Derivative Securities: Lecture 5
American Options and Black Scholes PDE

Sources:

J. Hull

Avellaneda and Laurence

The Black Scholes PDE

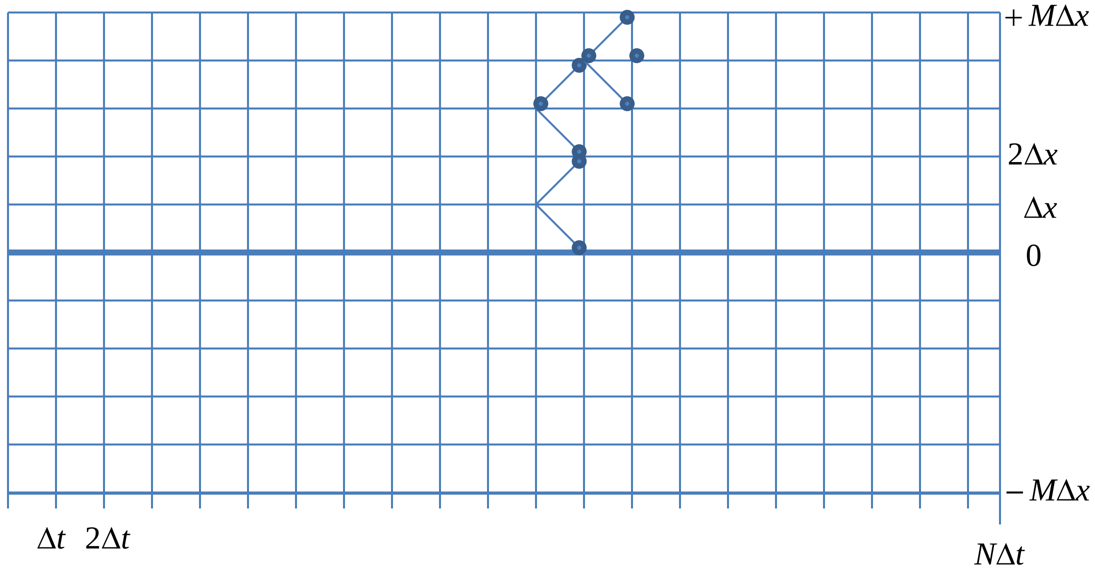
- The hedging argument for assets with normal returns presented at the end of Lecture 4 gave rise to the Black Scholes PDE

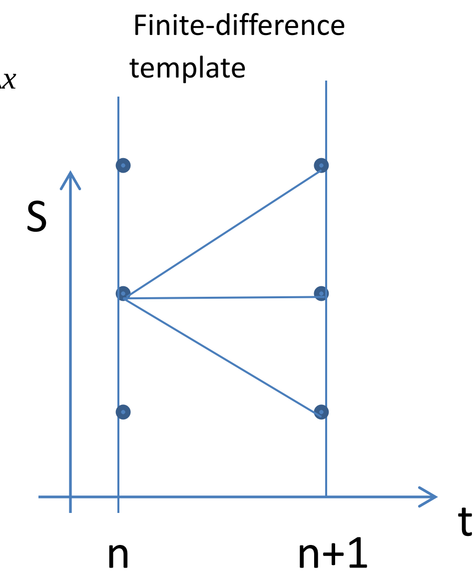
$$\frac{\partial C(S,t)}{\partial t}+\frac{\sigma^2 S^2}{2}\frac{\partial^2 C(S,t)}{\partial S^2}+(r-q)S\,\frac{\partial\ C(S,t)}{\partial S}-rC(S,t)=0$$

r=interest rate, q=dividend yield, $\sigma =$ volatility. The volatility is the annualized standard deviation of returns (it is not a market price or, rate, but rather a model input).

- We introduce a method for solving this PDE numerically on a grid.

Finite-difference scheme, or ``trinomial tree''





$$S_n^j = S_0 e^{j\Delta x}, \quad -M \le j \le +M$$

$$C_n^j \leftrightarrow C\left(S_n^j, n\Delta t\right), \; 0 \le n \le N$$

Change of variables

$$S = S_0 e^x$$

$$S\frac{\partial C}{\partial S} = S\frac{\partial C}{\partial x}\frac{\partial x}{\partial S} = S\frac{\partial C}{\partial x}\frac{1}{S} = \frac{\partial C}{\partial x}$$

$$S^2\frac{\partial^2 C}{\partial S^2} = S^2\frac{\partial}{\partial S}\left(\frac{1}{S}S\frac{\partial C}{\partial S}\right) = S^2\frac{\partial}{\partial S}\left(\frac{1}{S}\frac{\partial C}{\partial x}\right)$$

$$= S\frac{\partial}{\partial x}\left(\frac{1}{S}\frac{\partial C}{\partial x}\right)$$

$$= \frac{\partial^2 C}{\partial x^2} - \frac{\partial C}{\partial x}$$

BS equation in log-price

$$\boxed{\frac{\partial C}{\partial t} + \left(r - q - \frac{1}{2}\sigma^2\right)\frac{\partial C}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 C}{\partial x^2} - rC = 0}$$

Taylor expansion & symmetric finite-difference approximations for derivatives

$$f(x) = f(0) + f'(0)x + \frac{1}{2} f''(0)x^2 + \ldots$$

$$f(-x) = f(0) - f'(0)x + \frac{1}{2} f''(0)x^2 + \ldots$$

$$\therefore$$

$$f(x) - f(-x) = 2f'(0)x + o(x^2)$$
$$f(x) + f(-x) = 2f(0) + f''(0)x^2 + o(x^3)$$

$$\therefore$$

$$f'(0) = \frac{f(x) - f(-x)}{2x} + o(x)$$
$$f''(0) = \frac{f(x) + f(-x) - 2f(0)}{x^2} + o(x)$$

Symmetric finite difference approximations for first and second derivatives

Discretization of the PDE

$$\frac{\partial C(S,t)}{\partial t} \leftrightarrow \frac{C_{n+1}^{j} - C_{n}^{j}}{\Delta t}$$

Here we do not use symmetric differences

$$\frac{\partial C(S,t)}{\partial x} \leftrightarrow \frac{C_{n+1}^{j+1} - C_{n+1}^{j-1}}{2\Delta x}$$

