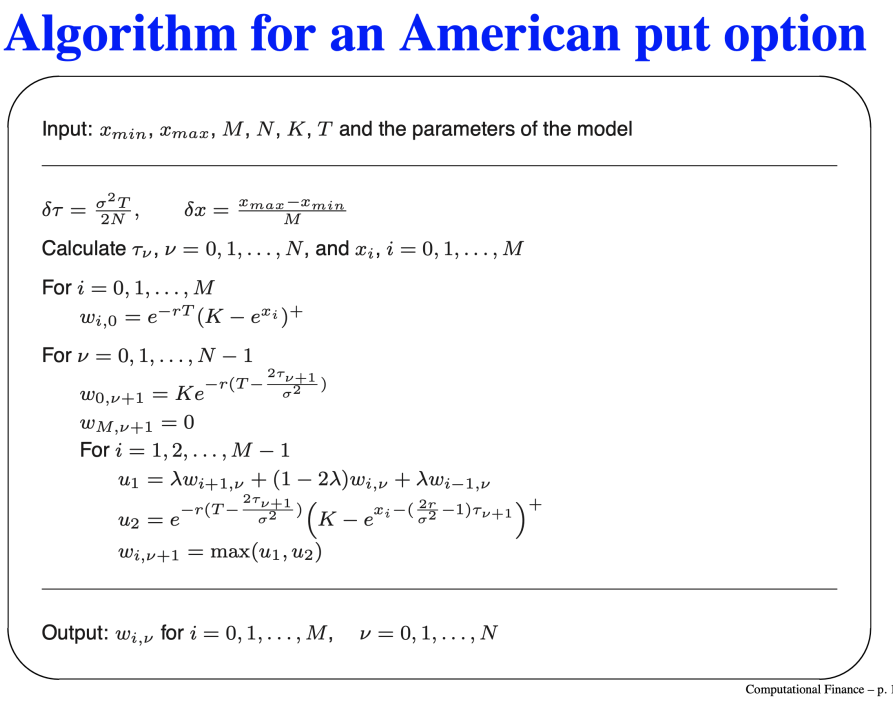


# Algorithm for an American put option

Input: $x_{min}$, $x_{max}$, $M$, $N$, $K$, $T$ and the parameters of the model

---

$\delta\tau = \frac{\sigma^2 T}{2N}$, $\quad \delta x = \frac{x_{max} - x_{min}}{M}$

Calculate $\tau_\nu$, $\nu = 0, 1, \ldots, N$, and $x_i$, $i = 0, 1, \ldots, M$

For $i = 0, 1, \ldots, M$

$\quad w_{i,0} = e^{-rT}(K - e^{x_i})^+$

For $\nu = 0, 1, \ldots, N - 1$

$\quad w_{0,\nu+1} = Ke^{-r(T - \frac{2\tau_{\nu+1}}{\sigma^2})}$

$\quad w_{M,\nu+1} = 0$

$\quad$ For $i = 1, 2, \ldots, M - 1$

$\qquad u_1 = \lambda w_{i+1,\nu} + (1 - 2\lambda)w_{i,\nu} + \lambda w_{i-1,\nu}$

$\qquad u_2 = e^{-r(T - \frac{2\tau_{\nu+1}}{\sigma^2})}\left(K - e^{x_i - (\frac{2r}{\sigma^2} - 1)\tau_{\nu+1}}\right)^+$

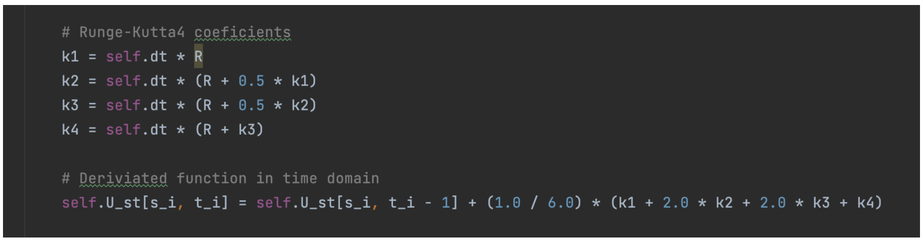
$\qquad w_{i,\nu+1} = \max(u_1, u_2)$

---

Output: $w_{i,\nu}$ for $i = 0, 1, \ldots, M$, $\quad \nu = 0, 1, \ldots, N$

Computational Finance – p. 1

Figure 1: Algorithm - American BS



```
# Runge-Kutta4 coeficients
k1 = self.dt * R
k2 = self.dt * (R + 0.5 * k1)
k3 = self.dt * (R + 0.5 * k2)
k4 = self.dt * (R + k3)

# Deriviated function in time domain
self.U_st[s_i, t_i] = self.U_st[s_i, t_i - 1] + (1.0 / 6.0) * (k1 + 2.0 * k2 + 2.0 * k3 + k4)
```

Figure 2: European Option computation

```python
# Runge-Kutta4 coeficients
k1 = self.dt * R
k2 = self.dt * (R + 0.5 * k1)
k3 = self.dt * (R + 0.5 * k2)
k4 = self.dt * (R + k3)

# Deriviated function in time domain
o = Option(s, self.k, 1, self.r, self.sigma)
actual_price = o.euro_vanilla_call()
calculated_price = self.U_st[s_i, t_i - 1] + (1.0 / 6.0) * (k1 + 2.0 * k2 + 2.0 * k3 + k4)

if actual_price >= calculated_price:
    self.U_st[s_i, t_i] = actual_price
else:
    self.U_st[s_i, t_i] = calculated_price
```

