6

Referência bibliográficas

ARAÚJO G.S. et. al. **Contornando os pressupostos de Black & Scholes:** Aplicação do Modelo de precificação de opções de Duan no mercado brasileiro**.** In: ENANPAD, 27. 2003, Atibaia. Anais... Rio de Janeiro: ANPAD, 2003.

BARBEDO, C.H.S.; ARAÚJO, G.S. e LEMGRUBER, E.F. **Simulação Histórica Filtrada:** Incorporação da Volatilidade ao Modelo Histórico de Cálculo de Risco para Ativos Não-Lineares. Banco Central do Brasil. Trabalho para Discussão nº 94. p. 1-24, 2005.

BARONE-ADESI, G.; ENGLE, R.F. & MANCINI, L. **A GARCH Option Pricing Model with Filtered Historical Simulation (January 2008).** Review of Financial Studies, 2008.

BLACK, F. & SCHOLES, M. **The pricing of options and corporate liabilities.** Journal of Political Economy, 81 (3):637-59. 1973.

BARONI-ADESI, G.; GIANNOPOULOS, K. & VOSPER, L. **VaR without Correlations for Nonlinear Portfolios, Journal of Futures Markets**. Journal of Futures Market. 19 (April), 583-602, 1999.

BELTRAMI, M. **Precificação de Opções sobre Ações por Modelos de Support Vector Regression**. Dissertação de Mestrado. Universidade Federal do Paraná, 2009.

BESSADA, O. e BOVESPA. **Como atuar no mercado de opções**. São Paulo, publicação disponível em www.bovespa.com.br. 2000.

BESSADA, O.; BARBEDO, C. e ARAÚJO, G. **Mercado de Derivativos no Brasil: conceitos, operações e estratégias**. Rio de Janeiro, RJ: Record, 2005.

BM&F – BOLSA DE MERCADORIAS E FUTUROS. Mercados. Ações. Cotações. **Cotações Históricas**. Disponível em /www.bmfbovespa.com.br/>. Acesso em 01 jan 2010.

BM&F – BOLSA DE MERCADORIAS E FUTUROS. Empresas listadas. VALE S.A. **Eventos Corporativos**. Disponível em /www.bmfbovespa.com.br/>. Acesso em 14 jan 2011.

BM&F – BOLSA DE MERCADORIAS E FUTUROS. Mercados. Mercadorias e Futuros. Boletim. Indicadores. Preços Referenciais. **Taxas Referenciais BM&F**. Disponível em /www.bmfbovespa.com.br/>. Acesso em 01 ago 2011.

BOLLERSLEV, T. **Generalized Autoregressive Conditional Heteroscedasticity**. Journal of Econometrics, 31: 307-327, 1986.

BOYLE, P.P. **Options: A Monte Carlo Approach.** Journal of Financial Economics, 4: 323-338, 1997.

CHRISTOFFERSEN, P. and JACOBS, K. ***Which GARCH Model for Option Valuation?*** Management Science, 50:1204-21, 2004.

COX, J.C. & ROSS, S.A. **The valuation of options for althernative stochastic processes**. Journal of Financial Economics, 3: 145-166. 1976

DUAN, J. **The GARCH Option Pricing Model.** Mathematical Finance, 5: 13-32. 1995.

DUAN, J. & SIMONATO, J.G. **Empirical Martingale Simulation for asset prices.** Management Science, 44: 1218-1233, 1998.

ENGLE, R. **Autoregressive Conditional Heteroscedasticity With Estimates Of The Variance Of U.K. Inflation.** Econometrica, 50, 987-1008, 1982.

EXAME. Seu dinheiro. Ações. Notícias. Disponível em: /http://exame.abril.com.br/>. Acesso em 15 nov 2010.

FIA (Futures Industry Association). Trading Volume Statistics. **Annual Volume Survey.** 2010. Disponível em http://www.futuresindustry.org/>. Acesso em 05 de Julho de 2011.

FARHI, M. **Derivativos Financeiros: Hedge, especulação e arbitragem.** Economia e Sociedade, Campinas, (13): 93-114, dez. 1999.

FRANSES, P.H. & van DIJK, D. **Nonlinear Time Series Models in Empirical Finance**. Cambrisge University Press, 2000.

