Demonstrating ImplicitFiniteDifference class

In this notebook, I will demonstrate the functionality of the `ImplicitFiniteDifference` and `Option` classes defined in `methods.implicitfinitedifference` and `methods.option` respectively. I will do this by working through exercises proposed in:

Hull, J. C. (2003). *Options, Futures, and Other Derivatives* (2nd ed.). Prentice-Hall. Chapter 14: Numerical Methods.

Contents

In [ ]:
```
import sys
import os
sys.path.append(os.path.abspath(os.path.join('..')))

from methods.implicitfinitedifference import ImplicitFiniteDifference
from methods.option import Option, Put, Call
from methods.node import Node

import matplotlib.pyplot as plt
```

Implicit Finite Differences

Ex 14.1 American Put on no-dividend asset

In [ ]:
```
# Time to maturity
T = 0.4167
# Number of timesteps
tsteps = 10
# Boundary value of asset price
Smax = 100
# Value steps
vsteps = 20
# Boundary option prices
fmax = 0
fmin = 50
# Current asset price
S = 50
# Option to price
X = 50
option = Put(X, american = True)
# Risk free interest rate and asset volatility (we assume these are constant up
# to maturity). ImplicitFiniteDifference class can be extended so that these
# can be a function of time.
r = 0.10
sigma = 0.40
```

In [ ]:
```
# Initialize and fit ImplicitFiniteDifference model
american_put_model = ImplicitFiniteDifference(T, tsteps, Smax, vsteps, fmax, fmin)
american_put_model.fit(r=r, sigma=sigma, S=S, option=option)
```

In [ ]:
```
print(f'The computed option price for the given american put is {american_put_model.f:.5f} dollars.')
```

```
The computed option price for the given american put is 4.06732 dollars.
```

In [ ]:
```
american_put_model.show()
```

Out[ ]:

Time (years)	0.00 [0]	0.04 [1]	0.08 [2]	0.13 [3]	0.17 [4]	0.21 [5]	0.25 [6]	0.29 [7]	0.33 [8]	0.38 [9]	0.42 [10]
Asset Price (dollars)											
100.00 [20]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95.00 [19]	0.02	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90.00 [18]	0.05	0.04	0.03	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00
85.00 [17]	0.09	0.07	0.05	0.03	0.02	0.01	0.01	0.00	0.00	0.00	0.00
80.00 [16]	0.16	0.12	0.09	0.07	0.04	0.03	0.02	0.01	0.00	0.00	0.00
75.00 [15]	0.27	0.22	0.17	0.13	0.09	0.06	0.03	0.02	0.01	0.00	0.00
70.00 [14]	0.47	0.39	0.32	0.25	0.18	0.13	0.08	0.04	0.02	0.00	0.00
65.00 [13]	0.82	0.71	0.60	0.49	0.38	0.28	0.19	0.11	0.05	0.02	0.00
60.00 [12]	1.42	1.27	1.11	0.95	0.78	0.62	0.45	0.30	0.16	0.05	0.00
55.00 [11]	2.43	2.24	2.05	1.83	1.61	1.36	1.09	0.81	0.51	0.22	0.00
50.00 [10]	4.07	3.88	3.67	3.45	3.19	2.91	2.57	2.17	1.66	0.99	0.00
45.00 [9]	6.58	6.44	6.29	6.13	5.96	5.77	5.57	5.36	5.17	5.02	5.00
40.00 [8]	10.15	10.10	10.05	10.01	10.00	10.00	10.00	10.00	10.00	10.00	10.00
35.00 [7]	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00
30.00 [6]	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
25.00 [5]	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00
20.00 [4]	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00
15.00 [3]	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
10.00 [2]	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
5.00 [1]	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00
0.00 [0]	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

In [ ]:
```
# A logarithmic scale is applied to give a visual representation of varying option prices as a function of time
# and underlying asset price
american_put_model.show(heat=True, scale='log')
```

Out[ ]:

