Today’s Topics

- Penalty Methods
- Variable Timestep Selection and Quadratic Convergence
- Properties of $\mathcal{M}$ Matrices
The Penalty Method: Motivation

There are three sets of constraints:

\[ AV^{n+1} - b^n \geq 0 \]
\[ (V^{n+1} - V^*) \geq 0 \]
\[ (V^{n+1} - V^*) \cdot (AV^{n+1} - b^n) = 0 \]

Penalizing violation of \( V^{n+1} \geq V^* \) as follows:

\[ AV^{n+1} - b^n - \text{Large} \cdot \max(V^* - V^{n+1}, 0) = 0 \]

where Large is a positive constant.
Assume that $V^{n+1}$ satisfies the penalized nonlinear equations above with a large penalty parameter.

Then

$$AV^{n+1} - b^n \geq 0$$

In addition, if $V_i^{n+1} \geq V_i^*$

$$\left( AV^{n+1} - b^n \right)_i = 0$$

If $V_i^{n+1} < V_i^*$

$$V_i^* - V_i^{n+1} = \frac{1}{\text{Large}} \left( AV^{n+1} - b^n \right)_i$$

where is expected to be very small for a large penalty parameter.
How do we solve these nonlinear algebraic equations?
Consider C-N for example.

\[
V^{n+1} = \begin{bmatrix}
V_0^{n+1} \\
V_1^{n+1} \\
\vdots \\
V_m^{n+1}
\end{bmatrix} ; \quad V^n = \begin{bmatrix}
V_0^n \\
V_1^n \\
\vdots \\
V_m^n
\end{bmatrix} ; \quad V^* = \begin{bmatrix}
V_0^* \\
V_1^* \\
\vdots \\
V_m^*
\end{bmatrix}
\]

with matrix $\hat{M}$ defined so that (row $i$)

\[
\left[\hat{M}V^n\right]_i = -\frac{\Delta \tau \alpha_i}{2}V_{i-1}^n + \frac{\Delta \tau (\alpha_i + \beta_i + r)}{2}V_i^n - \frac{\Delta \tau \beta_i}{2}V_{i+1}^n.
\]

Let the diagonal matrix $\tilde{P}(V^{n+1})$ be given by

\[
\tilde{P}(V^{n+1})_{ii} = \begin{cases}
\text{Large} ; & \text{if } V_i^{n+1} < V_i^* \\
0 ; & \text{otherwise}
\end{cases}
\]
then we can write the nonlinear equations as

\[
\begin{bmatrix}
I + \hat{M} + \bar{P}(V^{n+1})
\end{bmatrix} V^{n+1} = \begin{bmatrix}
I - \hat{M}
\end{bmatrix} V^n + \begin{bmatrix}
\bar{P}(V^{n+1})
\end{bmatrix} V^* 
\]
Penalty American Constraint Iteration

Let \((V^{n+1})^0 = V^n\) and \((V^{n+1})^k\) be the \(k^{th}\) estimate for \(V^{n+1}\).

For \(k = 0, \ldots \) until convergence

\[
\begin{bmatrix}
I + \hat{M} + \bar{P}((V^{n+1})^k)
\end{bmatrix}
\begin{bmatrix}
(V^{n+1})^{k+1}
\end{bmatrix}
= \\
\begin{bmatrix}
I - \hat{M}
\end{bmatrix}
\begin{bmatrix}
V^n
\end{bmatrix} + 
\bar{P}((V^{n+1})^k)V^*
\]

if \(\max_i \frac{|(V_i^{n+1})^{k+1} - (V_i^{n+1})^k|}{\max(1,|(V_i^{n+1})^{k+1}|)} < tol\) quit

EndFor

- Note that matrix must be refactored every iteration
How big should \textit{Large} be?

In the exercise region

\[ V_i^{n+1} = V_i^* - \epsilon \]

As \textit{Large} $\to \infty$, the solution becomes more accurate, i.e. $\epsilon \to 0$.

