IMPLIED VOLATILITY SURFACE AND MODEL OF VIX

RAN ZHAO

ABSTRACT.

1. INTRODUCTION

Black [2] first introduced the European futures options pricing based on Black-Scholes framework [3]. The common assumption for the process followed by future price F in risk-neutral framework is

$$dF = \sigma F dW, \tag{1}$$

where σ is a constant and W is the Wiener process. Similar to a non-dividend-paying stock, the differential equation satisfied by a derivative dependent on a futures price is

$$\frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial F^2}\sigma^2 F^2 = rf.$$

Then the European call price c and the European put price p for the futures option are given by following equations,

$$c = e^{-rT}[F_0 N(d_1) - KN(d_2)] \tag{2}$$

$$p = e^{-rT}[KN(-d_2) - F_0 N(-d_1)] \tag{3}$$

where

$$d_1 = \frac{\log(F_0/K) + \sigma^2 T/2}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\log(F_0/K) - \sigma^2 T/2}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Following the Black-Scholes framework, more recent researches come up with stochastic volatility (SV) model [5, 1, 4], where the volatility surface (w.r.t. different terms and strikes) are fitted better to the empirical observations. The SV modelings are motivated by the intermittency and clustering feature of volatility from underlying dynamics. A common stochastic volatility process satisfies the SDE:

$$dS_t = \mu_t S_t dt + \sqrt{v_t} S_t dW_1 dv_t = \alpha(S_t, v_t, t)dt + \eta\beta(S_t, v_t, t)\sqrt{v_t}dW_2$$

where $\mathbb{E}[dW_1 dW_2] = \rho dt$. μ_t is the (deterministic) instantaneous drift of stock price returns, η is the volatility of volatility and ρ is the correlation between random stock price returns and changes in v_t. $W1$ and W_2 are Wiener processes.

The volatility drift and vol of vol term are in most generic functional forms and can determine by time (t), instantaneous volatility level (v_t) and the spot price (S_t). Note that the SV setup ensures the standard time-dependent volatility version of the Black-Scholes formula may be retrieved in the limit $\eta \to 0$.

Particularly, S&P500 index returns, shown in Figure 1, reinforce the stochastic volatility and volatility mean-reverting characteristics. More importantly, the "volatility" of the S&P500 index is tradable. The VIX index represents the volatility of SPX in a precise way, and CBOE has listed futures on VIX index since 2004 and VIX option in 2006.

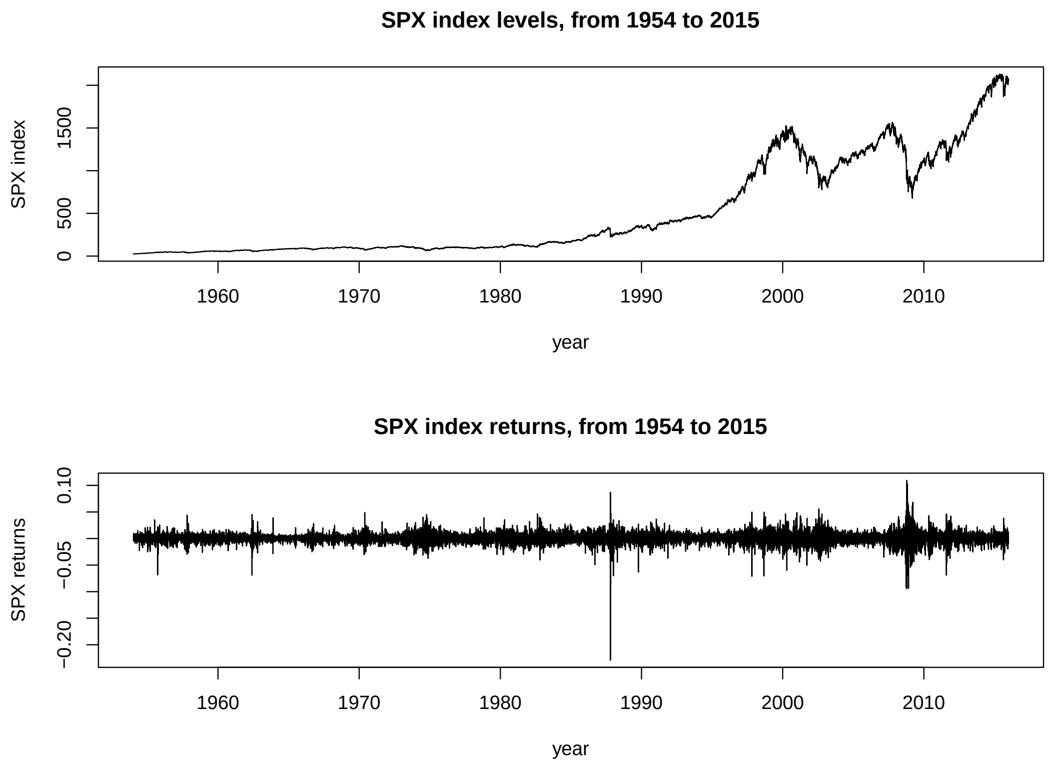


Figure 1 The SPX index levels and return on daily basis. Time period is from 1954 to 2015.