GLOSTEN, L.R.; JAGANNATHAN, R.; RUNKLE, D. E. **On The Relation Between Expected Value and The Volatility of The Nominal Excess Return on Stocks**. Journal of Finance, v. 48, p. 1779-1801, 1993.

GOURIEROUX, C,.A. MONFORT, & A. TROGNON, 1984. **Pseudo Maximum Likelihood Methods: Theory**. Econometrica, 52: 681-700.

HESTON, S., and S. NANDI. 1997. **A Closed-Form GARCH Option Pricing Model.** Review of Financial Studies, 13: 585-626.

HULL, J. 5th Edition, **Fundamentos dos Mercados Futuros e de Opções**, BM&F Brasil. 4. ed. rev. e amp. São Paulo: Bolsa de Mercadorias & Futuros, 2005.

HULL, J. & WHITE, A. **The pricing of options on assets with stochastic volatility.** Journal of Finance, 42: 281-300, 1987.

JUNIOR, A.M.D. **Hedge ótimo de carteiras de opções no Brasil.** Pesquisa Operacional para o Desenvolvimento. , v.1, n.1, p. 9-20, Janeiro a Abril de 2009.

LANARI, C.S.; SOUZA, A.A. e DUQUE, J.C. Desvios em Relação ao Modelos de Black & Scholes: Estudos Relacionados à Volatilidade dos Ativos Subjacentes às Opções. Anais, XIX ENEGEP. Rio de Janeiro, 1999.

LEHAR, A.; SCHEICHER, M. & SCHITTENKOPF, C. (2001). **GARCH vs stochastic volatility. Option pricing and risk management.** Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science", 52. SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, Vienna.

NELDER, J.A. & MEAD, R. **A Simplex method for function minimization.** The computer journal, 1965.

NELSON, D.B. **Conditional heteroskedasticity in asset returns: A new approach**. Econometrica, 59: 347-370. 1991

MERTON, R.C. **Theory of Rational Option Pricing**. Bell Journal of Economics and Management Science, 4, 141-183. 1973.

R Project. Packages. **Neldermead**. Disponível em /http://cran.r-project.org/>. Acesso em 03 nov 2010.

TSAY, RUEY S. **Analysis of Financial Time Series**, John Wiley & Sons, Inc., 2005.

WILMOTT, P. **Frequently asked questions in quantitative finance.** John Wiley & Sons, Ltd., Chichester, 2009.

