Penalty Methods in Financial Option Pricing

Ray Wu and Christina C. Christara

University of Toronto
Department of Computer Science

July 14, 2020

Outline

- Overview
- Borrow-Lend
  - Policy Iteration
  - Penalty Iteration
  - Numerical Results
- Stock Borrowing Fees
  - Policy Iteration
  - Penalty Iteration
  - Numerical Results
- Stock Borrowing Fees with American Early Exercise
  - Policy Iteration
  - Double Penalty Iteration
  - Numerical Results
- Summary & Conclusions

Overview

The Black-Scholes PDE [Black, Fischer and Scholes, Myron, 1973] framework models many pricing problems in finance.
Black-Scholes equation for European options is given by

$$V_\tau = \frac{\sigma^2 S^2}{2} V_{SS} + rSV_S - rV \equiv \mathcal{L}(r)V \tag{1}$$

- S – asset price variable
- τ – backward time variable from maturity T ($\tau = T - t$)
- σ – volatility of asset price
- r – interest rate

Some non-vanilla option pricing problems are obtained by adding terms or modifying existing terms in Equation (1).

Overview

Hamilton-Jacobi-Bellman (HJB) equations model many nonlinear pricing problems in finance.
General form of HJB equations in finance:

$$V_\tau = \sup_Q \left\{ a(S,\tau,Q)V_{SS} + b(S,\tau,Q)V_S + c(S,\tau,Q)V + d(S,\tau,Q) \right\} \tag{2}$$

- Q – control variables
- $aV_{SS} + bV_S + cV + d$ is $\mathcal{L}(\cdot)V$ with additional and/or modified terms
- The above is for short positions. For long positions, sup is replaced by inf.

Overview

We study the following nonlinear pricing problems in computational finance under the Black-Scholes framework

- Unequal Borrowing/Lending rates [Bergman, Yaacov Z, 1995]
- Above problem with stock borrowing fees
  [Duffie, Darrell and Garleanu, Nicolae and Pedersen, Lasse Heje, 2002]
- Stock Borrowing Fee problem (above) with American-style exercise rights
  [Forsyth and Labahn, 2007]

formulated as HJB equations and as nonlinear PDEs.

We consider the solution of the HJB equations with policy iteration [Forsyth and Labahn, 2007]

We derive penalty-like (penalty) iteration algorithms for the solution of the nonlinear PDEs, inspired by [Forsyth and Vetzal, 2002, Y. Chen and C. Christara, 2020].

We only consider short position except for Stock Borrowing Fee problem with American options in the interest of brevity.

Parameters

Table 1 gives the parameters and their values used in our problems.

Note that not all parameters are used in all problems.

Variable name	Symbol	Value
End time	T	1
Space Truncation Boundary	$S_{\max}$	1000
Strike price	K	100
Volatility	σ	0.30
Borrowing interest rate	r_b	0.05
Lending interest rate	r_l	0.03
Stock Borrowing fee	r_f	0.004
Payoff function	$V^*(S)$	abs$(S - K)$
American penalty parameter	$\rho = \epsilon^{-1}$	10^6

Table 1: Parameters used in our problems

Outline

- Overview
- Borrow-Lend
  - Policy Iteration
  - Penalty Iteration
  - Numerical Results
- Stock Borrowing Fees
  - Policy Iteration
  - Penalty Iteration
  - Numerical Results
- Stock Borrowing Fees with American Early Exercise
  - Policy Iteration
  - Double Penalty Iteration
  - Numerical Results
- Summary & Conclusions

Borrow-Lend

Control problem [Forsyth and Labahn, 2007] is

$$V_\tau = \sup_{q\in\{r_l,r_b\}} \left\{ \frac{\sigma^2 S^2}{2} V_{SS} + qSV_S - qV \right\} \equiv \sup_{q\in\{r_l,r_b\}} \{\mathcal{L}(q)\}\,. \tag{3}$$

We write it as the nonlinear PDE

$$\begin{aligned} V_\tau &= \frac{\sigma^2 S^2}{2} V_{SS} + r_b S V_S - r_b V + (r_b - r_l)\max(V - SV_S, 0) \\ &\equiv \mathcal{L}(r_b) + (r_b - r_l)\max(V - SV_S, 0) \end{aligned} \tag{4}$$

We use Crank-Nicolson-Rannacher timestepping and centered finite differences for time and space discretization.