Here use symmetric differences

$$\frac{\partial^2 C(S,t)}{\partial x^2} \leftrightarrow \frac{C_{n+1}^{j+1} + C_{n+1}^{j-1} - 2C_{n+1}^{j}}{(\Delta x)^2}$$

$$\frac{C_{n+1}^{j} - C_{n}^{j}}{\Delta t} + (r - q - \frac{\sigma^2}{2})\frac{C_{n+1}^{j+1} - C_{n+1}^{j-1}}{2\Delta x} + \frac{\sigma^2}{2}\frac{C_{n+1}^{j+1} + C_{n+1}^{j-1} - 2C_{n+1}^{j}}{(\Delta x)^2} - rC_{n}^{j} = 0$$

From PDE to recursive scheme

$$\frac{C_{n+1}^{j}-C_{n}^{j}}{\Delta t}+(r-q-\frac{\sigma^{2}}{2})\frac{C_{n+1}^{j+1}-C_{n+1}^{j-1}}{2\Delta x}+\frac{\sigma^{2}}{2}\frac{C_{n+1}^{j+1}+C_{n+1}^{j-1}-2C_{n+1}^{j}}{\left(\Delta x\right)^{2}}-rC_{n}^{j}=0$$

$$C_{n}^{j}=C_{n+1}^{j}+\left(\frac{\sigma^{2}\Delta t}{2\left(\Delta x\right)^{2}}+\frac{(r-q-\sigma^{2}/2)\Delta t}{2\Delta x}\right)C_{n+1}^{j+1}+\left(1-\frac{\sigma^{2}\Delta t}{\left(\Delta x\right)^{2}}\right)C_{n+1}^{j}+\left(\frac{\sigma^{2}\Delta t}{2\left(\Delta x\right)^{2}}-\frac{(r-q-\sigma^{2}/2)\Delta t}{2\Delta x}\right)C_{n+1}^{j-1}-r\Delta tC_{n}^{j}$$

$$\boxed{\begin{gathered}C_{n}^{j}=\frac{1}{1+r\Delta t}\left(p_{\mathrm{U}}C_{n+1}^{j+1}+p_{\mathrm{M}}C_{n+1}^{j}+p_{\mathrm{D}}C_{n+1}^{j-1}\right)\\ \left\{\begin{aligned}p_{\mathrm{U}}&=\frac{\sigma^{2}\Delta t}{2\left(\Delta x\right)^{2}}+\frac{(r-q-\sigma^{2}/2)\Delta t}{2\Delta x}\\ p_{\mathrm{M}}&=1-\frac{\sigma^{2}\Delta t}{\left(\Delta x\right)^{2}}\\ p_{\mathrm{D}}&=\frac{\sigma^{2}\Delta t}{2\left(\Delta x\right)^{2}}-\frac{(r-q-\sigma^{2}/2)\Delta t}{2\Delta x}\end{aligned}\right.\end{gathered}}$$

Interpreting the weights

- Notice that

$$p_{\mathrm{U}} + p_{\mathrm{M}} + p_{\mathrm{D}} = 1$$

- Set

$$\Delta x = \sigma_{\max}\sqrt{\Delta t}$$
$$\mu = r - q - \frac{\sigma^2}{2}$$
$$p = \frac{\sigma^2 \Delta t}{2(\Delta x)^2} = \frac{\sigma^2}{2\sigma_{\max}^2}$$

- The weights become

$$p_{\mathrm{U}} = p + \frac{\mu\sqrt{\Delta t}}{2\sigma_{\max}}$$
$$p_{\mathrm{M}} = 1 - 2p$$
$$p_{\mathrm{D}} = p - \frac{\mu\sqrt{\Delta t}}{2\sigma_{\max}}$$

Stability conditions & probabilities

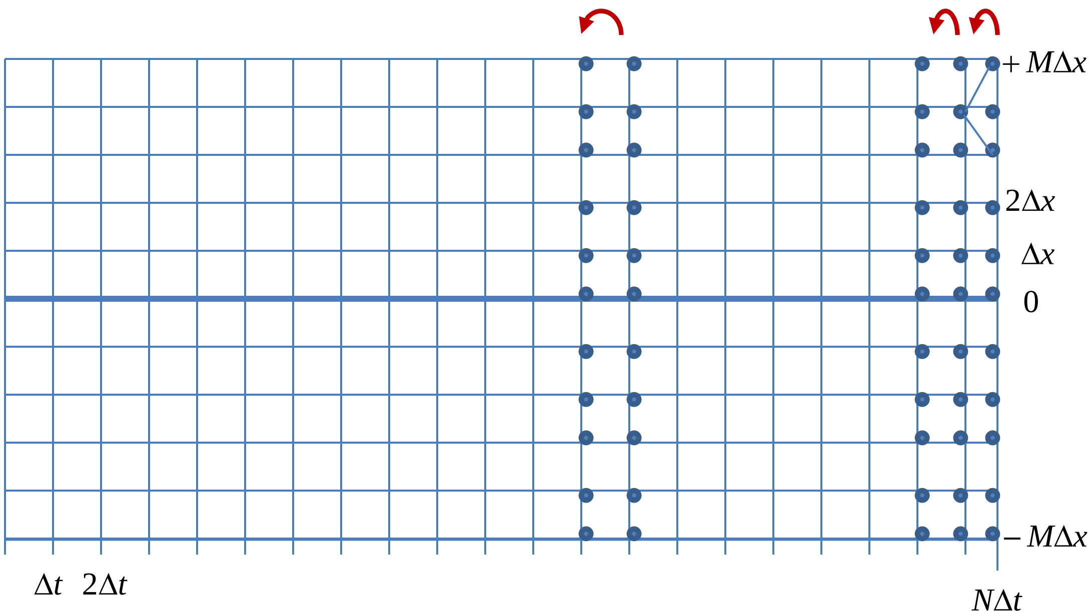
$$p < 1/2$$
and
$$\frac{|\mu|\sqrt{\Delta t}}{\sigma_{\max}} < 1 \qquad \Rightarrow \qquad p_U > 0,\, p_M > 0,\, p_D > 0$$

- In this case, the discretization of the PDE corresponds to discounting over probabilities

$$C_n^j = \frac{1}{1+r\Delta t}\left(p_{\mathrm{U}} C_{n+1}^{j+1} + p_{\mathrm{M}} C_{n+1}^{j} + p_{\mathrm{D}} C_{n+1}^{j-1}\right)$$

