiro, apenas comprando e vendendo o próprio índice. Contudo esta é uma simplificação válida dado que diversos instrumentos financeiros são indexados ao índice, logo apurarmos o resultado da variação do índice equivale a apurar o resultado de diversos outros instrumentos, só que de maneira mais simplificada.
- ✓ Assumimos que o investidor "zera" a sua posição ao final do dia, com o preço de fechamento. Se ele estava comprado no índice, ele o vende no fechamento e se estava vendido ele o compra no fechamento, zerando sua exposição e apurando o resultado com preço final igual ao de fechamento.
- ✓ O resultado acumulado não considera nenhum custo de oportunidade ou desconto intertemporal de capital.

Após assumir estas premissas executamos nossas estratégias sobre a janela de dados de novembro/08 a março/09 e obtivemos um retorno acumulado de 30% sobre o capital investido, enquanto que o Ibovespa obteve retorno de 6% sobre o capital no mesmo período.

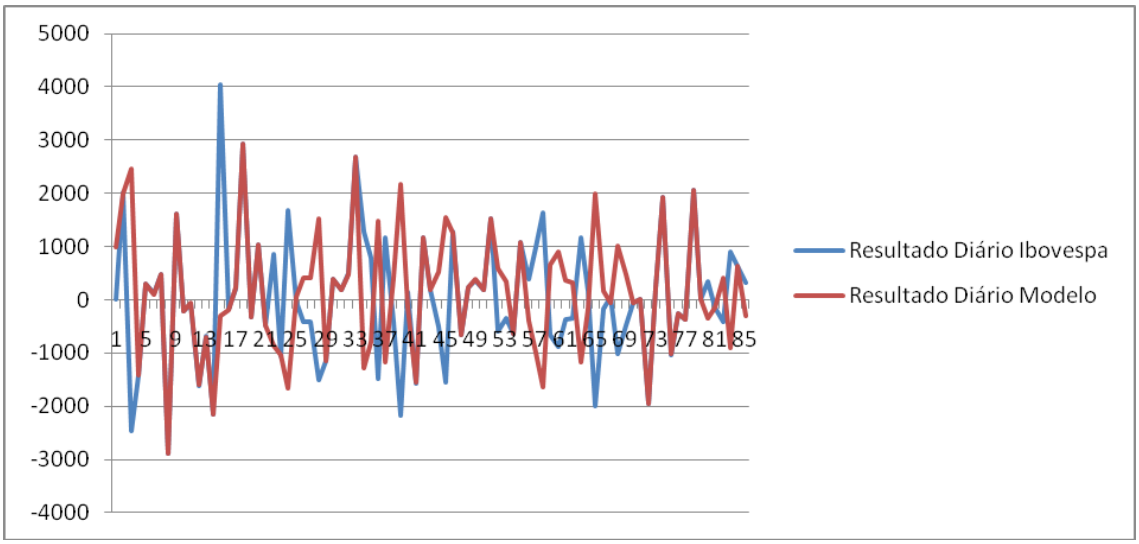

Figura 4.1.1 – Gráfico comparando o resultado diário do Ibovespa x resultado diário das estratégias utilizando o modelo de redes neurais