7
Apêndice

7.1
Apêndice A - Rotinas para calibração do modelo GARCH-SHF com inovações empíricas

```
#-------------------------------------------------------------------------------------#
#### ------------------  PARAM_AGARCH_FREE_REAL ----------------------#### 
#-------------------------------------------------------------------------------------#

param_agarch_free_real = function( param_free )
#    param_agarch_free_real ( param_free ) recupera o verdadeiro valor do parâmetro do GJR GARCH ,
#   param_free são os parâmetros auxiliares que podem variar em todo R^4.
#   Obs: para simplificar, as restrições estão em termos de alpha2/2 = alpha2h

{
param_constr = c(0, 0, 0, 0)
param_constr[1] = exp( param_free[1] )
den = (   1 + exp( param_free[2] ) + exp( param_free[3] ) + exp( param_free[4] )   )
param_constr[2] = exp( param_free[2] ) / den
param_constr[3] = exp( param_free[3] ) / den
param_constr[4] = exp( param_free[4] ) / den
return(param_constr)
}

#-------------------------------------------------------------------------#
#### ----------------------  PARAM_AGARCH_REAL_FREE ----------------------#### 
#-------------------------------------------------------------------------#

param_agarch_real_free= function( param_veri )
#   alpha0, alpha1, delta1, alpha2h  |---> função paramgarch_real_free  |--->  param_free
# Obs: para simplificar, as restrições estão em termos de alpha2/2 = alpha2h

{
param_in = c(0, 0, 0, 0)
alpha0 = param_veri[1]
alpha1 = param_veri[2]
delta1 = param_veri[3]
```

```
alpha2h = param_veri[4]
param_in[1] = log ( alpha0 )
param_in[2] =  log ( alpha1 / (1 - alpha1 - delta1 - alpha2h) )
param_in[3] =  log ( delta1 / (1 - alpha1 - delta1 - alpha2h) )
param_in[4] =  log ( alpha2h / (1 - alpha1 - delta1 - alpha2h) )
return(param_in)
}

#-------------------------------------------------------------------------------------#
#### ----------------------  LOGLIKEFREE_A_GARCH ----------------------#### 
#-------------------------------------------------------------------------------------#

loglikefree_a_garch= function( param)

{

load(file="yyy.Rdata")
T = length(y)
rho0 = param[1]

param_constr_var = param_agarch_free_real( param[2:5] )
alpha0 = param_constr_var[1]
alpha1 = param_constr_var[2]
beta   = param_constr_var[3]
alpha2 = param_constr_var[4] * 2 # > 0 quando o efeito alavancagem existe

#---------------------------VALORES INICIAIS: sigma2_0, e2_0-----------------------------------
e = y - rho0
s2 =( t(e) %*% e)/T
lambda = 0.7
j = 0 : (T-1)
lambdaj = lambda^j

e2 = e^2

sigma21 = lambda^T * s2 + ( 1 - lambda ) *  sum( e2 * lambdaj)
e2[1] = sigma21
#-------------------------------------------------------------------------------

sigma2 = sigma21
loglike = 0
for (i in 1 : T)

{    if (i == 1)
       {d = 0
```

```
            sigma2 = alpha0 + ( alpha1 + alpha2 * d ) * sigma21 + beta * sigma21 # gma2 t-1 a
sigma2 t
            } else{d=as.numeric((e[i-1]) < 0) #  d = 1, quando e(i-1) = 1.
            sigma2 = alpha0 + ( alpha1 + alpha2 * d ) * e2[i-1] + beta * sigma2  # da sigma2 t-1 a
sigma2 t. %remark: nel r.h.s. sigma2 e e2 sono al tempo t-1
            }
          u2 = e2[i] / sigma2
          loglike = loglike - 0.5 * (  log(2*pi) + log(sigma2) + u2  )
        }
        l =  - loglike
        return(l)
        }

        #-------------------------------------------------------------------------------#
        #### ----------------------  LOGLIKE_A_GARCH ----------------------####
        #-------------------------------------------------------------------------------#


        loglike_a_garch= function( param)

        {
        load(file="yyy.Rdata")

        T = length(y)
        rho0 = param[1]

        alpha0 = param[2]
        alpha1 = param[3]
        beta   = param[4]
        alpha2 = param[5] # > 0 quando há efeito de alavancagem