- This gives a simple and intuitive interpretation of the B-S PDE

European Options



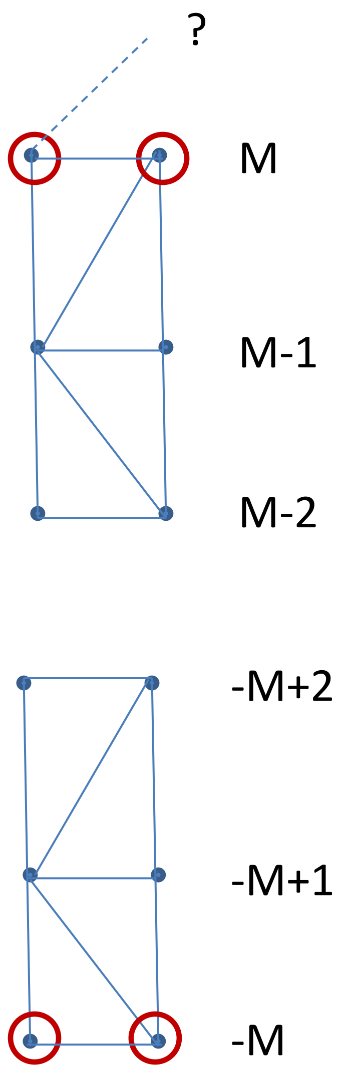
- Value at expiration date

$$C_N^j = \max\left(S_0 e^{j\sigma_{\max}\sqrt{\Delta t}} - K, 0\right), \quad -M \le j \le +M \qquad \text{(call)}$$
$$C_N^j = \max\left(K - S_0 e^{j\sigma_{\max}\sqrt{\Delta t}}, 0\right), \quad -M \le j \le +M \qquad \text{(put)}$$

- Solve recursively

$$C_n^j = \frac{1}{1+r\Delta t}\left[p_{\mathrm{U}} C_{n+1}^{j+1} + p_{\mathrm{M}} C_{n+1}^{j} + p_{\mathrm{D}} C_{n+1}^{j-1}\right], \quad -M < j < +M, \; n = N-1, N-2, \ldots, 0$$

Boundary Nodes



$$C_n^M = 2C_n^{M-1} - C_n^{M-2} \qquad \text{(upper boundary)}$$

$$C_n^{-M} = 2C_n^{-M+1} - C_n^{-M+2} \qquad \text{(lower boundary)}$$

These boundary conditions are called ``radiation boundary conditions'' or ``zero-gamma'' boundary conditions. They assume that there is <u>no convexity at the boundary</u>, so the values at the boundary will not affect the computation significantly.

(More on this later…)

VB pseudo code (1)

```
Function BSCall(ByVal S As Double, ByVal T As Double, ByVal K As Double, ByVal r As Double, _
ByVal q As Double, ByVal sigma As Double) As Double

'set mesh = 1day
Dim dt As Double
dt = 1# / 252
'set number of time steps
Dim N As Integer
N = CInt(T / dt)
'set carry
Dim mu As Double
mu = r – q-0.5 * sigma*sigma

'set sigma max for stability requirements
Dim smax As Double
smax = 2 * Abs(mu) * sqrt(dt)
If smax < sigma * sqrt(2) Then
smax = sigma * sqrt(2)
End If
If smax = 0 Then
BSCall = -9999
End If
```

This ensures that smax is large enough

VB Code (II)

```
'allocate arrays
Dim M As Integer
M = CInt(5 * sqrt(N))

Dim S() As Double
Dim C() As Double
Dim pC() As Double

ReDim C(1 To 2 * M + 1)
ReDim pC(1 To 2 * M + 1)
ReDim S(1 To 2 * M + 1)

'probabilities
Dim PU, PM, PD As Double
Dim p As Double
p = 0.5 * sigma * sigma / (smax * smax)

PU = p + 0.5 * mu * sqrt(dt) / smax
PM = 1 - 2 * p
PD = p - 0.5 * mu * sqrt(dt) / smax
```

This sets the vertical dimension

From the discretization of
The PDE

VB Code (III)

```
'initialize call payoff
Dim D, E As Double
D = 1# / (1 + r * dt)
E = Exp(smax * sqrt(dt))


S(1) = S * Exp(-M * smax * sqrt(dt))
For j = 2 To 2 * M + 1
S(j) = S(j - 1) * E
Next j

For j = 1 To 2 * M + 1
C(j) = Max(S(j) - K, 0)
Next j

' time loop
For K = 1 To N
   'interior nodes
   For j = 2 To 2 * M
   pC(j) = PU * C(j + 1) + PM * C(j) + PD * C(j - 1)
   pC(j) = pC(j) * D
   Next j

   'boundary nodes
   pC(1) = 2 * pC(2) - pC(3)
   pC(2 * M + 1) = 2 * pC(2 * M) - pC(2 * M - 1)

   'copy array
   For j = 1 To 2 * M + 1
   C(j) = pC(j)
   Next j

Next K

BSCall = C(M + 1)
End Function
```

Discount factor and vertical mesh size

Main loop

Answer= central vertical node

Discussion

- This is called an explicit scheme, which means that we ``roll back'', solving time n in terms of time n+1
- For this to work, we need smax large enough so that the ``probabilities'' are positive (stability)
- The requirement that M=5*sqrt(N) has to do with the fact that the grid must be large enough to avoid ``feeling the boundary''
- The result at the end is the full vertical array at n=0, so we get more information than just the central node, if we wish.

American Options

- We must enforce the requirement that, at each node, the value of the option is greater than the payoff (intrinsic value

$$C_n^j \geq \max\left(S_n^j - K, 0\right) \quad \text{(call)}$$
$$C_n^j \geq \max\left(K - S_n^j, 0\right) \quad \text{(put)}$$

Let $F(S)$ be the intrinsic value. Then,

$$C_n^j = \max\left[F\left(S_n^j\right), \frac{1}{1+r\Delta t}\left(p_{\mathrm{U}} C_{n+1}^{j+1} + p_{\mathrm{M}} C_{n+1}^{j} + p_{\mathrm{D}} C_{n+1}^{j-1}\right)\right]$$

Why is the numerical scheme correct?