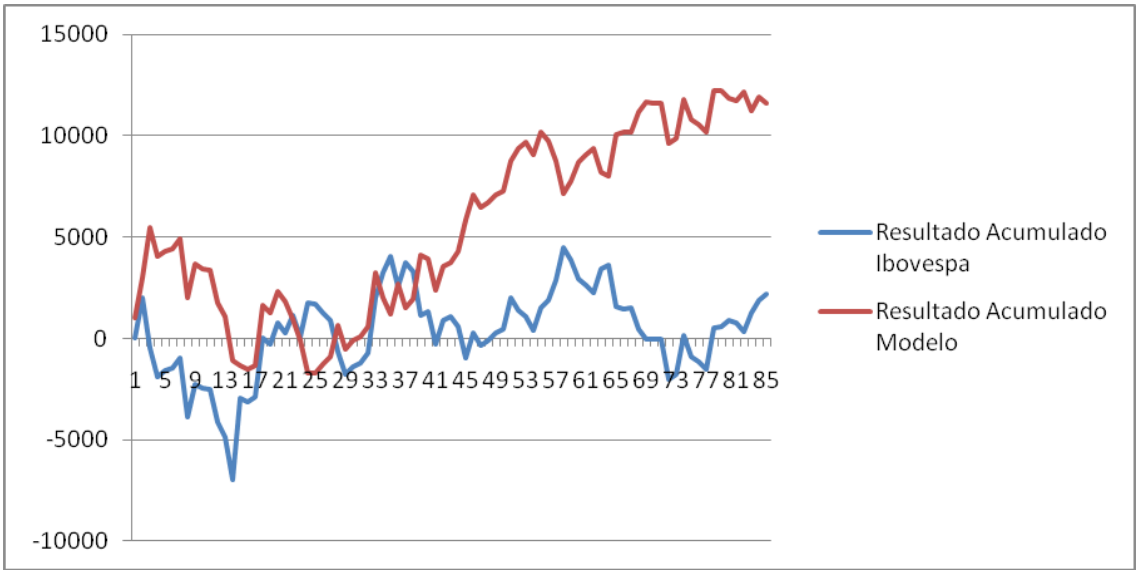

Figura 4.1.2 – Gráfico comparando o resultado acumulado do Ibovespa x resultado acumulado das estratégias utilizando o modelo de redes neurais

Assumimos agora que o investidor vai utilizar as informações de máxima e mínima esperadas do dia para zerar sua posição à um preço mais favorável. Se estiver em uma estratégia comprada, o investidor vai zerar sua posição vendendo assim que o mercado atingir o preço máximo esperado do dia. Da mesma maneira, se estiver em uma estratégia vendida, o investidor vai zerar sua posição comprando assim que o

mercado atingir o preço mínimo esperado do dia. Se estes preços limites de máxima e mínima não forem atingidos durante o dia, o investidor zera sua posição com preço de fechamento.

Utilizamos uma faixa de erro de 2% (pois o erro médio foi de 1,5% aproximadamente) para os valores de máxima e mínima esperados do dia, logo se o o valor de mercado observado estiver à menos de 2% de diferença do máximo ou mínimo esperados, a posição é zerada a este valor de mercado.

Assumindo estas novas premissas, obtivemos o retorno diário e acumulado do modelo utilzando estratégias com valores intraday, comparando com o retorno do Ibovespa no período. Podemos observar que o retorno foi ainda maior que na estratégia anterior, que considerava apenas os preços de fechamento para zerar as posições. O retorno sobre o capital investido chega a 45% contra os 30% obtidos com a estratégia de fechamento.