        # VALORES INICIAIS: sigma2_0, e2_0

        e = y - rho0
        s2 =( t(e) %*% e)/T
        lambda = 0.7
        j = 0 : (T-1) # 1xT
        lambdaj = lambda^j

        e2 = e^2

        sigma21 = lambda^T * s2 + ( 1 - lambda ) *  sum( e2 * lambdaj)
        e2[1] = sigma21
```

```
# ---- inicializando a variância.

sigma2 = sigma21

loglike = 0
for (i in 1 : T)

{   if (i == 1)
      {d = 0
      sigma2 = alpha0 + ( alpha1 + alpha2 * d ) * sigma21 + beta * sigma21
      } else{d=as.numeric((e[i-1]) < 0) #  d = 1, quando e(i-1) = 1.
      sigma2 = alpha0 + ( alpha1 + alpha2 * d ) * e2[i-1] + beta * sigma2
      }
   u2 = e2[i] / sigma2
   loglike = loglike - 0.5 * (  log(2*pi) + log(sigma2) + u2  )
}
l =  - loglike #  minimizamos a verossimilhança negativa mas padronizadas para o tamanho
da amostra?
return(l)
}

#-------------------------------------------------------------------------------------#
#### ----------------------  AGARCH_ESTIMATION--------------------####
#-------------------------------------------------------------------------------------#

agarch_estimation=function(thatday, vale_logprice, vale_date)
{

# Amostra: até t0 (a quarta-feira a ser trabalhada na base de opções).
Day0 = find(thatday== vale_date )

num_obs = 800
y = Valeindex$vale_logprice[(Day0-num_obs):Day0]
y=y[2:length(y)]-y[1:(length(y)-1)]

save(y,file="yyy.Rdata")

# Estimação do modelo GARCH sob medida física
start_val=c(0.05, 0.05,  0.1,  0.8,  0.01)

# RESTRITO
sol = fminsearch( loglikefree_a_garch, c(start_val[1], param_agarch_real_free(start_val[2:5]
* c(1, 1, 1, 0.5)))  , optimset(TolFun=5e-6,MaxFunEvals=1500))
theta=sol$x
theta = c(theta[1],param_agarch_free_real((theta[2:5]) * c(1, 1, 1, 2)))
```

```
save(theta,file="solution.Rdata")
}

#-------------------------------------------------------------------------------------#
#### ------------------- AGARCH_FILTERING_AUX ------------------#### 
#-------------------------------------------------------------------------------------#

agarch_filtering_auxVariables=function (thatDay, vale_logprice, vale_date, theta, dataWedFHS)
{
Day0 = find(thatDay== vale_date)
num_obs = 800

y=Valeindex$vale_logprice[(Day0-num_obs):Day0]
y=y[2:length(y)]-y[1:(length(y)-1)]

### Volatilidades estimadas

T = length(y)
sig2=matrix(nrow=T,ncol=1)
echapeu=y-paramGarchP[i_fhs,1]
echapeu2=echapeu^2

s2 =( t(echapeu) %*% echapeu)/T
lambda = 0.7
j = 0 : (T-1)
lambdaj = lambda^j
sigma21 = lambda^T * s2 + ( 1 - lambda ) *  sum( echapeu2 * lambdaj)

teste<-as.numeric(echapeu<0)
d=lag(teste)

for (i in 1 :T)
{
    if (i == 1)
      {d0 = 0
       sig2[i] = paramGarchP[i_fhs,2]+ ( paramGarchP[i_fhs,3] + paramGarchP[i_fhs,5] * d0 ) * sigma21 + paramGarchP[i_fhs,4] * sigma21
       } else{sig2[i] = paramGarchP[i_fhs,2]+ (paramGarchP[i_fhs,3] + paramGarchP[i_fhs,5] * d[i] ) * echapeu2[(i-1)] + paramGarchP[i_fhs,4] * sig2[(i-1)]
            }
}
sig=sqrt(sig2)

z_fhs = y[2:length(y)] - paramGarchP[i_fhs,1]
```

```
sig   = sqrt( sig2[2:length(sig2)] )
z_fhs = z_fhs/ sig
longest_maturity = dataWedFHS[nrow(dataWedFHS), 6]
p_0     = exp(Valeindex$vale_logprice[Day0])
r_free  = mean(dataWedFHS[ ,2]) /100 /365
r_div   = dataWedFHS[1,7] /100 /365
sig2T_0 = sig2[nrow(sig2)]
sigT_0   = sqrt(sig2T_0)
z2T_0   = z_fhs[length(z_fhs)]^2
zT_0    = z_fhs[length(z_fhs)]

save(list=c("p_0","sig2T_0","sigT_0","z2T_0","zT_0","r_free","r_div","longest_maturity"),file="
aux_variables.Rdata")

z_fhs = c(mean(z_fhs), z_fhs)
save(z_fhs,file="filtered_innovations.Rdata")
}

#-------------------------------------------------------------------------------------------#
#### ---------- FHS_GARCH_OPTIONOUTMON_DIV_FUN ---------- ####
#-------------------------------------------------------------------------------------------#


fhs_garch_optionOutMon_div_fun=function( param_free )