- An American-style option is always greater than the IV
- Suppose that you know the value of the American option at time $t_{n+1} = (n+1)\Delta t$.
- A European option with payoff $F(S, t_{n+1}) = C_{n+1}^{j},\ \ S = S_0 e^{j\Delta x}$ expiring at time t_{n+1} has a value at time $t_n = n\Delta t$ equal to

$$V_n^{j} = \frac{1}{1+r\Delta t}\left[p_{\mathrm{U}} C_{n+1}^{j+1} + p_{\mathrm{M}} C_{n+1}^{j} + p_{\mathrm{D}} C_{n+1}^{j-1}\right]$$

- An American option gives the right to exercise at time t_n or to continue. If you continue, this is like holding the European-style derivative for one more time period. Therefore,

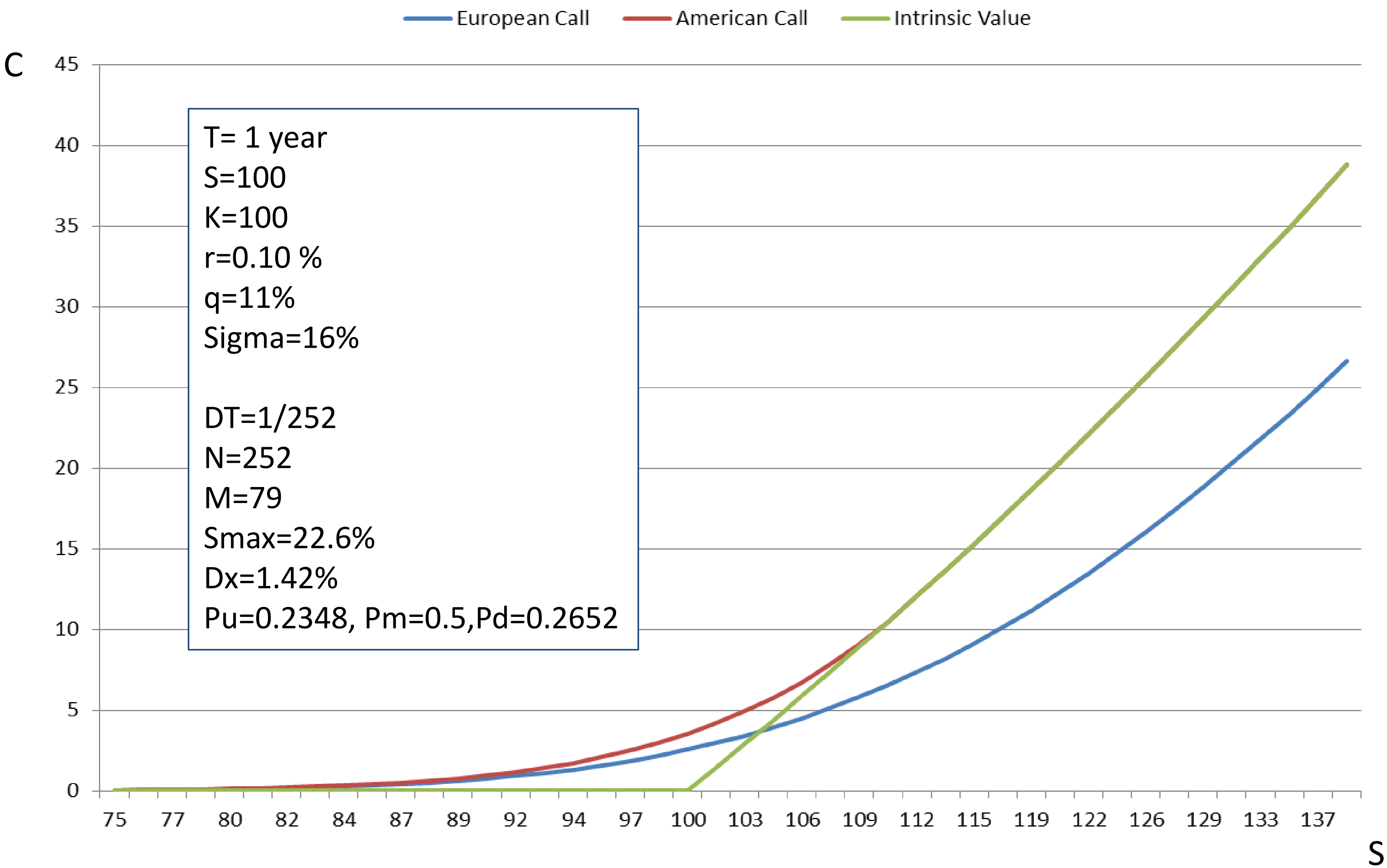
$$C_n^{j} = \max\left[IV\left(S_n^{j}\right), V_n^{j}\right] = \max\left[IV\left(S_n^{j}\right), \frac{1}{1+r\Delta t}\left[p_{\mathrm{U}} C_{n+1}^{j+1} + p_{\mathrm{M}} C_{n+1}^{j} + p_{\mathrm{D}} C_{n+1}^{j-1}\right]\right]$$

VB code for American Call

```
' time loop
For K = 1 To N
   'interior nodes
   For j = 2 To 2 * M
   pC(j) = PU * C(j + 1) + PM * C(j) + PD * C(j - 1)
   pC(j) = pC(j) * D
   Next j

   'boundary nodes
   pC(1) = 2 * pC(2) - pC(3)
   pC(2 * M + 1) = 2 * pC(2 * M) - pC(2 * M - 1)

   'copy array & compare with intrinsic value
   For j = 1 To 2 * M + 1
   C(j) = pC(j)

         If C(j)<Max(S(j)-K,0)  then
         C(j)=Max(S(j)-K,0)
         End if
   Next j

Next K

BSCall = C(M + 1)
End Function
```

This guarantees that C is at least equal to the intrinsic value. Everything else is the same.