- But numerical roundoff may cause problems if $\textit{Large} > 1/(\textit{machine eps})$
It can be shown, pp 142 in course notes, that as $Large \rightarrow \infty$, then, in the exercise region ($V_i^* \neq 0$)

\[
\left| \frac{V_i^* - V_i^{n+1}}{V_i^*} \right| \sim \frac{1}{Large}
\]

Therefore, if we want a solution which has a relative error $tol$ (the convergence tolerance in the penalty algorithm), we choose

\[
Large \sim \frac{1}{tol}.
\]

So, $Large$ can be specified (not arbitrary).
Nonlinear Iterations: Penalty Method

- American Put, \( S = 100, K = 100, \sigma = .2, T = .25, r = .10 \)
- C-N timestepping (two steps implicit at start, then C-N)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>( \Delta \tau )</th>
<th>( \sigma = .2 )</th>
<th>( \sigma = .8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>.004</td>
<td>2.3</td>
<td>2.2</td>
</tr>
<tr>
<td>200</td>
<td>.002</td>
<td>2.3</td>
<td>2.2</td>
</tr>
<tr>
<td>400</td>
<td>.001</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td>800</td>
<td>.0005</td>
<td>2.2</td>
<td>2.3</td>
</tr>
</tbody>
</table>
• Iterations/timestep roughly constant.
• Not sensitive to $\sigma$. 
American Option Test

- We will carefully analyze the convergence of American options as timestep, grid size reduced
- We use Rannacher smoothing in all examples
- $T = .25$, $r = .10$, $K = 100$, $S = 100$, $\sigma = .2$
- American put
- $flops = $ number of multiplies/divides
<table>
<thead>
<tr>
<th>Nodes</th>
<th>time steps</th>
<th>Nonlin itns</th>
<th>Value</th>
<th>Change</th>
<th>Ratio</th>
<th>Work (flops)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>explicit constraint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>25</td>
<td>25</td>
<td>3.04600</td>
<td>.01449</td>
<td></td>
<td>8.3 \times 10^3</td>
</tr>
<tr>
<td>109</td>
<td>50</td>
<td>50</td>
<td>3.06049</td>
<td>.00549</td>
<td>2.6</td>
<td>3.3 \times 10^4</td>
</tr>
<tr>
<td>217</td>
<td>100</td>
<td>100</td>
<td>3.06598</td>
<td>.00224</td>
<td>2.5</td>
<td>1.3 \times 10^5</td>
</tr>
<tr>
<td>433</td>
<td>200</td>
<td>200</td>
<td>3.06822</td>
<td>.00100</td>
<td>2.2</td>
<td>5.2 \times 10^5</td>
</tr>
<tr>
<td>865</td>
<td>400</td>
<td>400</td>
<td>3.06922</td>
<td>.00035</td>
<td>2.8</td>
<td>2.1 \times 10^6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implicit constraint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>25</td>
<td>58</td>
<td>3.05607</td>
<td>.00948</td>
<td>3.2</td>
<td>1.7 \times 10^4</td>
</tr>
<tr>
<td>109</td>
<td>50</td>
<td>117</td>
<td>3.06555</td>
<td>.00299</td>
<td>3.0</td>
<td>6.7 \times 10^4</td>
</tr>
<tr>
<td>217</td>
<td>100</td>
<td>234</td>
<td>3.06854</td>
<td>.00099</td>
<td>3.0</td>
<td>2.7 \times 10^5</td>
</tr>
<tr>
<td>433</td>
<td>200</td>
<td>471</td>
<td>3.06953</td>
<td>.00035</td>
<td>3.0</td>
<td>1.1 \times 10^6</td>
</tr>
<tr>
<td>865</td>
<td>400</td>
<td>937</td>
<td>3.06988</td>
<td>.00035</td>
<td>2.8</td>
<td>4.3 \times 10^6</td>
</tr>
</tbody>
</table>
American Option Test

- $T = .25, r = .10, K = 100, S = 100, \sigma = .8$
- American put
- \textit{flops} = number of multiplies/divides