Time (years)	0.00 [0]	0.04 [1]	0.08 [2]	0.13 [3]	0.17 [4]	0.21 [5]	0.25 [6]	0.29 [7]	0.33 [8]	0.38 [9]	0.42 [10]
Asset Price (dollars)											
100.00 [20]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95.00 [19]	0.02	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90.00 [18]	0.05	0.04	0.03	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00
85.00 [17]	0.09	0.07	0.05	0.03	0.02	0.01	0.01	0.00	0.00	0.00	0.00
80.00 [16]	0.16	0.12	0.09	0.07	0.04	0.03	0.02	0.01	0.00	0.00	0.00
75.00 [15]	0.27	0.22	0.17	0.13	0.09	0.06	0.03	0.02	0.01	0.00	0.00
70.00 [14]	0.47	0.39	0.32	0.25	0.18	0.13	0.08	0.04	0.02	0.00	0.00
65.00 [13]	0.82	0.71	0.60	0.49	0.38	0.28	0.19	0.11	0.05	0.02	0.00
60.00 [12]	1.42	1.27	1.11	0.95	0.78	0.62	0.45	0.30	0.16	0.05	0.00
55.00 [11]	2.43	2.24	2.05	1.83	1.61	1.36	1.09	0.81	0.51	0.22	0.00
50.00 [10]	4.07	3.88	3.67	3.45	3.19	2.91	2.57	2.17	1.66	0.99	0.00
45.00 [9]	6.58	6.44	6.29	6.13	5.96	5.77	5.57	5.36	5.17	5.02	5.00
40.00 [8]	10.15	10.10	10.05	10.01	10.00	10.00	10.00	10.00	10.00	10.00	10.00
35.00 [7]	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00
30.00 [6]	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
25.00 [5]	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00
20.00 [4]	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00
15.00 [3]	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
10.00 [2]	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
5.00 [1]	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00
0.00 [0]	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

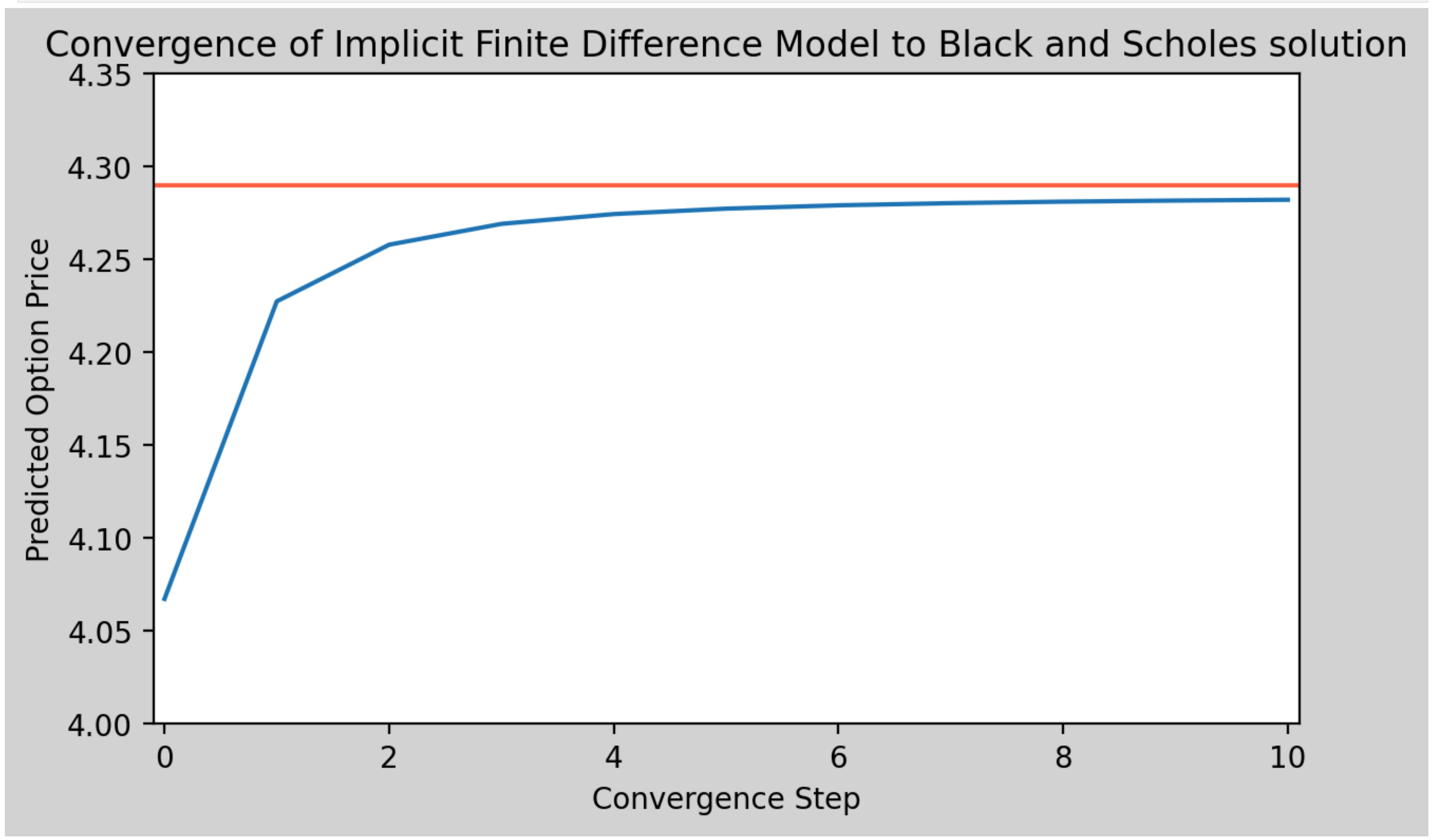
Convergence to solution of Black-Scholes differential equation