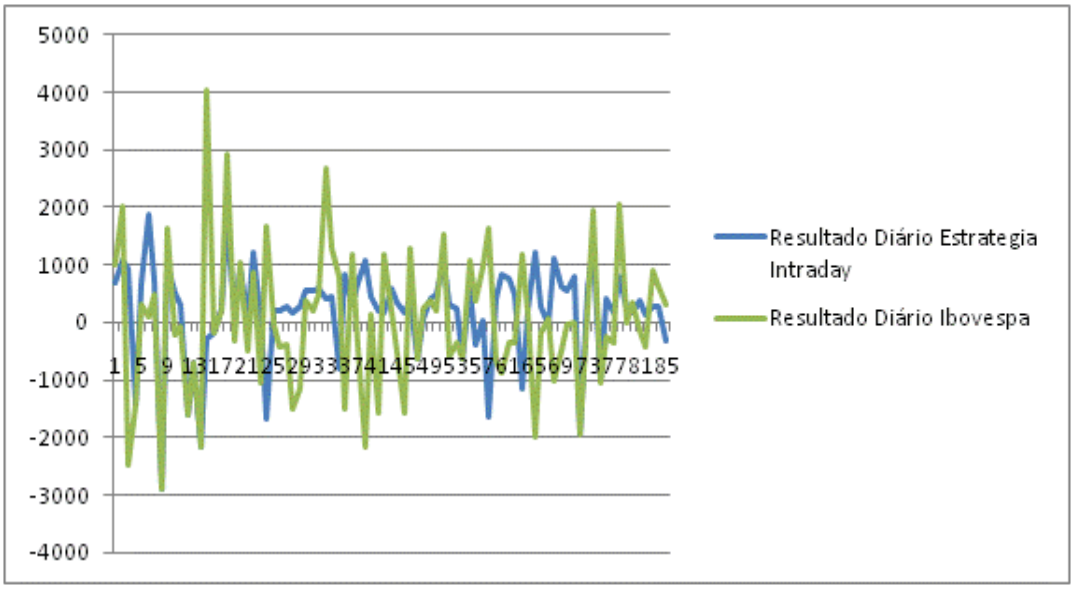

Figura 4.1.3 – Gráfico comparando o resultado diário do Ibovespa x resultado diário das estratégias utilizando estratégia intraday

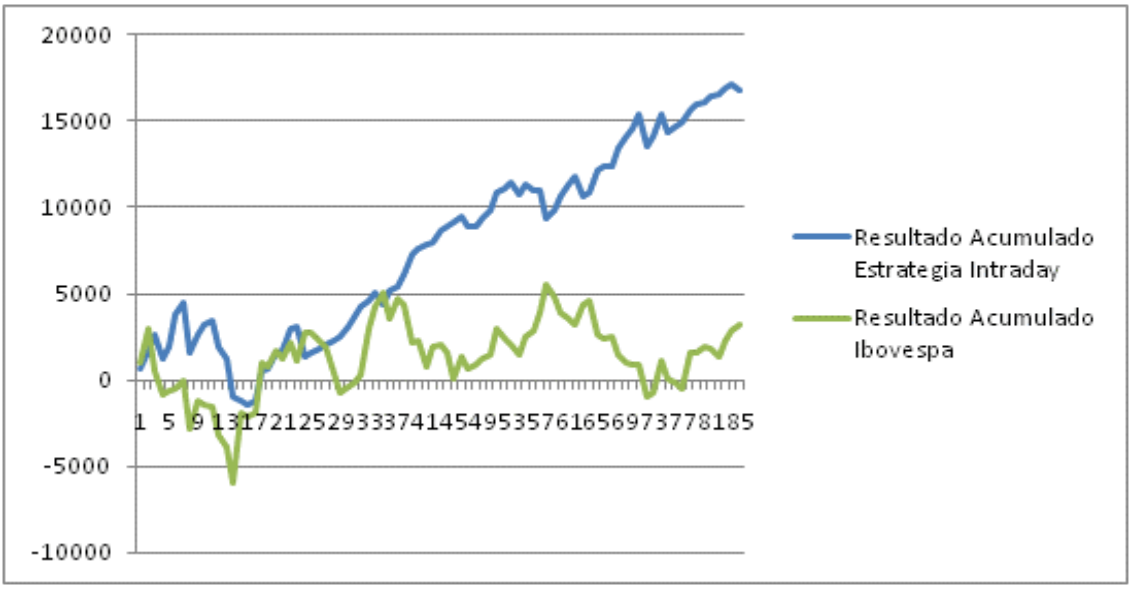

Figura 4.1.4 – Gráfico comparando o resultado acumulado do Ibovespa x resultado acumulado das estratégias utilizando estratégia intraday

4.2 – Conclusões

O retorno total sobre o capital investido obtido com as estratégias adotadas à partir dos resultados das simulações foi de até 45% enquanto que o retorno sobre o mesmo capital investido no Ibovespa no período foi de 5,7%. Obtivemos portanto um retorno mais de 5 vezes maior que o Ibovespa, um resultado interessante para uma operação no mercado de renda variável.

Demonstramos portanto que, se realizados os tratamentos necessários sobre os dados de entrada e tomadas as devidas precauções na escolha dos dias para treinar a rede neural, é possível obter retornos acima do normal com previsões sobre o fechamento do Ibovespa utilizando redes neurais com função de ativação linear.