Pricing a 1-year call numerically

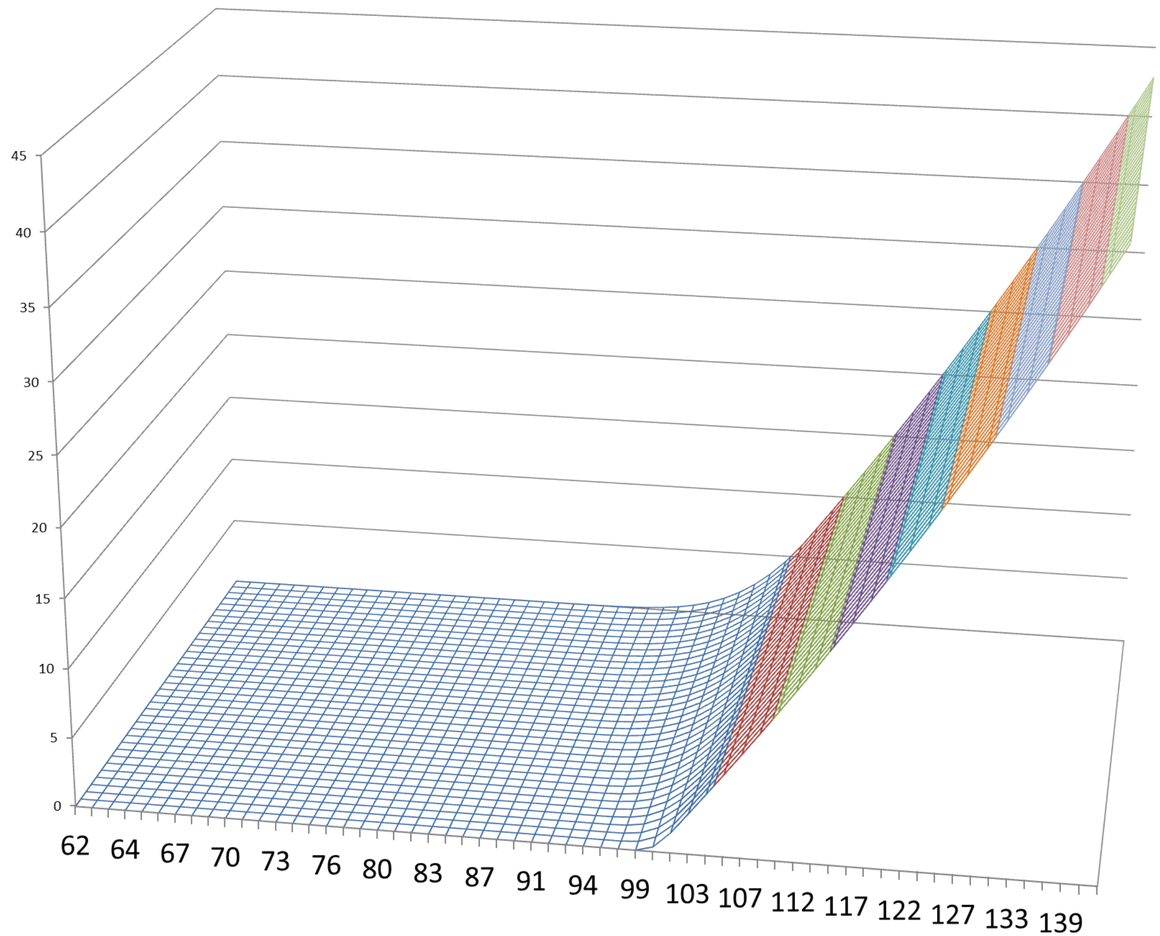


Numerical solution vs. Black-Scholes for European options

S	Numerical	Black-Scholes	Diff
74.13	0.02184	0.022141	3E-04
75.2	0.02903	0.029406	4E-04
76.28	0.03833	0.038791	5E-04
77.37	0.05027	0.050829	6E-04
78.48	0.06549	0.06616	7E-04
79.61	0.08474	0.085546	8E-04
80.75	0.10892	0.109889	1E-03
81.91	0.13909	0.14024	0.001
83.09	0.17647	0.177822	0.001
84.28	0.22244	0.224035	0.002
85.49	0.27862	0.280472	0.002
86.72	0.34677	0.348926	0.002
87.96	0.42891	0.431395	0.002
89.22	0.52724	0.530085	0.003
90.5	0.64416	0.647404	0.003
91.8	0.78228	0.785954	0.004
93.12	0.94437	0.948515	0.004
94.46	1.13339	1.138026	0.005
95.81	1.35239	1.357555	0.005
97.19	1.60454	1.610268	0.006
98.58	1.89309	1.899388	0.006
100	2.22126	2.228156	0.007
101.4	2.59228	2.59978	0.008
102.9	3.00928	3.017386	0.008
104.4	3.47525	3.483969	0.009
105.9	3.99303	4.002341	0.009
107.4	4.56521	4.575086	0.01
108.9	5.1941	5.204516	0.01
110.5	5.88171	5.892626	0.011
112.1	6.62971	6.641072	0.011
113.7	7.43938	7.45114	0.012
115.3	8.31164	8.323734	0.012
117	9.24702	9.259366	0.012
118.7	10.2456	10.25817	0.013
120.4	11.3073	11.31989	0.013
122.1	12.4313	12.44395	0.013
123.8	13.6168	13.62941	0.013
125.6	14.8626	14.87508	0.012

- Same parameters as previous example
- Compared BS with numerical scheme
- Adjust the time-step to produce acceptable error
- Use numerical code to price American options

Numerical solution as a surface



Early-exercise boundary

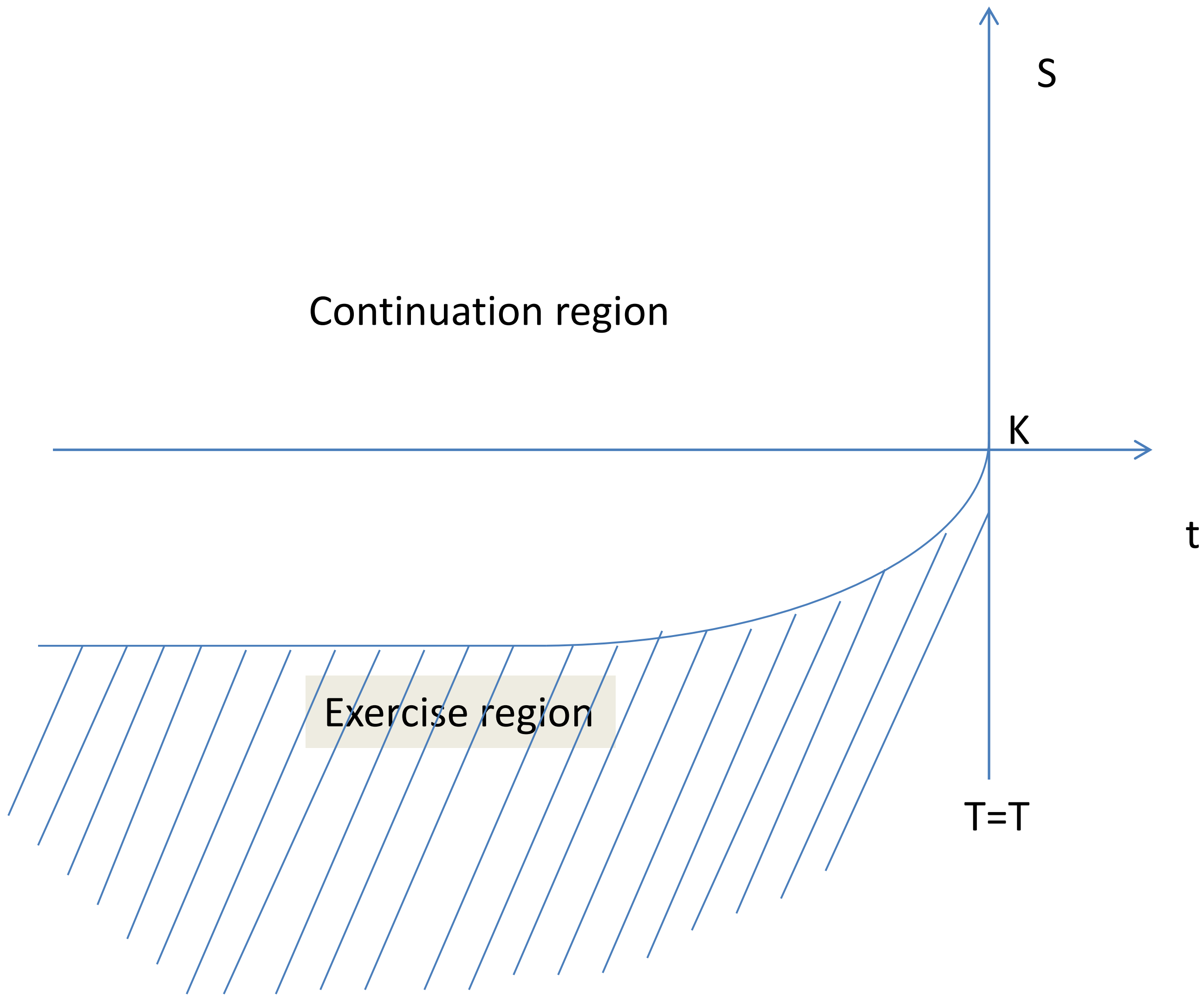
- Exercising a call option makes sense only if the stock is sufficiently high (and the dividend income is greater than the potential increase in the option price).