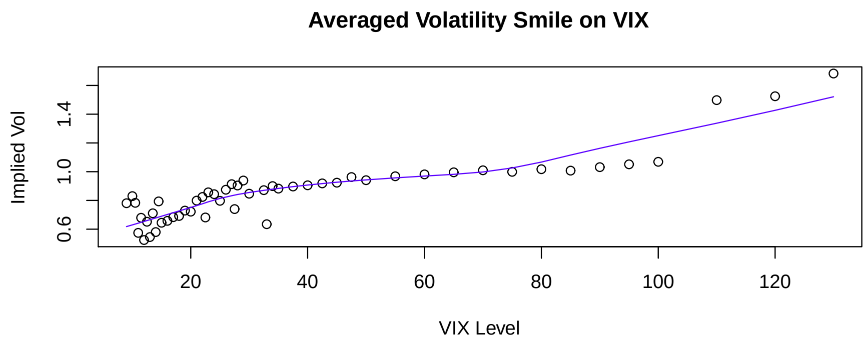
The rest of the paper lays out as following: Section 2 presents the observations and findings on implied volatility from VIX options. Section 3 conjectures a two-factor VIX model and discusses the joint VIX model with SPX dynamics.

2. VIX IMPLIED VOLATILITY

The option data from OptionMetrics contains daily standardized VIX option bid/offer price, Delta of the option and other information including expiration and strike. The forward price, here the underlying future price, is interpolated from OptionMetrics' standard volatility surface using log-linear interpolation. The reasoning of selecting log-linear interpolation is that the future price is positively correlated with discount factor, and we assume a flat-forward interpolation keeps most of properties from term structure. The zero yield curve is also from OptionMetrics database.

The VIX future has multiplier of 1000, so the strike price of options are in magnitude of 1000 times VIX. To back-out the implied volatility of VIX options, we use the Black model as specified in 2. The market premium is calculated from the average of best bid and best offer price. The underlying future price is interpolated log-linearly from standard option volatility, and the interest rate is interpolated linearly from the term structure of zero yield.

3. MODELING VIX AND SPX JOINTLY



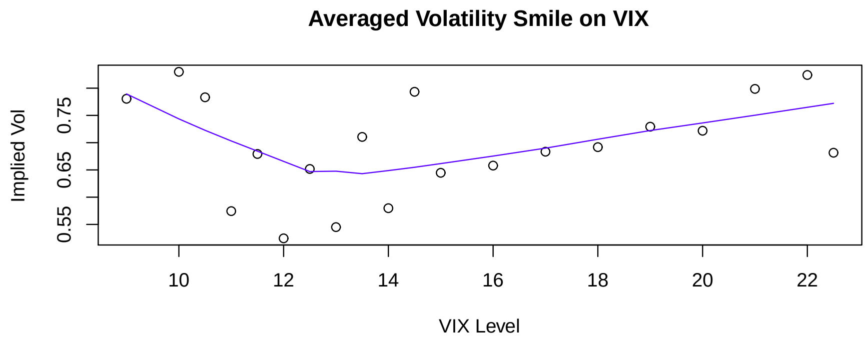


Figure 2 The averaged VIX option implied volatility over different terms. The graph plots the implied volatility dynamic on various strikes. The blue line is the fitted implied volatility smile by lowess regression. The graph on the top has strikes (of VIX) from 5 to 130. The graph on the bottom has strikes from 5 to 22.

References

[1] Bates, David S. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *Review of Financial Studies*, 9:69–107, 1996.

[2] Black, Fischer. the pricing of commodity contracts. *Journal of Financial Economics*, 3:167–179, 1976.

[3] Black, Fischer and Scholes, Myron. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–654, 1973.

[4] Duffie, Darrell, Jun Pun, and Kenneth Singleton. Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, 68:1343–1376, 2000.

[5] Heston, Steven L. A closed-form soultion for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–343, 1993.