Este estudo sugere um aprofundamento maior na estimação das variáveis utilizando redes neurais com mais camadas e modelos de treinamento mais complexos, buscando também a não linearidade do comportamento do mercado financeiro.

Bibliografia

[1]WASSEMAN, P. D. Neural Computing, Theory and Pratice. New York: Van Nostrand Reinhold, 1989

[2]CALÔBA, L.O, CALÔBA, L.P, CONTADOR, C.R, Delta –Neutral Volatility Trading using Neural Networks, Int. J. of Engineering Intelligent Systems, Vol. 9, No. 4, December 2001, pp. 243-249.

[3] CALÔBA, L.P., Introdução ao Uso de Redes Neurais na Modelagem de Sistemas Dinâmicos e Séries Temporais, Livro de Minicursos do XIV Congresso Brasileiro de Automática, Natal, 2002.

[4]CALÔBA, L. P., BRAGA P.C., Ibovespa Forecasting Using Hybrid Models, COPPE/UFRJ – 2007.

Apêndice A

Programa "*Plota* Média Móvel"

```
function varargout = plota_grafico(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @plota_grafico_OpeningFcn, ...
                   'gui_OutputFcn',  @plota_grafico_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before plota_grafico is made visible.
function plota_grafico_OpeningFcn(hObject, eventdata, handles, varar-
gin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to plota_grafico (see VARARGIN)

% Choose default command line output for plota_grafico
handles.output = hObject;

set(hObject,'toolbar','figure');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes plota_grafico wait for user response (see UIRESUME)
% uiwait(handles.lag_ch1);


% --- Outputs from this function are returned to the command line.
function varargout = plota_grafico_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

global ch1;
if (get(hObject,'Value') == get(hObject,'Max'))

ch1=1;

else

ch1=0;

end;

% --- Executes on button press in clear_pushbutton3.
function clear_pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to clear_pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


cla(handles.axes1,'reset')
guidata(hObject, handles); %updates the handles


% --- Executes on button press in plot_pushbutton4.
function plot_pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to plot_pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load estudo_media_movel.mat
global ch1;
global ch2;
global ch3;
global ch4;
global ch5;
global ch6;
global ch7;
global ch8;
global ch9;
global ch10;

axes(handles.axes1)

if ch1==1
lag_med_mov = str2num(get(handles.lag1,'String'))
plot(tsmovavg(k1_Ordem_dxAbert_d_1,'s',lag_med_mov,1))
hold all
end;

if ch2==1
lag_med_mov = str2num(get(handles.lag2,'String'))
plot(tsmovavg(k2_Ordem_dxMax_d_1,'s',lag_med_mov,1))
hold all
end;
```

```
if ch3==1
lag_med_mov = str2num(get(handles.lag3,'String'))
plot(tsmovavg(k3_Ordem_dxOrdem_d_1,'s',lag_med_mov,1))
hold all
end;

if ch4==1
lag_med_mov = str2num(get(handles.lag4,'String'))
plot(tsmovavg(k4_Max_dxFech_d_1,'s',lag_med_mov,1))
hold all
end;

if ch5==1
lag_med_mov = str2num(get(handles.lag5,'String'))
plot(tsmovavg(k5_Max_dxAbert_d_1,'s',lag_med_mov,1))
hold all
end;

if ch6==1
lag_med_mov = str2num(get(handles.lag6,'String'))
plot(tsmovavg(k6_Max_dxOrdem_d_1,'s',lag_med_mov,1))
hold all
end;

if ch7==1
lag_med_mov = str2num(get(handles.lag7,'String'))
plot(tsmovavg(k7_Min_dxFech_d_1,'s',lag_med_mov,1))
hold all
end;

if ch8==1
lag_med_mov = str2num(get(handles.lag8,'String'))
plot(tsmovavg(k8_Min_dxOrdem_d_1,'s',lag_med_mov,1))
hold all
end;

if ch9==1
lag_med_mov = str2num(get(handles.lag9,'String'))
plot(tsmovavg(k9_Fech_dxFech_d_1,'s',lag_med_mov,1))
hold all
end;

if ch10==1
lag_med_mov = str2num(get(handles.lag10,'String'))
plot(tsmovavg(k10_Fech_dxOrdem_d_1,'s',lag_med_mov,1))
hold all
end;


guidata(hObject, handles);



function lag1_Callback(hObject, eventdata, handles)
% hObject    handle to lag1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function lag1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)

global ch2;
if (get(hObject,'Value') == get(hObject,'Max'))

ch2=1;

else

ch2=0;

 end;

function lag2_Callback(hObject, eventdata, handles)
% hObject    handle to lag2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag2 as text
%        str2double(get(hObject,'String')) returns contents of lag2 as
a double