- Exercising a put option makes sense only if the stock is sufficiently low, so that the interest income from the cash received exceeds the expected gains from increase in option price.

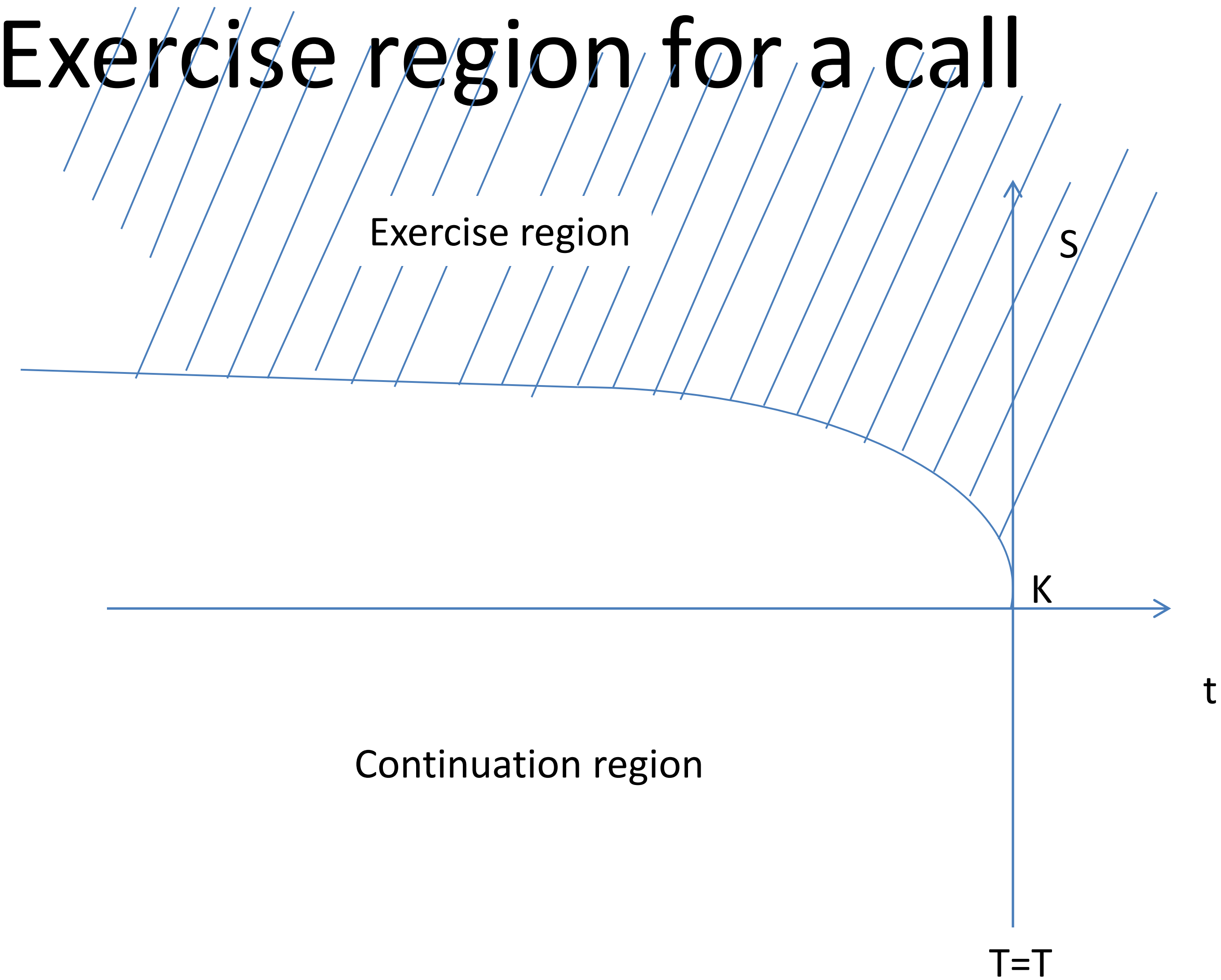
- Therefore, the **exercise regions** for calls/puts will be of the form

$$\boxed{ER_{call}=\{ (S,t):\ S>S^*(t)\} \qquad ER_{put}=\{ (S,t):\ S<S^{**}(t)\}}$$

Exercise region for a put



Exercise region for a call



Applications of the Black-Scholes pricing model

- In the case of European-style options, we obtain a compact formula for the value of options:

$$C_{eur} = BSCall(S, K, T, r, d, \sigma) \quad P_{eur} = BSPut(S, K, T, r, d, \sigma)$$

- In the case of American-style options, we have numerical scheme which depends on the same 6 parameters and gives the value with arbitrary precision.