We can visualize how the model converges to the solution of the Black Scholes equation with the given boundary conditions by increasing the value of tsteps (i.e. bringing the module closer to continuous time computation) and vsteps (underlying asset price closer to continuous spectrum). The text followed suggest keeping the price steps proportional to the square root of the time steps as they approach zero: this give vsteps proportional to square root of tsteps.

In [ ]:
```
initial_vvalue = 20
initial_tvalue = 10
alpha = initial_vvalue/(initial_tvalue)**(1/2)

vvalues = [20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220]
tvalues = [int((v/alpha)**2) for v in vvalues]
fvalues = []
for i in range(len(vvalues)):
    model = ImplicitFiniteDifference(T, tvalues[i], Smax, vvalues[i], fmax, fmin)
    model.fit(r=r, sigma=sigma, S=S, option=option)
    fvalues.append(model.f)
```

In [ ]:
```
plt.figure(figsize=(7,4), dpi=200, facecolor='lightgray')
plt.plot(range(len(vvalues)), fvalues)
plt.title('Convergence of Implicit Finite Difference Model to Black and Scholes solution')
plt.xlabel('Convergence Step')
plt.ylabel('Predicted Option Price')
plt.ylim((4,4.35))
plt.xlim((-0.1,len(vvalues)-0.9))
plt.axhline(4.29, color='tomato')
plt.show()
```



Explicit Finite Differences

The `ImplicitFiniteDifference` class also implements explicit finite differences computation: note that this method is based on the assumption that the delta and gamma of the derivative considered are extremely similar between adjacent time points and are therefore interchangeable in the finite differences approximation of the Black-Scholes equation. This is close to true when timesteps are very small but in some cases can lead to null results as is shown below.

This is done to simplify the equations behind implicit finite differences and to avoid inverting a matrix: however, inverting a matrix with the numpy library is very efficient and the explicit method's necessity of smaller timesteps slows computation down considerably when decreasing timestep and underlying value step size to converge to the Black-Scholes equation's solution.

We will work with the same example as above.