input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3
global ch3;
if (get(hObject,'Value') == get(hObject,'Max'))

ch3=1;

else

ch3=0;

 end;

function lag3_Callback(hObject, eventdata, handles)
% hObject    handle to lag3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag3 as text
%        str2double(get(hObject,'String')) returns contents of lag3 as
a double

input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4
global ch4;
if (get(hObject,'Value') == get(hObject,'Max'))

ch4=1;

else

ch4=0;

end;

function lag4_Callback(hObject, eventdata, handles)
% hObject    handle to lag4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag4 as text
%        str2double(get(hObject,'String')) returns contents of lag4 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox5
global ch5;
if (get(hObject,'Value') == get(hObject,'Max'))

ch5=1;
```

```
else

ch5=0;

end;

function lag5_Callback(hObject, eventdata, handles)
% hObject    handle to lag5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag5 as text
%        str2double(get(hObject,'String')) returns contents of lag5 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox6
global ch6;
if (get(hObject,'Value') == get(hObject,'Max'))

ch6=1;

else

ch6=0;

end;

function lag6_Callback(hObject, eventdata, handles)
% hObject    handle to lag6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of lag6 as text
%        str2double(get(hObject,'String')) returns contents of lag6 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7
global ch7;
if (get(hObject,'Value') == get(hObject,'Max'))

ch7=1;

else

ch7=0;

end;

function lag7_Callback(hObject, eventdata, handles)
% hObject    handle to lag7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag7 as text
%        str2double(get(hObject,'String')) returns contents of lag7 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
```

```
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox8
global ch8;
if (get(hObject,'Value') == get(hObject,'Max'))

ch8=1;

else

ch8=0;

end;

function lag8_Callback(hObject, eventdata, handles)
% hObject    handle to lag8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag8 as text
%        str2double(get(hObject,'String')) returns contents of lag8 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox9.
function checkbox9_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox9
global ch9;
if (get(hObject,'Value') == get(hObject,'Max'))

ch9=1;

else

ch9=0;

 end;



function lag9_Callback(hObject, eventdata, handles)
% hObject    handle to lag9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag9 as text
%        str2double(get(hObject,'String')) returns contents of lag9 as
a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in checkbox10.
function checkbox10_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox10
global ch10;
if (get(hObject,'Value') == get(hObject,'Max'))

ch10=1;

else

ch10=0;


end;



function lag10_Callback(hObject, eventdata, handles)
% hObject    handle to lag10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lag10 as text
%        str2double(get(hObject,'String')) returns contents of lag10
as a double


input = str2num(get(hObject,'String'));
%checks to see if input is empty. if so, default input1_editText to
zero
if (isempty(input))
     set(hObject,'String','0')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function lag10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lag10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Programa “Negocia”

```
function varargout = negocia(varargin)
% NEGOCIA M-file for negocia.fig
%      NEGOCIA, by itself, creates a new NEGOCIA or raises the exist-
ing
%      singleton*.
%
%      H = NEGOCIA returns the handle to a new NEGOCIA or the handle
to
%      the existing singleton*.
%
%      NEGOCIA('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in NEGOCIA.M with the given input argu-
ments.
%
%      NEGOCIA('Property','Value',...) creates a new NEGOCIA or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before negocia_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property ap-
plication
%      stop.  All inputs are passed to negocia_OpeningFcn via varar-
gin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
```