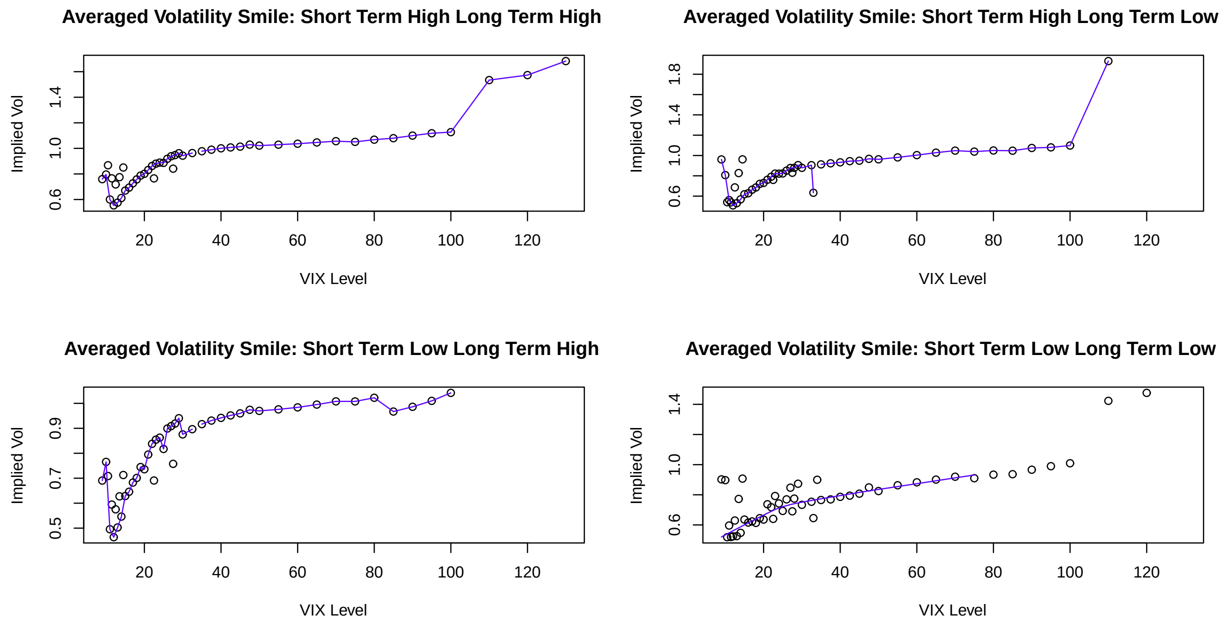


Figure 3 The averaged VIX option implied volatility over different terms, when the short/long term implied volatilities are high/low. Four graphs display the combination of the short- and long-term VIX. Each graph plots the implied volatility dynamic on various strikes.