# fhs_garch_optionPut_div_fun( param ) gives GARCH call option prices computed using
MC simulations and filtered innovations.

{
num_sim = 20000

load(file="aux_variables.Rdata")     # load: longest_maturity p_0   r_free   r_div   sig2T_0
z2T_0  zT_0
load(file="filtered_innovations.Rdata") # load: z_fhs
sig2_sim  =  matrix(NaN,nrow=num_sim,  longest_maturity)  #  cada  linha=  um  caminho
simulado

# ---- VALUES AT TIME T=0 ----
sig2T = matrix(1,nrow= num_sim, ncol=1 ) * sig2T_0
z2T   = matrix(1,nrow= num_sim, ncol=1 ) * z2T_0

I_levT = as.numeric(zT_0 < 0) # = 1, when zT_0 < 0.
# p = p_0 ; % closing S&P 500 index price.
```

```
        set.seed(123)
        time_boot                                                                    =
matrix(sample(length(z_fhs),sum(num_sim*longest_maturity),replace='true'),num_sim,
longest_maturity)

        # --- plain MC/bootstrap ---
        z_sim = matrix(z_fhs[time_boot], ncol=longest_maturity)

        I_lev  = matrix(as.numeric(z_sim < 0),ncol=longest_maturity)
        z2_sim = z_sim^2

        param_real = param_agarch_free_real(param_free)
        #param_free)   parameter in daily base for variance in percent.
        alpha0 = param_real[1]
        alpha1 = param_real[2]
        beta   = param_real[3]
        alpha2 = param_real[4]*2   #as we imposed the constraint on alpha2/2

        for (i in 1 : longest_maturity) # we are in T. T+1, T+2, ... are possible date for maturity of
options.
        {   if (i == 1){sig2_sim[,i]=alpha0 +( alpha1*z2T + alpha2*I_levT * z2T + beta ) * sig2T
              } else {sig2_sim[,i]=alpha0 +( alpha1*z2_sim[,i-1] + alpha2*I_lev[,i-1] * z2_sim[,i-1] +
beta )*sig2_sim[,i-1]
                    }
        }

        sig2_sim = sig2_sim
        sig_sim  = sqrt(sig2_sim)
        logret_sim    = (repmat(taxasdi[1:longest_maturity,i_fhs],num_sim,1) - sig2_sim/2) + sig_sim
* z_sim   #  daily logret non in percent.
        logprice_sim = t(apply(logret_sim, 1, cumsum))
        price_sim  =  p_0  *  exp(logprice_sim)   #  num_sim  x  longest_maturity.  each  row  =  one
simulated path.

        load(file="fhs_calibration_work.Rdata")  # load: dataWedFHS  maturityDay

        drift_correction  =  p_0*exp(  (taxasdi[maturityDay  ,i_fhs])  *  maturityDay  )  /  colMeans(
price_sim[, maturityDay] )
        price_sim[, maturityDay] = price_sim[, maturityDay] * repmat(drift_correction, num_sim, 1)
        K_all = dataWedFHS[, 5]
        maturity_all = dataWedFHS[, 6]
        put_prices_tmp  = c(NaN)
        call_prices_tmp = c(NaN)
```

```
for (T_tmp in maturityDay)
{
  # N.B.: the number of strike prices changes with maturity and database

  K_tmp = K_all[ find(maturity_all == T_tmp) ]  # all the strike prices associated to that maturity.
  num_K = length(K_tmp)  # number of strike prices or options for that maturity.

  #  C A L L

  call_tmp = repmat(price_sim[, T_tmp], 1, num_K) - repmat(K_tmp, num_sim, 1) # num_sim x num_K
  call_tmp[ find(call_tmp < 0) ] = 0 # values of S_T - K < 0 are set to 0.
  call_tmp = colMeans(call_tmp) # mean along each column (recall, one row = one simulated path)
  # OR when r_free = mean(r_free)
  call_tmp = t(call_tmp) * exp(-taxasdi[T_tmp,i_fhs] * T_tmp)
  call_prices_tmp = c(call_prices_tmp, call_tmp) # column vector of call prices
}

call_all = call_prices_tmp[2:length(call_prices_tmp)]

fhsOption=call_all
return(fhsOption)
}



# ------------------------------------------------------------------------------------------#
#### ------------ FHS_OPTION_OUT_MON_CALIBRATED ----------------- ####
# ------------------------------------------------------------------------------------------#


fhs_optionOutMon_calibrated=function(dataWedFHS, thatDay, vale_date, vale_logprice, param_start)
{
currentDay = find( thatDay == vale_date )

maturityDay=as.vector(by(dataWedFHS$maturity,dataWedFHS$maturity, mean))
save(list=c("dataWedFHS","maturityDay"),file="fhs_calibration_work.Rdata")