In [ ]:
```
# Time to maturity
T = 0.4167
# Number of timesteps
tsteps = 10
# Boundary value of asset price
Smax = 100
# Value steps
vsteps = 20
# Boundary option prices
fmax = 0
fmin = 50
# Current asset price
S = 50
# Option to price
X = 50
option = Put(X, american = True)
# Risk free interest rate and asset volatility (we assume these are constant up
# to maturity). ImplicitFiniteDifference class can be extended so that these
# can be a function of time.
r = 0.10
sigma = 0.40
```

In [ ]:
```
# Initialize both implicit and explicit models, note they are identical but
# we will use a different fit function on each.
implicit_model = ImplicitFiniteDifference(T, tsteps, Smax, vsteps, fmax, fmin)
explicit_model = ImplicitFiniteDifference(T, tsteps, Smax, vsteps, fmax, fmin)

implicit_model.fit(r=r, sigma=sigma, S=S, option=option)
explicit_model.explicit_fit(r=r, sigma=sigma, S=S, option=option)
```

In [ ]:
```
print(f'The computed option price for the implicit model is {implicit_model.f:.5f} dollars.')
print(f'The computed option price for the explicit model is {explicit_model.f:.5f} dollars.')
```

```
The computed option price for the implicit model is 4.06732 dollars.
The computed option price for the explicit model is 4.25695 dollars.
```

We can see that the two models give considerably different results. The explicit model's map also shows something unusual happening:

In [ ]:
```
explicit_model.show(heat=True, scale='log')
```

Out[ ]:

Time (years)	0.00 [0]	0.04 [1]	0.08 [2]	0.13 [3]	0.17 [4]	0.21 [5]	0.25 [6]	0.29 [7]	0.33 [8]	0.38 [9]	0.42 [10]
Asset Price (dollars)											
100.00 [20]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95.00 [19]	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90.00 [18]	-0.11	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
85.00 [17]	0.28	-0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
80.00 [16]	-0.13	0.20	-0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
75.00 [15]	0.46	0.06	0.20	0.04	0.06	0.00	0.00	0.00	0.00	0.00	0.00
70.00 [14]	0.32	0.46	0.23	0.25	0.10	0.09	0.00	0.00	0.00	0.00	0.00
65.00 [13]	0.91	0.68	0.63	0.44	0.37	0.21	0.14	0.00	0.00	0.00	0.00
60.00 [12]	1.48	1.37	1.17	1.02	0.81	0.65	0.42	0.27	0.00	0.00	0.00
55.00 [11]	2.59	2.39	2.21	1.99	1.77	1.50	1.24	0.90	0.59	0.00	0.00
50.00 [10]	4.26	4.08	3.89	3.68	3.44	3.18	2.87	2.53	2.07	1.56	0.00
45.00 [9]	6.76	6.61	6.47	6.31	6.15	5.96	5.75	5.50	5.24	5.00	5.00
40.00 [8]	10.28	10.20	10.13	10.06	10.01	10.00	10.00	10.00	10.00	10.00	10.00
35.00 [7]	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00
30.00 [6]	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
25.00 [5]	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00
20.00 [4]	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00
15.00 [3]	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
10.00 [2]	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
5.00 [1]	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00
0.00 [0]	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

We can see that the model allows negative values for the derivative's price (the ability to exercise the put early is overwritten to show this problem). This shows that the assumption which allows explicit finite differences does not hold well in this case and therefore leads to negative values for an option with positive payoff, which is not reasonable.

By increasing the tsteps attribute, meaning timesteps in the models are smaller, the assumption holds better and the two models predict much closer values.