For spatial discretization, we use a nonuniform grid from [Clarke, Nigel and Parrott, Kevin, 1999]. The grid is sufficiently smooth, so second-order convergence is not affected.

We let A denote the matrix that computes the spatial discretization of $a(S,\tau,Q)V_{SS} + b(S,\tau,Q)V_S + c(S,\tau,Q)V$, in this case $\mathcal{L}(q)$.

We solve (3) with a policy iteration algorithm and (4) with a penalty iteration algorithm.

Discretization of HJB Equation

Discretization of Equation (2) leads us to solve, at each timestep j, the following system of equations for v^j, the algorithm's approximation to the solution $V(\tau^j, \cdot)$

$$(I - \theta\Delta\tau A^j(Q))v^j = (I + (1-\theta)\Delta\tau A^{j-1})v^{j-1} + \theta\Delta\tau D^j(Q) + (1-\theta)\Delta\tau D^{j-1} \qquad (5)$$

subject to the maximization condition

$$Q_i = \arg\sup_{Q \in \hat{Q}} [A^j(Q)v^j + D^j(Q)]_i \qquad (6)$$

where A is the discretization of the terms involving a, b, and c in Equation (2) (boundary conditions are taken into account), and $D(Q)$ is the vector of values of d.

Here, d is 0 but it is nonzero for American options.

Discretization of other HJB equations is very similar.

The only difference is the functions a, b, c, and d, which are used in the definition of A and D.

A Policy Iteration algorithm for Borrow-Lend

Algorithm 1 Policy iteration for Borrow-Lend at step j with θ timestepping [Forsyth and Labahn, 2007]

Require: Solve $(I - \theta \Delta\tau A^j(Q))v^j = g^j$
subject to $Q_i = \arg\sup_{Q \in \hat{Q}} [A^j(Q)v^j]_i$
where $g^j = (I + (1-\theta)\Delta\tau A^{j-1})v^{j-1}$

1: Initialize $v^{j,0} = v^{j-1}$ and $Q^{j,0} = Q^{j-1}$
2: **for** $k = 1, \ldots, maxit$ **do**
3:     Solve $(I - \theta\Delta\tau A^j(Q^{j,k-1}))v^{j,k} = g^j$
4:     Compute $Q_i^{j,k} = \arg\sup_{Q \in \hat{Q}} [A^j(Q)v^{j,k}]_i$
5:     **if** stopping criterion satisfied **then**
6:         Break
7:     **end if**
8: **end for**
9: Set $v^j = v^{j,k}$
10: Set $Q^j = Q^{j,k}$

This algorithm requires a matrix-vector product in line 4 for each admissable control.

We compare this with the penalty methods that we introduce.

Penalty matrix definition

We use a penalty matrix to discretize the term $(r_b - r_l)\max(V - SV_S, 0)$ in Equation (4).

Let D_S denote a diagonal matrix with S_i (gridpoints) on its diagonal.

Let T_1 denote the matrix that discretizes V_S with finite differences.

Thus, SV_S is discretized as $D_S T_1$.

We define the penalty matrix $P = P(v)$ as a diagonal matrix with entries defined by

$$P_{ii} = \begin{cases} r_b - r_l & \text{if } v_i > [D_S T_1 v]_i \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Then the term $(r_b - r_l)\max(V - SV_S, 0)$ is discretized as $(P - PD_S T_1)v$.

P is not treated fully implicitly as in [Forsyth and Vetzal, 2002], but in the same nature as A (Crank-Nicolson).