APPENDIX: CODE

```
setwd('C:\\Users\\ranzhao\\Documents\\Empirical-Asset-Pricing\\Assignment 7')
setwd('D:\\PhD FE\\Empirical-Asset-Pricing\\Assignment 7')
setwd('D:\\Empirical-Asset-Pricing\\Assignment 7')

spx_index_values = read.csv('spx_index_values.csv', header = TRUE)
par(mfrow=c(2,1))
plot(as.Date(as.character(spx_index_values$Date), "%m/%d/%Y"), spx_index_values$SPX.
    Index, type='l',
      main='SPX index levels, from 1954 to 2015',
      xlab='year', ylab='SPX index')

# calculate the return series
spx_index_values$Return = rep(0, dim(spx_index_values)[1])
spx_index_values$Return[2:length(spx_index_values$Return)] =
  log(spx_index_values$SPX.Index[2:length(spx_index_values$SPX.Index)] /
        spx_index_values$SPX.Index[1:(length(spx_index_values$SPX.Index)-1)])
data.length = length(spx_index_values$Return)

plot(as.Date(as.character(spx_index_values$Date), "%m/%d/%Y"), spx_index_values$Return
    , type='l',
      main='SPX index returns, from 1954 to 2015',
      xlab='year', ylab='SPX returns')

require(data.table)
option.data = fread('VIXoptions.csv', header = T, sep = ',')
implied.data = fread('VIXoptionsStd.csv', header = T, sep = ',')
ir.data = fread('zeroyieldcurve.csv', header = T)
```

```
option.data = as.data.frame(option.data)
implied.data = as.data.frame(implied.data)
ir.data = as.data.frame(ir.data)

option.data = option.data[,c("date","exdate","cp_flag","strike_price","best_bid","best
    _offer","delta")]
option.data$days = as.numeric(as.Date(as.character(option.data$exdate), format="%Y%m%d
    ")-as.Date(as.character(option.data$date), format="%Y%m%d"))
option.data$T = option.data$days/360

option.data = option.data[!is.na(option.data$delta),]
implied.data = implied.data[implied.data$cp_flag=="C",c("date","days","forward_price")
    ]

# interpolation future prices and interest rate
option.data$F0 = 0
option.data$r = 0

all.dates = unique(option.data$date)
for (i in all.dates){ # slow
  all.days = unique(option.data$days[option.data$date==i])
  for (j in all.days){
    ix0 = implied.data$days[implied.data$date==i]
    iy0 = implied.data$forward_price[implied.data$date==i]
    option.data$F0[option.data$date==i&option.data$days==j] = linear.inter(ix0, iy0, j
        , "log")
    # handle missing data
    ir.ix0 = ir.data$days[ir.data$date==i]
    ir.iy0 = ir.data$rate[ir.data$date==i]/100
    if (length(ir.ix0)==0){
      ir.ix0 = ir.data$days[ir.data$date==i.last]
      ir.iy0 = ir.data$rate[ir.data$date==i.last]/100
    }
    option.data$r[option.data$date==i&option.data$days==j] = linear.inter(ir.ix0, ir.
        iy0, j, "linear")
  }
  i.last = i
}

# calculate the BS implied volatility
option.data$BS_iv = 0
for (k in 1:length(option.data$date)){ # very slow
  option.data$BS_iv[k] = bs.iv(option.data$F0[k],
                               option.data$strike_price[k]/1000,
                               option.data$T[k],
                               option.data$r[k],
                               0.5*(option.data$best_bid[k]+option.data$best_offer[k])
                                   ,
                               option.data$cp_flag[k])
}
# option.data$BS_iv = lapply(cbind(option.data$F0, option.data$strike_price/1000,
    option.data$T, option.data$r, 0.5*(option.data$best_bid+option.data$best_offer),
    option.data$cp_flag), bs.iv)

option.data = option.data[!is.na(option.data$BS_iv),]
all.strikes = sort(unique(option.data$strike_price))
vix.vol.smile = rep(0, length(all.strikes))
```

```
for (s in 1:length(all.strikes)){
  vix.vol.smile[s] = mean(option.data$BS_iv[option.data$strike_price==all.strikes[s]])
}

par(mfrow=c(2,1))
plot(all.strikes/1000, vix.vol.smile, type='p', main='Averaged Volatility Smile on VIX
    ',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes/1000,vix.vol.smile), col="blue") # lowess line (x,y)

plot(all.strikes[1:20]/1000, vix.vol.smile[1:20], type='p', main='Averaged Volatility
    Smile on VIX',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes[1:20]/1000,vix.vol.smile[1:20]), col="blue") # lowess line (x
    ,y)

short.term.vol.thres = mean(option.data$BS_iv[option.data$days<=60])
long.term.vol.thres = mean(option.data$BS_iv[option.data$days>=120])

short.term.high.long.term.high = c()
short.term.high.long.term.low = c()
short.term.low.long.term.high = c()
short.term.low.long.term.low = c()

for (i in all.dates){
  short.term.vol = mean(option.data$BS_iv[option.data$date == i & option.data$days
      <=60])
  long.term.vol = mean(option.data$BS_iv[option.data$date == i & option.data$days
      >=120])
  if (short.term.vol >= short.term.vol.thres & long.term.vol >= long.term.vol.thres){