```
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help negocia

% Last Modified by GUIDE v2.5 12-Jul-2009 11:56:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @negocia_OpeningFcn, ...
                   'gui_OutputFcn',  @negocia_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


function negocia_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to negocia (see VARARGIN)

% Choose default command line output for negocia
handles.output = hObject;

set(hObject,'toolbar','figure');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes negocia wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = negocia_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

global ch1;
if (get(hObject,'Value') == get(hObject,'Max'))
ch1=1;
else
ch1=0;
end;

% --- Executes on button press in plot_pushbutton1.
function plot_pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to plot_pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load estudo_dias_opera.mat

global ch1;
global ch2;
global ch3;
global ch4;
global ch5;
global ch6;
global ch7;
global ch8;
global ch9;
global ch10;
global ch11;

axes(handles.axes1)

if ch1==1
plotyy(dias,med_mov_k1,dias,opera_k1,'plot','stem')
end;

if ch2==1
plotyy(dias,med_mov_k2,dias,opera_k2,'plot','stem')
end;

if ch3==1
plotyy(dias,med_mov_k3,dias,opera_k3,'plot','stem')
end;

if ch4==1
plotyy(dias,med_mov_k4,dias,opera_k4,'plot','stem')
end;

if ch5==1
plotyy(dias,med_mov_k5,dias,opera_k5,'plot','stem')
end;

if ch6==1
plotyy(dias,med_mov_k6,dias,opera_k6,'plot','stem')
end;

if ch7==1
plotyy(dias,med_mov_k7,dias,opera_k7,'plot','stem')
end;

if ch8==1
```

```
plotyy(dias,med_mov_k8,dias,opera_k8,'plot','stem')
end;

if ch9==1
plotyy(dias,med_mov_k9,dias,opera_k9,'plot','stem')
end;

if ch10==1
plotyy(dias,med_mov_k10,dias,opera_k10,'plot','stem')
end;

if ch11==1
plotyy(dias,bovespa,dias,opera_bov,'plot','stem')
end;

guidata(hObject, handles);

% --- Executes on button press in clear_pushbutton2.
function clear_pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to clear_pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.axes1,'reset')
guidata(hObject, handles); %updates the handles


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2
global ch2;
if (get(hObject,'Value') == get(hObject,'Max'))
ch2=1;
else
ch2=0;
end;

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3
 global ch3;
if (get(hObject,'Value') == get(hObject,'Max'))
ch3=1;
else
ch3=0;
end;

% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of checkbox4
 global ch4;
if (get(hObject,'Value') == get(hObject,'Max'))
ch4=1;
else
ch4=0;
end;


% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox5

global ch5;
if (get(hObject,'Value') == get(hObject,'Max'))
ch5=1;
else
ch5=0;
end;

% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox6

global ch6;
if (get(hObject,'Value') == get(hObject,'Max'))
ch6=1;
else
ch6=0;
end;

% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7


global ch7;
if (get(hObject,'Value') == get(hObject,'Max'))
ch7=1;
else
ch7=0;
end;


% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of checkbox8


global ch8;
if (get(hObject,'Value') == get(hObject,'Max'))
ch8=1;
else
ch8=0;
end;


% --- Executes on button press in checkbox9.
function checkbox9_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox9


global ch9;
if (get(hObject,'Value') == get(hObject,'Max'))
ch9=1;
else
ch9=0;
end;

% --- Executes on button press in checkbox10.
function checkbox10_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox10

global ch10;
if (get(hObject,'Value') == get(hObject,'Max'))
ch10=1;
else
ch10=0;
end;

% --- Executes on button press in checkbox11.
function checkbox11_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox11


global ch11;
if (get(hObject,'Value') == get(hObject,'Max'))
ch11=1;
else
ch11=0;
end;
```