Diagonal Penalty Iteration Algorithm for Borrow-Lend

Algorithm 2 Diagonal penalty iteration for the Borrow-Lend problem at step j, with θ-timestepping

---

Require: Solve $(I - \theta\Delta\tau(A + P^j - P^j D_S T_1))v^j = g^j$
where $g^j = (I + (1-\theta)\Delta\tau(A + P^{j-1} - P^{j-1} D_S T_1))v^{j-1}$

Initialize $v^{j,0} = v^{j-1}$ and $P^{j,0} = P(v^{j-1})$
for $k = 1, \ldots, maxit$ **do**
  Solve $(I - \theta\Delta\tau(A + P^{j,k-1}))v^{j,k} = g^j - \theta\Delta\tau P^{j,k-1} D_S T_1 v^{j,k-1}$
  Compute $P^{j,k} = P(v^{j,k})$
  if stopping criterion satisfied **then**
    Break
  end if
end for
Set $v^j = v^{j,k}$

The nonlinearity arises between $P^j(v^j)$ and v^j.
We approximate P^j with $P^{j,k-1}$ and v^j with $v^{j,k}$ for the V component, but $v^{j,k-1}$ for the SV_S component.
In other words, the penalty matrix P is delayed by 1 iteration.

Tridiagonal Penalty Iteration Algorithm for Borrow-Lend

Algorithm 3 Tridiagonal penalty iteration for the Borrow-Lend problem at step j, with θ-timestepping

---

Require: Solve $(I - \theta\Delta\tau(A + P^j - P^j D_S T_1))v^j = g^j$
    where $g^j = (I + (1-\theta)\Delta\tau(A + P^{j-1} - P^{j-1} D_S T_1))v^{j-1}$
Initialize $v^{j,0} = v^{j-1}$ and $P^{j,0} = P(v^{j-1})$
for $k = 1, \ldots, maxit$ **do**
    Solve $(I - \theta\Delta\tau(A + P^{j,k-1} - P^{j,k-1} D_S T_1))v^{j,k} = g^j$
    Compute $P^{j,k} = P(v^{j,k})$
    if stopping criterion satisfied **then**
        Break
    end if
end for
Set $v^j = v^{j,k}$

The difference from the diagonal penalty iteration is that the term SV_S is applied to the same iteration step.

Stopping Criterion

The stopping criteron in Algorithm 1 is

$$(Q^{j,k-1} = Q^{j,k}) \text{ or } (\max_i \frac{|v_i^{j,k} - v_i^{j,k-1}|}{\max(1, |v_i^{j,k}|)} \leq tol). \tag{8}$$

The stopping criterion in Algorithm 2 is

$$\max_i \frac{|v_i^{j,k} - v_i^{j,k-1}|}{\max(1, |v_i^{j,k}|)} \leq tol, \tag{9}$$

while the stopping criterion in Algorithm 3 is

$$(P^{j,k-1} = P^{j,k}) \text{ or } (\max_i \frac{|v_i^{j,k} - v_i^{j,k-1}|}{\max(1, |v_i^{j,k}|)} \leq tol). \tag{10}$$

Note that, in Algorithm 3, if the equality of the penalty matrices is satisfied, the other part will automatically be satisfied on the next iteration.

Numerical results (comparing choice of grid)

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	102	23.723409	0.00e+00	0.00	0.000000
200	202	202	23.985540	2.62e-01	0.00	0.000000
400	402	402	24.049544	6.40e-02	2.03	24.070221
800	802	802	24.065266	1.57e-02	2.03	24.070385

Table 2: Borrow-Lend problem solved with Policy Iteration, uniform grid

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	102	24.057902	0.00e+00	0.00	0.000000
200	202	203	24.067267	9.36e-03	0.00	0.000000
400	402	406	24.069607	2.34e-03	2.00	24.070386
800	802	809	24.070191	5.85e-04	2.00	24.070386

Table 3: Borrow-Lend problem solved with Policy Iteration, nonuniform grid

Note that nonuniform grid reduces the change by a factor of around 25, subsequently we will use nonuniform grids.

Numerical results (penalty methods)

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	307	24.057902	0.00e+00	0.00	0.000000
200	202	484	24.067267	9.36e-03	0.00	0.000000
400	402	864	24.069607	2.34e-03	2.00	24.070386
800	802	1650	24.070191	5.85e-04	2.00	24.070386

Table 4: Borrow-Lend problem solved with diagonal penalty iteration

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	103	24.057902	0.00e+00	0.00	0.000000
200	202	204	24.067267	9.36e-03	0.00	0.000000
400	402	406	24.069607	2.34e-03	2.00	24.070386
800	802	809	24.070191	5.85e-04	2.00	24.070386

Table 5: Borrow-Lend problem solved with tridiagonal penalty iteration

Diagonal policy iteration doubles the iteration count; subsequently we will have tridiagonal treatment of any penalty terms.