Figure 3: American Option computation

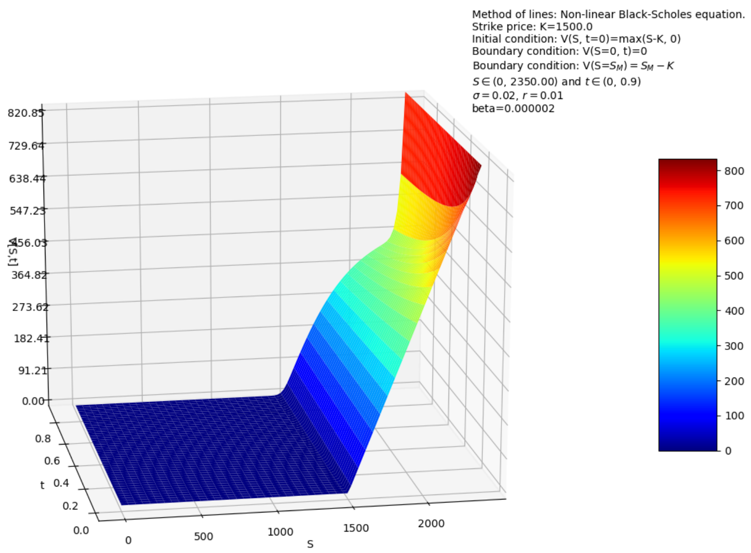



Figure 4: European Option - non-linear BS

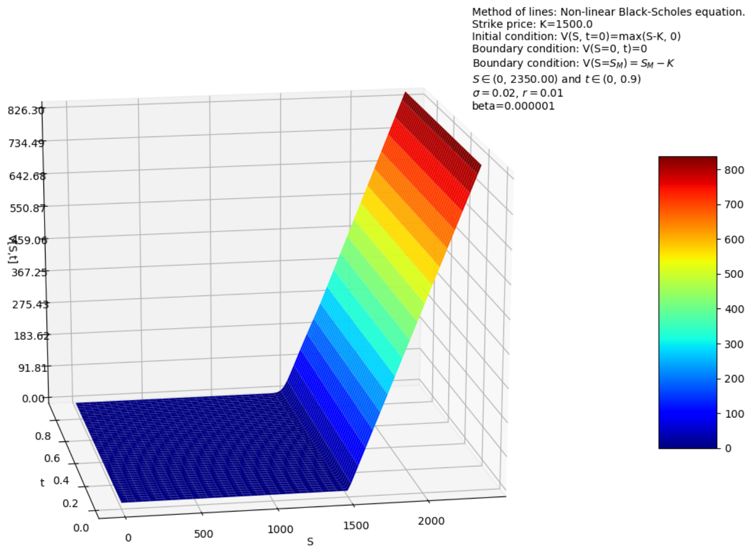



Figure 5: American Option - non-linear BS

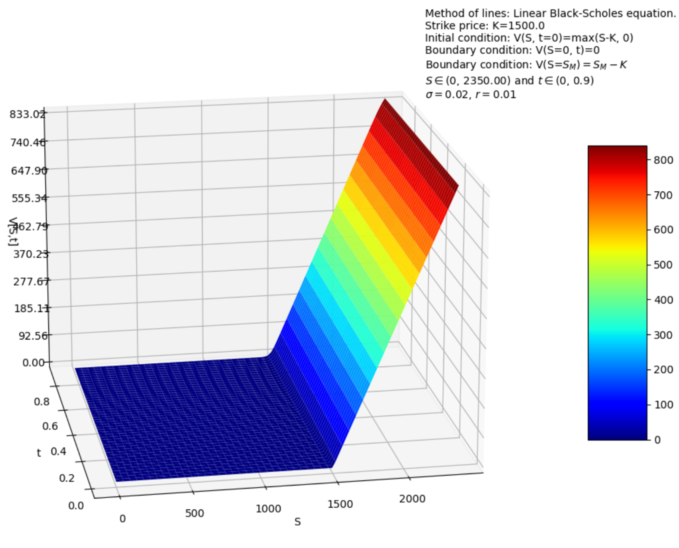



Figure 6: European Option - linear BS

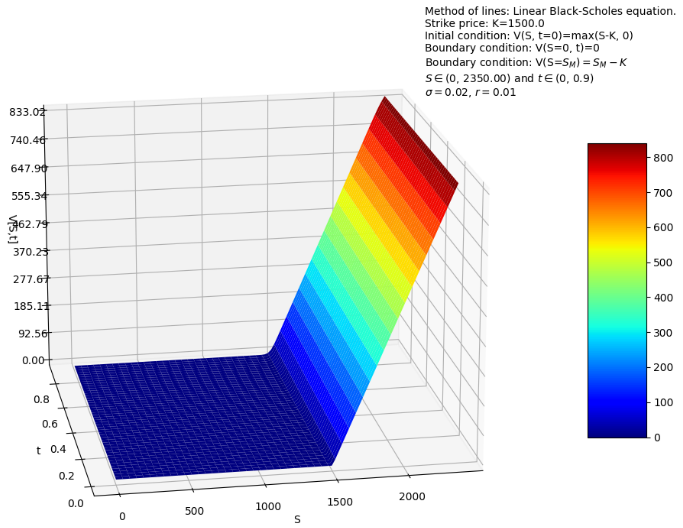



Figure 7: American Option - linear BS