- Of the 6 parameters, 5 of them are observable or derivable from the market (e.g. implied dividend)

- The volatility parameter is NOT observable or derivable from the market in an unequivocal way. It is an essential component of the model.

Implied Volatility

- The implied volatility of an option is the volatility that makes the Black-Scholes pricing formula true

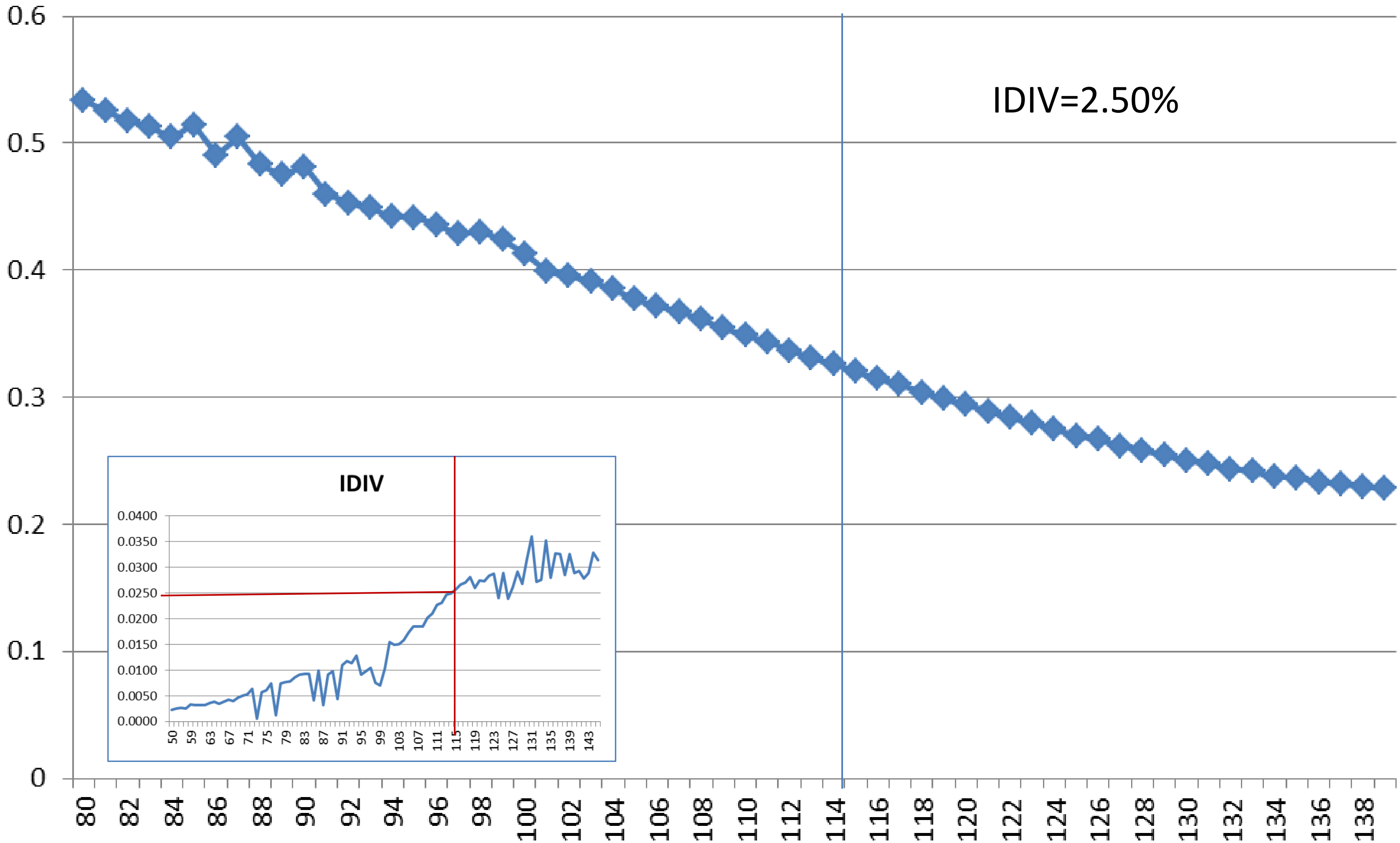
$$C = BSCall(S, T, K, r, d, \sigma_{imp}), \quad P = BSPut(S, T, K, r, d, \sigma_{imp})$$

- Given (S,K,T,r,q) and the price of an option, there is a unique implied vol associated with a given price. The reason is that

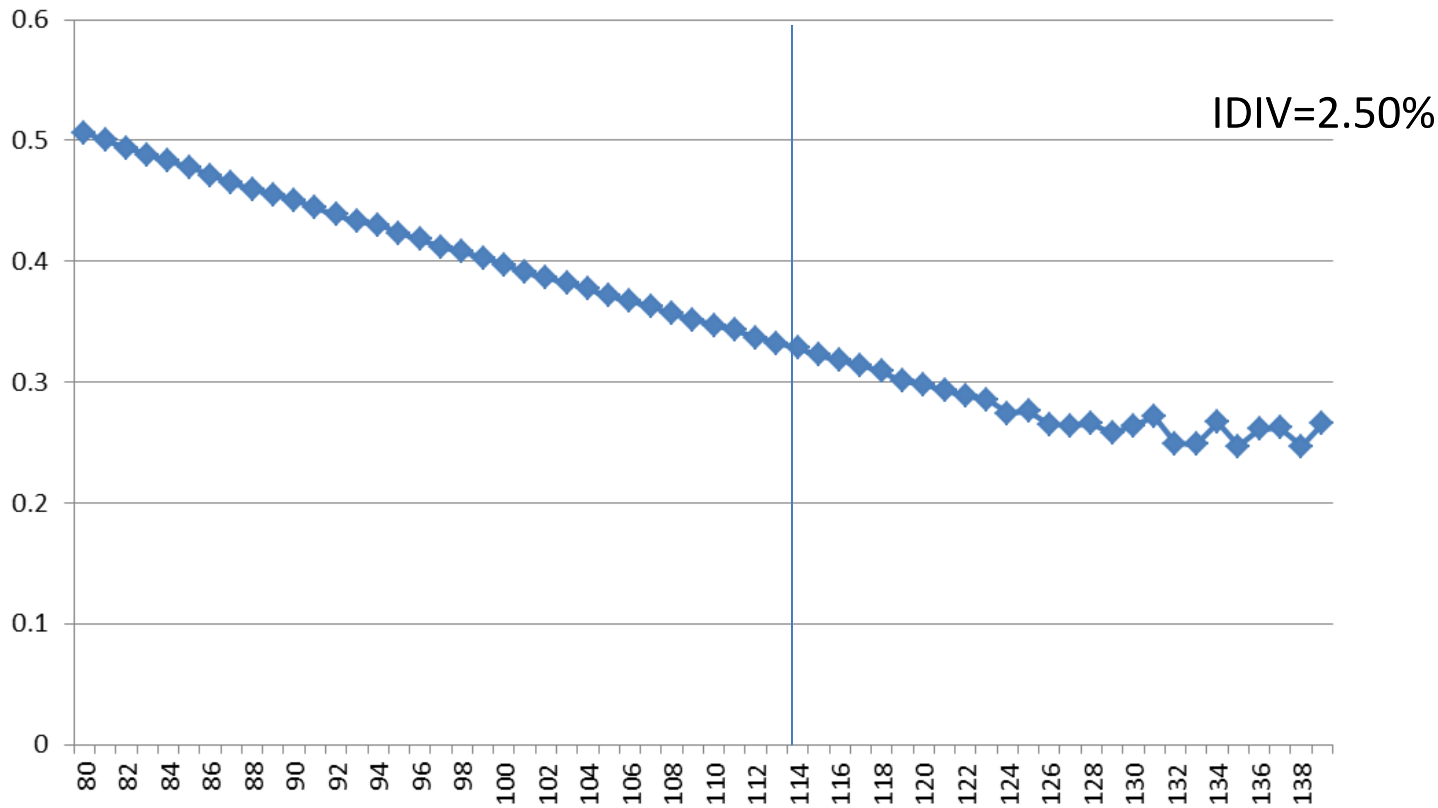
$$\frac{\partial BSCall}{\partial \sigma} > 0, \quad \frac{\partial BSPut}{\partial \sigma} > 0$$