#    C A L I B R A T I O N
calibration=fminsearch(fhs_garch_option_error_OutMon, param_agarch_real_free(param_start) , optimset(  Display='iter',TolFun=0.1, MaxFunEvals=1500))
```

```
save(calibration,file="calibration.Rdata")
param_free=as.vector(calibration$x)

# Calibrated GARCH parameter
paramFHS= param_agarch_free_real(param_free) * c(1, 1, 1, 2)
callFHS = 0
save(paramFHS,file=paste("paramFHS",i_fhs,".Rdata",sep =""))
return(paramFHS)
}

# ------------------------------------------------------------------------------------------#
#### ---------- FHS_GARCH_OPTION_ERROR_OUT_MON ----------#
# ------------------------------------------------------------------------------------------#

fhs_garch_option_error_OutMon=function( param_free )
{

# OBSERVED PRICES
load(file="fhs_calibration_work.Rdata")  # load: dataWedFHS  maturityDay
option_market = dataWedFHS[,4] # i.e. out-of-the-money calls and puts.

# THEORETICAL PRICES
option_model = fhs_garch_optionOutMon_div_fun( param_free ) # WITH DIVIDENDS

results=cbind(option_market,option_model, moneynessWed, dataWedFHS)
save(results,file=paste("ResultsIS",i_fhs,".Rdata",sep =""))

# MSE to be minimized:
option_error = sum(abs(option_model - option_market)^2)^(1/2)
RMSEFHS=option_error / sqrt(length(option_model))

save(RMSEFHS,file=paste("RMSEFHS",i_fhs,".Rdata",sep =""))

return(option_error)
}

# ------------------------------------------------------------------------------#
#### ----------  FHS_CALIBRATION ----------####
# ------------------------------------------------------------------------------#

taxasdi<-read.csv("C:/Documents                                                    and
Settings/nlgomes/Desktop/Dissertacao/Bases/matriz_taxas.csv",sep=";", as.is=1, header=FALSE)
taxasdi=taxasdi/100/360
```

```
data_base_Weds<-read.csv("C:/Documents and Settings/nlgomes/Desktop/Dissertacao/Bases/data_base_Weds.csv",sep=";", header=TRUE, as.is=1)

wednesday_noholidays=read.csv("C:/Documents and Settings/nlgomes/Desktop/Dissertacao/Bases/wednesday_noholidays.csv",sep=";", header=TRUE, as.is=1)

Valeindex <-read.csv("C:/Documents and Settings/nlgomes/Desktop/Dissertacao/Bases/ln_vale5.csv",sep=";", header=TRUE, as.is=1)

## ---------- FIT the GARCH model, every Wednesday ----------
 paramGarchP = matrix(NaN, length(wednesday_noholidays$WedFeasible), 5) # calibrated GARCH parameter
 convP = matrix(NaN, length(wednesday_noholidays$WedFeasible), 1)


for (i_garch in 1:50)
 {
  #  FIT THE GARCH MODEL to historical returns:
    agarch_estimation(wednesday_noholidays$WedFeasible[i_garch], Valeindex$vale_logprice, Valeindex$vale_date)
load(file="solution.Rdata")
paramGarchP[i_garch,]= theta
}
 # save("paramGarchP" , file="res_estimation.Rdata")   # save estimated parameter
# ------------------------------------------------------------------------

#load(file="res_estimation.Rdata")  # LOAD: paramGarchP[1,]

paramFHS = matrix(NaN, length(wednesday_noholidays$WedFeasible), 4)
RMSEFHS = matrix(NaN, length(wednesday_noholidays$WedFeasible),1)
save(paramFHS,file="paramFHS.Rdata")

for (i_fhs in 37:50) #59   64   88   104   105   107   144  131
{

   param_start = paramGarchP[i_fhs,2:5]* c(1, 1, 1, 0.5)
   iWed =  find(wednesday_noholidays$WedFeasible[i_fhs] == data_base_Weds$date)
   dataWedFHS  = data_base_Weds[iWed,]
   dataWedFHS =dataWedFHS[find(dataWedFHS$optionOutMon>0.10 & dataWedFHS$totneg>10 ),]
```

```
moneynessWed = dataWedFHS$price/ dataWedFHS$strike*exp(-
taxasdi[dataWedFHS$maturity,i_fhs]*dataWedFHS$maturity)

agarch_filtering_auxVariables(wednesday_noholidays$WedFeasible[i_fhs],
Valeindex$vale_logprice, Valeindex$vale_date, paramGarchP[i_fhs,], dataWedFHS)

fhs_optionOutMon_calibrated(dataWedFHS, wednesday_noholidays$WedFeasible[i_fhs],
Valeindex$vale_date, Valeindex$vale_logprice, param_start)

}
```

7.2
Apêndice B - Restrições impostas aos parâmetros GJR GARCH

Para atender as restrições impostas nos parâmetros do modelo GJR GARCH e evitar restrições diretas na função minimizada na calibração, foram utilizados parâmetros auxiliares no lugar dos parâmetros. Para isso, faz-se necessária a transformação abaixo nos valores iniciais dos parâmetros GJR GARCH sobre retornos afim de se obter os valores iniciais dos parâmetros auxiliares a serem estimados.

Os parâmetros estimados foram nomeados como param_free[i].

$$param\_free[1] = \ln(\alpha_0)$$

$$param\_free[2] = \ln\left(\frac{\alpha_1}{1-\alpha_1-\beta-\alpha_2}\right)$$

$$param\_free[3] = \ln\left(\frac{\beta}{1-\alpha_1-\beta-\alpha_2}\right)$$

$$param\_free[4] = \ln\left(\frac{\alpha_2}{1-\alpha_1-\beta-\alpha_2}\right)$$

Após a calibração dos parâmetros acima, os parâmetros GJR GARCH de apreçamento são recuperados da seguinte forma:

$$\alpha_0 = \exp(param\_free[1])$$
$$\alpha_1 = \exp(param\_free[2]) / den$$
$$\beta = \exp(param\_free[3]) / den$$
$$\alpha_2 = \exp(param\_free[4]) / den$$

$$den = 1 + \exp(param\_free[2]) + \exp(param\_free[3]) + \exp(param\_free[4])$$

7.3
Apêndice C - Algoritmo fminsearch

A função fminsearch encontrada em softwares como Matlab e R Project foi utilizada como ferramenta para minimização do erro de apreçamento das opções. O algoritmo Nelder-Mead é utilizado como instrumento para a busca do mínimo irrestrito de uma dada função custo de n variáveis, sem a necessidade do cálculo de derivadas da mesma. O algoritmo se baseia na atualização de um simplex formado por um conjunto de k ≥ n+1 vértices, em que cada vértice está associado a um ponto e um valor da função.

A função possui os seguintes argumento:

fminsearch(fun = NULL, x0 = NULL, options = NULL)

Argumentos:

fun - função custo

x0 – Vetor numérico de valores iniciais (tamanho n).

options - lista de opções de otimização que definem o comportamento do fminsearch:

MaxIter – Número máximo de iterações. Default =200 * n

MaxFunEvals – Número máximo de avaliações da função custo. Default =200*n

TolFun – Tolerância absoluta no valor da função. Default=1.e-4

TolX – Tolerância absoluta no tamanho do simples. Default=1.e-4

Display – Nível de detalhe

OutputFcn – Função output ou lista de funções output chamadas no final de cada iteração. Default=NULL

PlotFcns – função ou lista de funções plot chamadas no final de cada iteração. Default=empty.

O critério de parada da função considera:

Ssize – Tamanho atual do simplex

Shiftfv – Diferença do valor absoluto entre o maior e menor vértice

O usuário define valores de tolerância do TolX e TolFun (Default=1e-4). Caso ambos ssize < TolX e shiftfv < TolFun sejam satisfeitos, as iterações são interrompidas.