Outline

Stock Borrowing Fees

Control problem:

$$V_\tau = \sup_Q \left\{ \frac{\sigma^2 S^2}{2} V_{SS} + q_3 q_1 (S V_S - V) + (1 - q_3)((r_l - r_f) S V_S - q_2 V) \right\} \quad (11)$$

with $Q = (q_1, q_2, q_3)$, $q_1 \in \{r_l, r_b\}$, $q_2 \in \{r_l, r_b\}$, $q_3 \in \{0, 1\}$, so 8 cases.

Nonlinear PDE:

$$\begin{aligned} V_\tau &= \frac{\sigma^2 S^2}{2} V_{SS} + r_l(S V_S - V) + \max\{(r_b - r_l)(S V_S - V), -r_f S V_S, 0\} \quad (12) \\ &\equiv \mathcal{L}(r_l) + \max\{(r_b - r_l)(S V_S - V), -r_f S V_S, 0\} \end{aligned}$$

Space and time discretization remain the same as in Borrow-Lend problem.

Algorithms are very similar as well.

Policy Iteration

The policy iteration algorithm is very similar to Algorithm 1. In fact, the steps are exactly the same other than the maximization step, which has a different implementation due to the different number of controls.

However, we still have to enumerate all the possible combinations (8 in this case) to find the maximum for each component.

"Curse of Dimensionality": As the number of controls increase, the number of combinations that need to be enumerated increases exponentially – when American options are considered, the number doubles again.

The penalty matrix is very similar to the one for Borrow-Lend, however, there is a max over 3 terms we need to consider, since the term of interest is now

$$\max\{(r_b - r_l)(SV_S - V), -r_f SV_S, 0\} \tag{13}$$

Let now A, P_1 and P_2 the tridiagonal matrices arising from the discretization of $\mathcal{L}(r_l)V$, $(r_b - r_l)(SV_S - V)$ and $-r_f SV_S$, respectively.

Note that $P_1 = (r_b - r_l)(D_S T_1 - I)$ and $P_2 = -r_f D_S T_1$.

Define the tridiagonal penalty matrix $P = P(v^j)$ by

$$P_{i,:} = \begin{cases} 0 & \text{if } [P_1 v^j]_i \leq 0 \text{ and } [P_2 v^j]_i \leq 0 \\ [P_1]_{i,:} & \text{if } [P_1 v^j]_i > 0 \text{ and } [P_1 v^j]_i > [P_2 v^j]_i \\ [P_2]_{i,:} & \text{if } [P_2 v^j]_i > 0 \text{ and } [P_1 v^j]_i \leq [P_2 v^j]_i, \end{cases} \tag{14}$$

For convenience, we have borrowed the colon notation from matlab.

Tridiagonal Penalty Iteration

Due to the better performance of Algorithm 3 compared to Algorithm 2, we have only a tridiagonal penalty iteration.

Algorithm 4 Tridiagonal penalty iteration for the Stock Borrowing Fees problem at step j, with θ-timestepping

Require: Solve $(I - \theta\Delta\tau(A + P(v^j)))v^j = g^j$
  where $g^j = (I + (1-\theta)\Delta\tau(A + P(v^{j-1})))v^{j-1}$

Initialize $v^{j,0} = v^{j-1}$ and $P^{j,0} = P(v^{j-1})$
for $k = 1, \ldots, maxit$ **do**
  Solve $(I - \theta\Delta\tau(A + P^{j,k-1}))v^{j,k} = g^j$
  Compute $P^{j,k} = P(v^{j,k})$
  if stopping criterion satisfied **then**
    Break
  end if
end for
Set $v^j = v^{j,k}$

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	113	24.121945	0.00e+00	0.00	0.000000
200	202	223	24.131388	9.44e-03	0.00	0.000000
400	402	444	24.133747	2.36e-03	2.00	24.134533
800	802	882	24.134336	5.89e-04	2.00	24.134532

Table 6: Stock Borrowing Fee problem solved with Policy Iteration

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	113	24.121945	0.00e+00	0.00	0.000000
200	202	222	24.131388	9.44e-03	0.00	0.000000
400	402	443	24.133747	2.36e-03	2.00	24.134533
800	802	882	24.134336	5.89e-04	2.00	24.134532

