*# S = Stock price.*

*# K = Strike price of option.*

*# r = Risk-free interest rate.*

*# T = Time to expiration in years.*

*# cm = call price*

*# σ = Volatility of the relative price change of the underlying stock price.*

*# N(x) = The cumulative normal distribution function.*

***#for call***

*import* math

*import* numpy *as* np

*import* scipy.stats *as* si

*from* scipy.optimize *import* fmin

S = 45

K = 40

r = 0.05

T = 0.5

cm = 6.86

def **NORMSDIST**(*x*):

NORMSDIST = si.norm.cdf(*x*,0.0,1.0)

*return*(NORMSDIST)

def **ImpliedVolatilityPut**(*s*):

d1 = ( (np.log(S/K)+(r+0.5\**s*[0]\*\*2)\*T) / (*s*[0]\*np.sqrt(T)) )

d2 = ( (np.log(S/K)+(r-0.5\**s*[0]\*\*2)\*T) / (*s*[0]\*np.sqrt(T)) )

of = ( K\*np.exp(-r\*T)\*NORMSDIST(-d2) - S\*NORMSDIST(-d1) ) - cm

val = of\*\*2

print("[σ]=",*s*,", Object Function Value:", val)

*return*(val)

s = fmin(ImpliedVolatilityPut, [0.1])

***#for put***

*import* math

*from* scipy.optimize *import* fmin

S = 45

K = 40

r = 0.05

T = 0.5

pm = 0.88

def **ImpliedVolatilityPut**(*s*):

d1 = ( (np.log(S/K)+(r+0.5\**s*[0]\*\*2)\*T) / (*s*[0]\*np.sqrt(T)) )

d2 = ( (np.log(S/K)+(r-0.5\**s*[0]\*\*2)\*T) / (*s*[0]\*np.sqrt(T)) )

of = ( K\*np.exp(-r\*T)\*NORMSDIST(-d2) - S\*NORMSDIST(-d1) ) - pm

val = of\*\*2

print("[σ]=",*s*,", Object Function Value:", val)

*return*(val)

s = fmin(ImpliedVolatilityPut, [0.3])

*import* numpy *as* np

*from* scipy.stats *import* norm

N = norm.cdf

def **bs\_call**(*S*, *K*, *T*, *r*, *vol*):

d1 = (np.log(*S*/*K*) + (*r* + 0.5\**vol*\*\*2)\**T*) / (*vol*\*np.sqrt(*T*))

d2 = d1 - *vol* \* np.sqrt(*T*)

*return* *S* \* norm.cdf(d1) - np.exp(-*r* \* *T*) \* *K* \* norm.cdf(d2)

def **bs\_vega**(*S*, *K*, *T*, *r*, *sigma*):

d1 = (np.log(*S* / *K*) + (*r* + 0.5 \* *sigma* \*\* 2) \* *T*) / (*sigma* \* np.sqrt(*T*))

*return* *S* \* norm.pdf(d1) \* np.sqrt(*T*)

def **find\_vol**(*target\_value*, *S*, *K*, *T*, *r*, \**args*):

MAX\_ITERATIONS = 200

PRECISION = 1.0e-5

sigma = 0.5

*for* i *in* range(0, MAX\_ITERATIONS):

price = bs\_call(*S*, *K*, *T*, *r*, sigma)

vega = bs\_vega(*S*, *K*, *T*, *r*, sigma)

diff = *target\_value* - price *# our root*

*if* (abs(diff) < PRECISION):

*return* sigma

sigma = sigma + diff/vega *# f(x) / f'(x)*

*return* sigma *# value wasn't found, return best guess so far*

S = 45

K = 40

T = 6

r = 0.05

vol = 0.25

V\_market = bs\_call(S, K, T, r, vol)

implied\_vol = find\_vol(V\_market, S, K, T, r)

print ('Implied vol: %.2f%%' % (implied\_vol \* 100))

print ('Market price = %.2f' % V\_market)

print ('Model price = %.2f' % bs\_call(S, K, T, r, implied\_vol))