7.4

Apêndice D – Resultados das calibrações

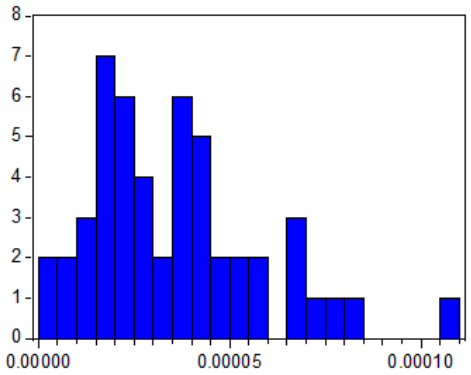

Figura 23: Histograma das estimativas de α_0^*

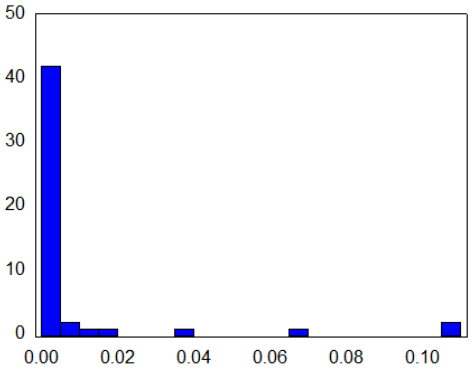

Figura 24: Histograma das estimativas de α_1^*

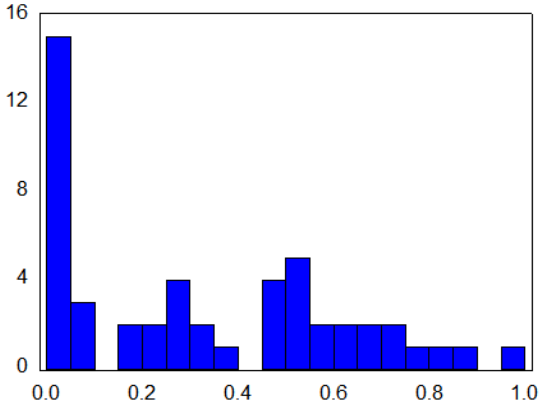

Figura 25: Histograma das estimativas de $\beta*$

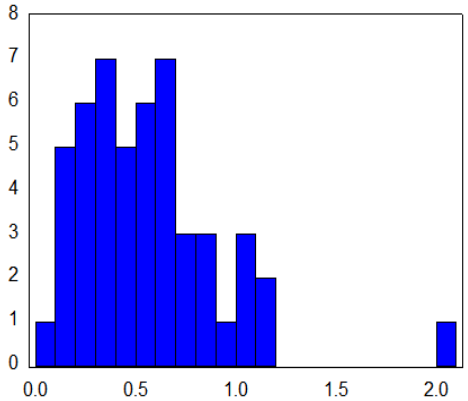

Figura 26: Histograma das estimativas de α_2^*

Tabela 12: MAPE dos modelos de apreçamento dentro da amostra segundo *moneyness* e prazo de vencimento, 2010.

Prazo de vencimento / *Moneyness*	Modelo	Menor que 30 dias	30 a 59 dias	60 dias ou mais	Total
Menor que 0,85	**B&S**		229,1%	150,8%	171,2%
	GARCH-SHF		4,7%	7,7%	6,9%
	GARCH-NORM		6,9%	8,8%	8,3%
	N		19	54	73
0,85 |- até 1	**B&S**	77,9%	67,6%	44,7%	59,6%
	GARCH-SHF	8,3%	3,6%	3,4%	4,4%
	GARCH-NORM	9,0%	4,2%	4,6%	5,2%
	N	78	171	186	435
1,00 |- até 1.15	**B&S**	6,8%	6,0%	7,1%	6,5%
	GARCH-SHF	3,6%	2,3%	2,5%	2,9%
	GARCH-NORM	2,9%	2,5%	2,5%	2,7%
	N	138	131	64	333
1,15 ou mais	**B&S**	1,3%	1,8%	2,3%	1,5%
	GARCH-SHF	1,3%	1,8%	2,1%	1,5%
	GARCH-NORM	1,3%	1,5%	1,5%	1,4%
	N	147	89	11	247
Total	**B&S**	19,9%	41,1%	53,8%	37,7%
	GARCH-SHF	3,7%	2,9%	3,9%	3,4%
	GARCH-NORM	3,6%	3,2%	4,8%	3,8%
	N	363	410	315	1088

Nota 1: B&S - Modelo de B&S alimentado com a volatilidade EWMA; GARCH-SHF - Modelo GARCH-SHF calibrado com inovações empíricas; GARCH-NORM - Modelo GARCH calibrado com inovações normais.

Nota 2: Os resultados ilustrados acima para os modelos GARCH-SHF e GARCH-NORM resultam da calibração dos parâmetros visando a minimização do MAPE ao invés da REQM.