Table 7: Stock Borrowing Fee problem solved with Penalty Iteration

Outline

HJB PDE:

$$V_\tau = \sup_{\{\mu,Q\}} \left\{ \frac{\sigma^2 S^2}{2} V_{SS} + q_3(q_1 S V_S - q_1 V) \right. \\ \left. + (1-q_3)((r_l - r_f) S V_S - q_2 V) + \mu \frac{V^* - V}{\epsilon} \right\} \quad (15)$$

with $Q = (q_1, q_2, q_3)$, $q_1 \in \{r_l, r_b\}$, $q_2 \in \{r_l, r_b\}$, $q_3 \in \{0, 1\}$, $\mu \in \{0, 1\}$, so 16 cases.

Penalized PDE:

$$V_\tau = \frac{\sigma^2 S^2}{2} V_{SS} + r_l(S V_S - V) \\ + \max\{(r_b - r_l)(S V_S - V), -r_f S V_S, 0\} \\ + \rho \max\{V^* - V, 0\} \quad (16)$$

where $\rho = 1/\epsilon$ is a large positive parameter.

Long positions

The long position of (15) is more interesting, in part because by replacing one of the sup by inf, we have an HJBI.

HJBI PDE:

$$V_\tau = \sup_\mu \inf_Q \left\{ \frac{\sigma^2 S^2}{2} V_{SS} + q_3(q_1 S V_S - q_1 V) \right.$$
$$\left. + (1 - q_3)((r_l - r_f) S V_S - q_2 V) + \mu \frac{V^* - V}{\epsilon} \right\} \quad (17)$$

with $Q = (q_1, q_2, q_3)$, $q_1 \in \{r_l, r_b\}$, $q_2 \in \{r_l, r_b\}$, $q_3 \in \{0, 1\}$, $\mu \in \{0, 1\}$.

Claimed to be more difficult.

Policy iteration

Same as the previous two cases; not much to discuss for short option. Long option is more interesting.

We compute the $\sup\inf$ by first computing the inf twice over q_1, q_2, q_3 with $\mu = 0$ and $\mu = 1$, and then compute the sup over the inf.

Penalty Matrices

There are two penalty matrices that we use here.

The first is the penalty matrix resulting from the nonlinear terms from the stock borrowing fee problem. Here, we use the same P as defined in (14).

However, we also consider the long position, we note that the PDE is

$$V_\tau = \frac{\sigma^2 S^2}{2} V_{SS} + r_b(SV_S - V) \\ + \min\{(r_l - r_b)(SV_S - V), -(r_b - r_l + r_f)SV_S, 0\} + \rho \max\{V^* - V, 0\} \quad (18)$$

The definitions do not change significantly (other than coefficients). Notably, since the max is replaced by min, all of the inequalities switch.

The other penalty matrix P_A is for American options, introduced in [Forsyth and Vetzal, 2002]:

$$[P_A(v)]_{ii} = \begin{cases} \rho & \text{if } v^* > v \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

P_A is treated fully implicitly

Double Penalty Iteration

Algorithm 5 Double Penalty Iteration for the American Stock Borrowing Fees problem at step j, with θ-timestepping

Require: Solve $[(I - \theta\Delta\tau(A + P(v^j))) + P_A(v^j)]v^j = g^j + P_A(v^j)v^*$
where $g^j = (I + (1-\theta)\Delta\tau(A + P(v^{j-1})))v^{j-1}$

```
Initialize $v^{j,0} = v^{j-1}$ and $P^{j,0} = P(v^{j-1})$
for $k = 1, \ldots, maxit$ do
    Solve $[(I - \theta\Delta\tau(A + P^{j,k-1})) + P_A^{j,k-1}]v^{j,k} = g^j + P_A(v^{j,k-1})v^*$
    Compute $P^{j,k} = P(v^{j,k})$, $P_A^{j,k} = P_A(v^{j,k})$
    if stopping criterion satisfied then
        Break
    end if
end for
Set $v^j = v^{j,k}$
```

Double Penalty Iteration (continued)

Note that in Algorithm 5 on the previous slide,

- P is multiplied with the $\theta\Delta\tau$ with A.
- P_A is not multiplied with the $\theta\Delta\tau$.