- Usually computed from mid-prices (bid+offer)/2. We can also talk about a **bid implied vol** and an **offer implied vol**, associated with bid prices and offer prices.

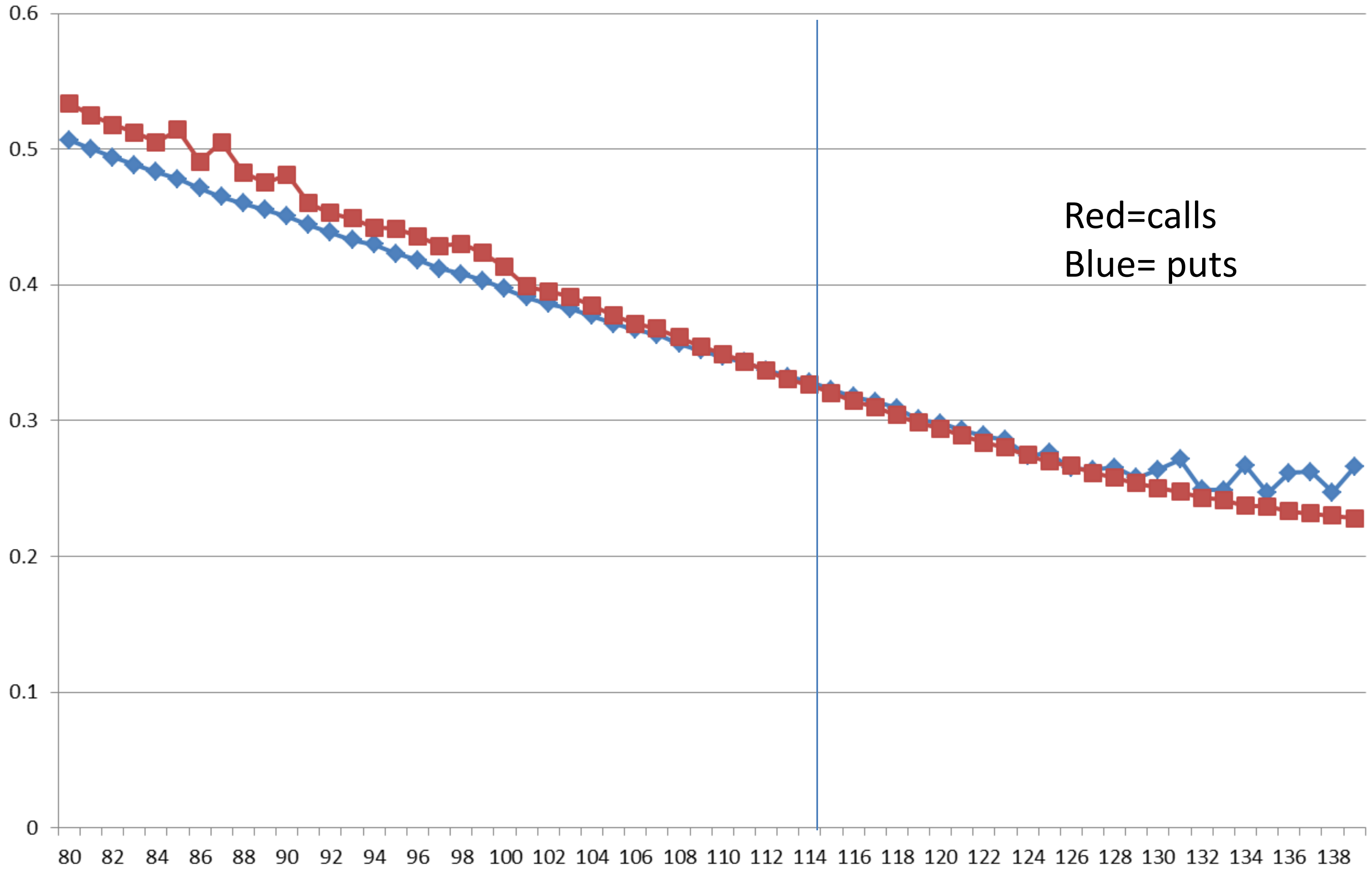
Call Implied Volatility (SPY Dec 17 calls)



Put Implied Vols (SPY Dec 17 Calls)



Calls and Puts together



Implied Volatility

- Implied volatility of OTM options are more stable than ITM
- IVOLS of calls and puts should be approximately equal due to the fact that we determined the dividend yields implicitly
- For SPY, the implied volatility is a decreasing function of the strike price. This is known as the **volatility skew** in the business.
- Volatilities are not constant across strikes, but they vary relatively smoothly.
- Option markets can be viewed as volatility markets, as we will soon see.