<table>
<thead>
<tr>
<th>Nodes</th>
<th>time steps</th>
<th>Nonlin itns</th>
<th>Value</th>
<th>Change</th>
<th>Ratio</th>
<th>Work (flops)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>explicit constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>25</td>
<td>25</td>
<td>14.61682</td>
<td>.4002</td>
<td>2.9</td>
<td>$1.0 \times 10^4$</td>
</tr>
<tr>
<td>135</td>
<td>50</td>
<td>50</td>
<td>14.65685</td>
<td>.01360</td>
<td>1.6</td>
<td>$4.1 \times 10^4$</td>
</tr>
<tr>
<td>269</td>
<td>100</td>
<td>100</td>
<td>14.67045</td>
<td>.00497</td>
<td>2.7</td>
<td>$6.5 \times 10^5$</td>
</tr>
<tr>
<td>537</td>
<td>200</td>
<td>200</td>
<td>14.67542</td>
<td>.00196</td>
<td>2.5</td>
<td>$2.6 \times 10^6$</td>
</tr>
<tr>
<td>1073</td>
<td>400</td>
<td>400</td>
<td>14.67738</td>
<td>.000196</td>
<td>2.5</td>
<td>$2.6 \times 10^6$</td>
</tr>
<tr>
<td></td>
<td>implicit constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>25</td>
<td>56</td>
<td>14.62708</td>
<td>.03511</td>
<td>3.2</td>
<td>$2.1 \times 10^4$</td>
</tr>
<tr>
<td>135</td>
<td>50</td>
<td>112</td>
<td>14.66219</td>
<td>.01105</td>
<td>3.5</td>
<td>$8.5 \times 10^4$</td>
</tr>
<tr>
<td>269</td>
<td>100</td>
<td>226</td>
<td>14.67324</td>
<td>.00362</td>
<td>3.1</td>
<td>$3.5 \times 10^5$</td>
</tr>
<tr>
<td>537</td>
<td>200</td>
<td>461</td>
<td>14.67686</td>
<td>.00127</td>
<td>2.9</td>
<td>$1.5 \times 10^6$</td>
</tr>
<tr>
<td>1073</td>
<td>400</td>
<td>940</td>
<td>14.67813</td>
<td>.000127</td>
<td>2.9</td>
<td>$6.0 \times 10^6$</td>
</tr>
</tbody>
</table>
Why is convergence not quadratic?

Theoretical analysis of behavior of put near exercise boundary, near \( \tau = T - t = 0 \), indicates

- The option value around \( \tau = 0 \) and the early exercise curve changes very quickly, see pp 137 course notes.

- If we take timestep \( \Delta\tau \) such that:

\[
\max_i (|V_{i}^{n+1} - V_{i}^{n}|) \simeq d
\]

where \( d \) is some constant, then the error \( O(d^2) \) can be achieved (i.e., the quadratic convergence rate can be achieved). For detailed (asymptotical) arguments, see (20.5), pp 166168, course notes.
The variable timestep selector we described before can achieves (1).

This suggests that quadratic convergence can be achieved using variable timestep selector.

To check quadratic convergence, use the variable timestep selector and at each refinement

- double the number of grid nodes in $S$
- Reduce $d_{norm}$ by half (roughly double the number of time steps)
- Reduce the initial time stepstep $\Delta \tau^0$ by 4
Same examples as before, now with timestep selector.

\( \sigma = .2 \)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>time steps</th>
<th>Nonlin itns</th>
<th>Value</th>
<th>Change</th>
<th>Ratio</th>
<th>Work (flops)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>explicit constraint</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>18</td>
<td>18</td>
<td>3.04499</td>
<td></td>
<td></td>
<td>9.9 \times 10^3</td>
</tr>
<tr>
<td>109</td>
<td>33</td>
<td>33</td>
<td>3.05825</td>
<td>.01326</td>
<td></td>
<td>3.6 \times 10^4</td>
</tr>
<tr>
<td>217</td>
<td>63</td>
<td>63</td>
<td>3.06425</td>
<td>.00600</td>
<td>2.2</td>
<td>1.4 \times 10^5</td>
</tr>
<tr>
<td>433</td>
<td>122</td>
<td>122</td>
<td>3.06717</td>
<td>.00292</td>
<td>2.1</td>
<td>5.3 \times 10^5</td>
</tr>
<tr>
<td>865</td>
<td>239</td>
<td>239</td>
<td>3.06863</td>
<td>.00146</td>
<td>2.0</td>
<td>2.1 \times 10^6</td>
</tr>
<tr>
<td><strong>implicit constraint</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>18</td>
<td>45</td>
<td>3.06403</td>
<td></td>
<td></td>
<td>1.5 \times 10^4</td>
</tr>
<tr>
<td>109</td>
<td>33</td>
<td>85</td>
<td>3.06867</td>
<td>.00464</td>
<td></td>
<td>5.7 \times 10^4</td>
</tr>
<tr>
<td>217</td>
<td>63</td>
<td>164</td>
<td>3.06975</td>
<td>.00108</td>
<td>4.3</td>
<td>2.2 \times 10^5</td>
</tr>
<tr>
<td>433</td>
<td>122</td>
<td>322</td>
<td>3.07002</td>
<td>.00027</td>
<td>4.0</td>
<td>8.5 \times 10^5</td>
</tr>
<tr>
<td>865</td>
<td>239</td>
<td>636</td>
<td>3.07008</td>
<td>.00006</td>
<td>4.5</td>
<td>3.2 \times 10^6</td>
</tr>
</tbody>
</table>

- Quadratic convergence!
- Also quadratic convergence for \( \sigma = .8 \)
Delta for American Option: Crank-Nicolson, course notes
pp170
American Put Option Gamma
American Option Delta : Fully Implicit
Summary: American Options

- Explicit handling of constraint
  - Delta not continuous at exercise boundary
  - Converges only at $O(\Delta \tau)$ rate
  - C-N wasted (do not get second order convergence)

- Implicit handling of constraint
  - Do not get quadratic convergence with constant timesteps
  - Need to use timestep selector for quadratic convergence

- Implicit constraint → solve nonlinear set of algebraic equations

- PSOR methods
  - Poor complexity, even if no American constraint triggered
  - Poor performance for multi-factor options
Penalty Method

- Converges in one iteration if no American constraint triggered
- Experiments show \# iterations per timestep are constant as \( \Delta S, \Delta \tau \to 0 \)
- Easy to generalize to multi-factor case
- Standard sparse matrix software used to solve linear equations
**M-matrices**

Recall that:

\[
V^{n+1} = \begin{bmatrix}
V_0^{n+1} \\
V_1^{n+1} \\
\vdots \\
V_m^{n+1}
\end{bmatrix}; \quad V^n = \begin{bmatrix}
V_0^n \\
V_1^n \\
\vdots \\
V_m^n
\end{bmatrix},
\]

(2)

Let \( M \) be the tridiagonal matrix with entries

\[
[MV^n]_i = -\Delta \tau \alpha_i V_{i-1}^n + \Delta \tau (\alpha_i + \beta_i + r) V_i^n - \Delta \tau \beta_i V_{i+1}^n
\]

Fully implicit timestepping can be written as

\[
[I + M]V^{n+1} = V^n.
\]
\textbf{M-matrix Definition}

\textbf{DEFINITION.} A matrix \( Q \) is an \( \mathcal{M} \) matrix if

\[
[Q]_{ii} > 0 ; \quad \forall i
\]
\[
[Q]_{ij} \leq 0 ; \quad i \neq j
\]  \hspace{1cm} (3)

and

\[
[Q]_{ii} \geq -\sum_{j \neq i} [Q]_{ij} ; \quad \forall i
\]  \hspace{1cm} (4)

with strict inequality in at least one row.
Theorem 1 (Inverse of an $\mathcal{M}$ matrix) If a matrix $Q$ is an $\mathcal{M}$ matrix, then

$$Q^{-1} \geq 0$$

$$\text{diag}(Q^{-1}) > 0.$$

(5)

Proof. See Varga(1962,2000), *Matrix Iterative Analysis* □

- All entries of the inverse of an $\mathcal{M}$ matrix are nonnegative
- Diagonal entries of the inverse of an $\mathcal{M}$ matrix are strictly positive

$\mathcal{M}$-matrix property is essential in convergence analysis.