    short.term.high.long.term.high = rbind(short.term.high.long.term.high, option.data
        [option.data$date == i,])
  } else if (short.term.vol >= short.term.vol.thres & long.term.vol < long.term.vol.
      thres){
    short.term.high.long.term.low = rbind(short.term.high.long.term.low, option.data[
        option.data$date == i,])
  } else if (short.term.vol < short.term.vol.thres & long.term.vol >= long.term.vol.
      thres){
    short.term.low.long.term.high = rbind(short.term.low.long.term.high, option.data[
        option.data$date == i,])
  } else {
    short.term.low.long.term.low = rbind(short.term.low.long.term.low, option.data[
        option.data$date == i,])
  }
}

vix.vol.smile.short.term.high.long.term.high = rep(0, length(all.strikes))
vix.vol.smile.short.term.high.long.term.low = rep(0, length(all.strikes))
vix.vol.smile.short.term.low.long.term.high = rep(0, length(all.strikes))
vix.vol.smile.short.term.low.long.term.low = rep(0, length(all.strikes))

for (s in 1:length(all.strikes)){
  vix.vol.smile.short.term.high.long.term.high[s] = mean(short.term.high.long.term.
      high$BS_iv[short.term.high.long.term.high$strike_price==all.strikes[s]])
  vix.vol.smile.short.term.high.long.term.low[s] = mean(short.term.high.long.term.low$
      BS_iv[short.term.high.long.term.low$strike_price==all.strikes[s]])
  vix.vol.smile.short.term.low.long.term.high[s] = mean(short.term.low.long.term.high$
      BS_iv[short.term.low.long.term.high$strike_price==all.strikes[s]])
  vix.vol.smile.short.term.low.long.term.low[s] = mean(short.term.low.long.term.low$BS
      _iv[short.term.low.long.term.low$strike_price==all.strikes[s]])
```

```
}
par(mfrow=c(2,2))
plot(all.strikes/1000, vix.vol.smile.short.term.high.long.term.high, type='p', main='Averaged Volatility Smile: Short Term High Long Term High',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes/1000,vix.vol.smile.short.term.high.long.term.high), col="blue") # lowess line (x,y)

plot(all.strikes/1000, vix.vol.smile.short.term.high.long.term.low, type='p', main='Averaged Volatility Smile: Short Term High Long Term Low',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes/1000,vix.vol.smile.short.term.high.long.term.low), col="blue") # lowess line (x,y)

plot(all.strikes/1000, vix.vol.smile.short.term.low.long.term.high, type='p', main='Averaged Volatility Smile: Short Term Low Long Term High',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes/1000,vix.vol.smile.short.term.low.long.term.high), col="blue") # lowess line (x,y)

plot(all.strikes/1000, vix.vol.smile.short.term.low.long.term.low, type='p', main='Averaged Volatility Smile: Short Term Low Long Term Low',xlab='VIX Level',ylab='Implied Vol')
lines(lowess(all.strikes/1000,vix.vol.smile.short.term.low.long.term.low), col="blue") # lowess line (x,y)

linear.inter = function(ix0,iy0,ix,inter.method="log"){
# log-linear and linear interpolation function
    order = sort(ix0, index.return=TRUE)
  x0 = order$x
  y0 = iy0[order$ix]
  n = length(ix0)
  if (ix < min(ix0)){
    ix1 = ix0[1]
    ix2 = ix0[2]
    iy1 = iy0[1]
    iy2 = iy0[2]
  } else if (ix>max(ix0)){
    ix1 = ix0[n-1]
    ix2 = ix0[n]
    iy1 = iy0[n-1]
    iy2 = iy0[n]
  } else {
    ix1 = ix0[max(which(ix>=ix0))]
    ix2 = ix0[min(which(ix<=ix0))]
    iy1 = iy0[max(which(ix>=ix0))]
    iy2 = iy0[min(which(ix<=ix0))]
  }
  if (ix1 == ix2){
    return(iy1)
  }
  else {
    if (inter.method == "log"){
      iy = (ix-ix1)/(ix2-ix1)*log(iy2) + (ix2-ix)/(ix2-ix1)*log(iy1)
      return(exp(iy))
    } else if (inter.method == "linear") {
      iy = (ix-ix1)/(ix2-ix1)*iy2 + (ix2-ix)/(ix2-ix1)*iy1
      return(iy)
```

```
    }
  }
}


bs.iv = function(S, K, T, r, market, type){
# calculate Black-Scholes implied volatility
  sig <- 0.20
  sig.up <- 2
  sig.down <- 0.001
  count <- 0
  err <- BS(S, K, T, r, sig, type) - market

  ## repeat until error is sufficiently small or counter hits 1000
  while(abs(err) > 0.0001 && count<3000){
    if(err < 0){
      sig.down <- sig
      sig <- (sig.up + sig)/2
    }else{
      sig.up <- sig
      sig <- (sig.down + sig)/2
    }
    err <- BS(S, K, T, r, sig, type) - market
    count <- count + 1
  }

  ## return NA if counter hit 1000
  if(count==3000){
    return(NA)
  }else{
    return(sig)
  }
}

BS = function(S, K, T, r, sig, type="C"){
# calculation option price using Black-Scholes model
    d1 <- (log(S/K) + (r + sig^2/2)*T) / (sig*sqrt(T))
    d2 <- d1 - sig*sqrt(T)
    if(type=="C"){
      value <- S*exp(-r*T)*pnorm(d1) - K*exp(-r*T)*pnorm(d2)
    }
    if(type=="P"){
      value <- K*exp(-r*T)*pnorm(-d2) - S*exp(-r*T)*pnorm(-d1)
    }
    return(value)
  }
```

assignment7.R