In [ ]:
```
tsteps = 500

# Initialize both implicit and explicit models, note they are identical but
# we will use a different fit function on each.
implicit_model = ImplicitFiniteDifference(T, tsteps, Smax, vsteps, fmax, fmin)
explicit_model = ImplicitFiniteDifference(T, tsteps, Smax, vsteps, fmax, fmin)

implicit_model.fit(r=r, sigma=sigma, S=S, option=option)
explicit_model.explicit_fit(r=r, sigma=sigma, S=S, option=option)

print(f'The computed option price for the implicit model is {implicit_model.f:.5f} dollars.')
print(f'The computed option price for the explicit model is {explicit_model.f:.5f} dollars.')
```

```
The computed option price for the implicit model is 4.17212 dollars.
The computed option price for the explicit model is 4.17594 dollars.
```

Additionally, no negative values appear in the method's map:

In [ ]:
```
explicit_model.show(heat=True, scale='log')
```

Out[ ]:

Time (years)	0.00 [0]	0.00 [1]	0.00 [2]	0.00 [3]	0.00 [4]	0.00 [5]	0.01 [6]	0.01 [7]	0.01 [8]	0.01 [9]	0.01 [10]	0.01 [11]	0.01 [12]
Asset Price (dollars)													
100.00 [20]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95.00 [19]	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
90.00 [18]	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
85.00 [17]	0.08	0.08	0.08	0.08	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
80.00 [16]	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.13	0.13	0.13
75.00 [15]	0.26	0.26	0.26	0.26	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
70.00 [14]	0.47	0.47	0.46	0.46	0.46	0.46	0.46	0.46	0.45	0.45	0.45	0.45	0.45
65.00 [13]	0.83	0.83	0.83	0.83	0.83	0.82	0.82	0.82	0.82	0.81	0.81	0.81	0.81
60.00 [12]	1.46	1.46	1.46	1.45	1.45	1.45	1.45	1.44	1.44	1.44	1.43	1.43	1.43
55.00 [11]	2.51	2.51	2.50	2.50	2.50	2.49	2.49	2.48	2.48	2.48	2.47	2.47	2.47
50.00 [10]	4.18	4.17	4.17	4.17	4.16	4.16	4.15	4.15	4.15	4.14	4.14	4.14	4.13
45.00 [9]	6.69	6.69	6.68	6.68	6.68	6.68	6.67	6.67	6.67	6.66	6.66	6.66	6.66
40.00 [8]	10.25	10.25	10.25	10.25	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.23	10.23
35.00 [7]	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00
30.00 [6]	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
25.00 [5]	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00
20.00 [4]	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00
15.00 [3]	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
10.00 [2]	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
5.00 [1]	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00
0.00 [0]	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

Time (years)	0.01 [13]	0.01 [14]	0.01 [15]	0.01 [16]	0.01 [17]	0.02 [18]	0.02 [19]	0.02 [20]	0.02 [21]	0.02 [22]	0.02 [23]	0.02 [24]	0.02 [25]
Asset Price (dollars)													
100.00 [20]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95.00 [19]	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
90.00 [18]	0.04	0.04	0.04	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
85.00 [17]	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
80.00 [16]	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.12	0.12
75.00 [15]	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.23	0.23	0.23	0.23
70.00 [14]	0.45	0.44	0.44	0.44	0.44	0.44	0.44	0.43	0.43	0.43	0.43	0.43	0.43
65.00 [13]	0.80	0.80	0.80	0.80	0.79	0.79	0.79	0.79	0.79	0.78	0.78	0.78	0.78
60.00 [12]	1.42	1.42	1.42	1.42	1.41	1.41	1.41	1.40	1.40	1.40	1.39	1.39	1.39
55.00 [11]	2.46	2.46	2.46	2.45	2.45	2.45	2.44	2.44	2.43	2.43	2.43	2.42	2.42
50.00 [10]	4.13	4.13	4.12	4.12	4.11	4.11	4.11	4.10	4.10	4.10	4.09	4.09	4.09
45.00 [9]	6.65	6.65	6.65	6.64	6.64	6.64	6.64	6.63	6.63	6.63	6.62	6.62	6.62
40.00 [8]	10.23	10.23	10.23	10.23	10.23	10.22	10.22	10.22	10.22	10.22	10.22	10.22	10.21
35.00 [7]	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00
30.00 [6]	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
25.00 [5]	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00	25.00
20.00 [4]	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00
15.00 [3]	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
10.00 [2]	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
5.00 [1]	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00	45.00
0.00 [0]	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

In [ ]: