Today’s Topics

- Timestep Selection
- Explicit Evaluation of American Constraints
- Implicit Evaluation of American Constraints
- Penalty Methods
Timestep Selection

Constant timesteps are often used, but:

- Small timesteps required after discrete barrier observation to get accurate resolution of very rapid changes

- Long term contracts (e.g. pension guarantees, bonds)
  - After initial transients have died out (due to payoff) large timesteps can be taken.
A simple (heuristic), but effective timestep selector, which uses only current timestep information to predict the next timestep is the following

\[
\Delta \tau^{n+2} = \left[ \frac{\text{dnorm}}{\text{MaxRelChange}} \right] \Delta \tau^{n+1}
\]

\(\Delta \tau^{n+1} = \) current timestep

\(\Delta \tau^{n+2} = \) new timestep

\(\text{dnorm} = \) user specified target relative change

\(\text{MaxRelChange} = \) maximum change over timestep
MaxRelChange =
\[
\max_i \left[ \frac{|V(S_i, \tau^n + \Delta\tau^{n+1}) - V(S_i, \tau^n)|}{\max(D, |V(S_i, \tau^n + \Delta\tau^{n+1})|, |V(S_i, \tau^n)|)} \right]
\]

- This is the maximum change in $V$ which occurred over the timestep, over all nodes
- Normalized, so that it is a relative change
- The scale factor $D$ simply prevents small timesteps when the solution is very small. Depends on units ($D = 1$ fine for dollars).
Timestep Selection

\[ \Delta \tau^{n+2} = \left[ \frac{\text{dnorm}}{\text{MaxRelChange}} \right] \Delta \tau^{n+1} \]

So, what is this doing?

\[ \left[ \frac{\text{MaxRelChange}}{\Delta \tau^{n+1}} \right] \Delta \tau^{n+2} = \text{dnorm} \]

⇒ choose \( \Delta \tau^{n+2} \) so that the relative change equals the target \( \text{dnorm} \). (The first term approximates the derivative)
• Given an initial timestep size, if the user specifies a value of $\text{dnorm} = .05$ say

• New timesteps are selected so that the maximum relative change will be about .05 in each timestep.

• Can be shown that if $\text{dnorm} \to 0$, then truncation error tends to zero.

Note: In assignment 3, for convergence rate test, $\text{dnom}$ needs to be halved when $\Delta S$ is halved.
• If the solution changes rapidly, small timesteps taken; if the solution changes slowly, large timesteps taken.

• Note: no timestep selector will do anything sensible near a discrete barrier, turn off checking near these points
Summary

• For normal B-S equation, first order upstream rarely necessary for positive coefficients

• Positive coefficient method ensures that positive payoffs $\rightarrow$ positive option value (finite grid size, timesteps)

• Fully implicit is positive coefficient, but only converges at $O(\Delta \tau)$ rate

• C-N is not positive coefficient if timestep $> \text{twice explicit timestep}$ size

• Non-smooth payoffs may cause oscillations in solution, slow convergence for C-N
• Cure:
  – Two steps implicit, then C-N
  – For discontinuous payoffs, smoothing required
  – No smoothing required if payoff continuous but non-smooth, as long as there is a node at $S = K$
  – Second order convergence, not positive coefficient, but seems to work well

• Couple with timestep selector: small initial steps help to smooth out rough payoff
American Options

Suppose we want to price an American option, with payoff

\[ \text{Payoff} = V^*(S, t). \]

Formally, this can be stated as a partial differential LCP:

\[
V_\tau - \left( \frac{\sigma^2}{2} S^2 V_{SS} + rSV_S - rV \right) \geq 0 \\
(V - V^*) \geq 0 \\
(V - V^*) \left( V_\tau - \left( \frac{\sigma^2}{2} S^2 V_{SS} + rSV_S - rV \right) \right) = 0
\]
Matrix Form of Equations

Define the vectors

\[ V^{n+1} = \begin{bmatrix} V_{0}^{n+1} \\ V_{1}^{n+1} \\ \vdots \\ V_{m}^{n+1} \end{bmatrix} ; \quad V^{n} = \begin{bmatrix} V_{0}^{n} \\ V_{1}^{n} \\ \vdots \\ V_{m}^{n} \end{bmatrix}, \quad (1) \]

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

\[
[ MV^{n} ]_{i} = -\Delta \tau \alpha_{i} V_{i-1}^{n} + \Delta \tau (\alpha_{i} + \beta_{i} + r) V_{i}^{n} - \Delta \tau \beta_{i} V_{i+1}^{n}
\]

Explicit (European): \[ V^{n+1} = (I - M)V^{n} \]

Implicit (European): \[ [I + M]V^{n+1} = V^{n} \]
American option pricing via solving LCP (explicit):

\[ V^{n+1} - (I - M)V^n \geq 0 \]

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

\[ (V^{n+1} - V^*) \cdot (V^{n+1} - (I - M)V^n) = 0 \]

for notational simplicity \( V^* \) here is a vector of payoff at the \((m+1)\) grid points.

How can we solve this LCP?
Fully implicit and CN can be written as a finite dimensional linear LCP

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

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

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

The complexity of the problem depends, in general, on properties of \( A \).

It is a NP-hard problem for a general matrix \( A \).
Explicit Evaluation of American Constraint

Fully implicit case:

- Given old solution \( V^n_i \), solve for \( \hat{V}^{n+1}_i \) from
  \[
  [I + M] \hat{V}^{n+1} = V^n.
  \]

- Then, apply the constraint
  For \( i=0, \ldots, m \)
  \[
  V_i^{n+1} := \max(V_i^*, \hat{V}^{n+1}_i)
  \]
  EndFor
But:

- Solution is in an inconsistent state after constraint explicitly imposed
  - BS equation (or inequality) does not generally hold
  - Delta not continuous across early exercise boundary
  - If C-N timestepping used
    \[ \rightarrow \] Only first order convergence
    No benefit from C-N timestepping
Explicit vs. Implicit American Constraint

• American Put, $S = 100$, $K = 100$, $\sigma = .2$, $T = .25$, $r = .10$

• Rannacher C-N timestepping (two steps implicit at start, then C-N)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>$\Delta \tau$</th>
<th>Explicit American Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>.004</td>
<td>3.05060</td>
</tr>
<tr>
<td>200</td>
<td>.002</td>
<td>3.06037</td>
</tr>
<tr>
<td>400</td>
<td>.001</td>
<td>3.06519</td>
</tr>
<tr>
<td>800</td>
<td>.0005</td>
<td>3.06761</td>
</tr>
</tbody>
</table>

• Explicit constraint $\rightarrow$ linear convergence
Implicit Constraint: American Option

Methods for solving a finite dimensional LCP:

- projected SOR iterative methods
- $UL$ method which work for constant volatility and standard options (Brennan and Schwartz, 1977)
- Newton type optimization methods (Coleman, Li, Verma, 2002)
- Penalty methods (Forsyth, Vetzal, 2002)
Penalty Methods

- Solve nonlinear discretized algebraic equations iteratively.
- Generalized to multi-factor models.
- Each iteration generates a well-behaved sparse matrix, which can be solved using standard direct or iterative methods.
- Convergence to correct solution is ensured.
The Penalty Method: Motivation

There are three sets of constraints:

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

Approximate LCP: solving one nonlinear system of equations by penalizing violation of \( V^{n+1} \geq V^* \)

\[ AV^{n+1} - b^n - \text{Large} \cdot \max(V^* - V^{n+1}, 0) = 0 \]  \hfill (2)

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 $V_i^{n+1} \geq V_i^*$

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

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

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

where is expected to be very small for a large penalty parameter.
To solve
\[ AV^{n+1} - b^n - \text{Large} \cdot \max(V^* - V^{n+1}, 0) = 0 \]
Let the diagonal matrix \( \bar{P}(V^{n+1}) \) be given by
\[ (\bar{P}(V^{n+1}))_{ii} = \begin{cases} \text{Large} & \text{if } V^{n+1}_i < V^*_i \\ 0 & \text{otherwise} \end{cases} \]
then we can write the nonlinear equations as
\[ (A + \bar{P}(V^{n+1})) V^{n+1} = b^n + \bar{P}(V^{n+1})V^* \]
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 \( i \))

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

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

\[
\bar{P}(V^{n+1})_{ii} = \begin{cases}
\text{Large} & \text{if } V_i^{n+1} < V_i^* \\
0 & \text{otherwise}
\end{cases}
\]
then we can write the nonlinear equation (2) as

\[
\left[ I + \hat{M} + \tilde{P}(V^{n+1}) \right] V^{n+1} = \left[ I - \hat{M} \right] V^n + \left[ \tilde{P}(V^{n+1}) \right] V^* 
\]
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

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

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 *Large* be?

In the exercise region

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

As *Large* \( \to \infty \), the solution becomes more accurate, i.e. \( \epsilon \to 0 \).

- But numerical roundoff may cause problems if

  *Large* \( \geq 1/(\text{machine eps}) \)
It can be shown that as $Large \to \infty$, then, in the exercise region ($V_i^* \neq 0$)

$$\left| \frac{V_i^* - V_i^{n+1}}{V_i^*} \right| \approx \frac{1}{Large}$$

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

$$Large \approx \frac{1}{tol}.$$ 

So, $Large$ is well defined (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>
• If $\Delta S = \mathcal{O}(\Delta \tau)$, iterations/timestep roughly constant.
  → Contrast with other relaxation methods where iterations/timestep increases as $\Delta S, \Delta \tau \to 0$

• Not sensitive to $\sigma$. 
<table>
<thead>
<tr>
<th>Nodes</th>
<th>$\Delta \tau$</th>
<th>Implicit American Constraint</th>
<th>Explicit American Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>.004</td>
<td>3.06704</td>
<td>3.05060</td>
</tr>
<tr>
<td>200</td>
<td>.002</td>
<td>3.06910</td>
<td>3.06037</td>
</tr>
<tr>
<td>400</td>
<td>.001</td>
<td>3.06975</td>
<td>3.06519</td>
</tr>
<tr>
<td>800</td>
<td>.0005</td>
<td>3.06997</td>
<td>3.06761</td>
</tr>
</tbody>
</table>