We can show the diagonal dominance of the linear system under certain conditions.

Additionally, the stopping criterion used is

$$\left((P^{j,k-1} = P^{j,k}) \text{ and } (P_A^{j,k-1} = P_A^{j,k})\right) \text{ or } \left(\max_i \frac{|v_i^{j,k} - v_i^{j,k-1}|}{\max(1, |v_i^{j,k}|)} \leq tol\right). \tag{20}$$

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	136	24.312760	0.00e+00	0.00	0.000000
200	202	271	24.323903	1.11e-02	0.00	0.000000
400	402	542	24.326884	2.98e-03	1.90	24.327972
800	802	1091	24.327704	8.20e-04	1.86	24.328015

Table 8: American Stock Borrowing Fee problem solved with Policy Iteration

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	137	24.312489	0.00e+00	0.00	0.000000
200	202	273	24.323818	1.13e-02	0.00	0.000000
400	402	545	24.326851	3.03e-03	1.90	24.327959
800	802	1091	24.327690	8.39e-04	1.85	24.328011

Table 9: American Stock Borrowing Fee problem solved with Double Penalty method

Numerical results (Long positions)

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	137	23.069091	0.00e+00	0.00	0.000000
200	202	273	23.079792	1.07e-02	0.00	0.000000
400	402	543	23.082725	2.93e-03	1.87	23.083833
800	802	1075	23.083561	8.36e-04	1.81	23.083894

Table 10: American Stock Borrowing Fee problem solved with Policy Iteration

Nodes	Tstep	Iter	Value	Change	Rate	Pred
100	102	136	23.068281	0.00e+00	0.00	0.000000
200	202	268	23.079231	1.10e-02	0.00	0.000000
400	402	533	23.082250	3.02e-03	1.86	23.083399
800	802	1058	23.083114	8.64e-04	1.80	23.083461

Table 11: American Stock Borrowing Fee problem solved with Double Penalty method

Outline

Summary & Conclusions

- We have seen three different problems with algorithms that use the discrete penalty-like iteration.
- Double Penalty method is similar to [Y. Chen and C. Christara, 2020] for valuation adjustments.
- We can treat different penalty terms separately:
  - American penalty term is treated fully implicitly
  - Penalty term for nonlinearity in Borrow-Lend or Stock Borrowing Fee is treated the same as Crank-Nicolson – ensures second order convergence.
  - It is comparatively difficult to do this with HJB PDEs
- Diagonal Dominance of matrices in (double) penalty methods for Borrow-Lend (Algorithms 3 & 2), Stock Borrowing Fees (Algorithm 4), and Stock Borrowing Fees with American options (Algorithm 5) have all been proved – could lead to monotonicity and convergence – but there is no space here.
- Penalty (PDE) and Policy (HJB) methods require approximately the same number of iterations.
- Penalty methods avoid enumeration of all possible cases in the maximization step of policy iteration
  - This is especially useful as the number of controls increase.

Bergman, Yaacov Z (1995).
Option pricing with differential interest rates.
The Review of Financial Studies, 8(2):475–500.

Black, Fischer and Scholes, Myron (1973).
The pricing of options and corporate liabilities.
Journal of political economy, 81(3):637–654.

Clarke, Nigel and Parrott, Kevin (1999).
Multigrid for American option pricing with stochastic volatility.
Applied Mathematical Finance, 6(3):177–195.

Duffie, Darrell and Garleanu, Nicolae and Pedersen, Lasse Heje (2002).
Securities lending, shorting, and pricing.
Journal of Financial Economics, 66(2-3):307–339.

Forsyth, P. A. and Labahn, G. (2007).
Numerical methods for controlled Hamilton-Jacobi-Bellman PDEs in finance.
J. Comput. Finance, 11(2):1–43.

References II

Forsyth, P. A. and Vetzal, K. R. (2002).
Quadratic convergence for valuing American options using a penalty method.
SIAM J. Sci. Comput., 23(6):2095–2122.

Y. Chen and C. Christara (2020).
Penalty Methods for Bilateral XVA Pricing in European and American Contingent Claims by a PDE Model.
J. Comput. Finance.
to appear.