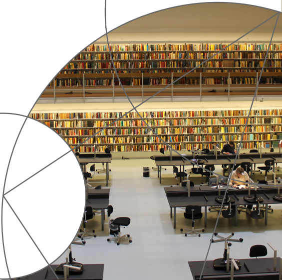
UNIVERSITY OF COPENHAGEN

Numerical pricing of Financial options

with simple Finite Difference Methods

Jens Hugger and Sima Mashayekhi
Department of Mathematical Sciences



April 2, 2014

Outline

1. Presentation of the problem and the BS-model
2. Visualisation of solution and error
3. Numerical issues
4. K_α-optimization
5. Rannacher time stepping
6. Mesh grading
7. Future works

European options

Option: A contract based on some underlying asset [eg. a stock] that gives you [the buyer/holder] **the right but not the obligation** to do something sometime in the future which may cost me [the seller] some money.

European Option: When "sometime in the future" is at a specific Expiration time T and the "something" that you may do cost me some money depending only on the price of the underlying asset at time T, i.e.

A contract based on some underlying asset [eg. a stock] that gives you [the buyer] **the right but not the obligation** to do something at expiration time T which may cost me [the seller] some money depending on the price of the underlying asset at time T.

Good thing about European Options: We know the exact solution, i.e. the fair price $V(S, t)$ that the option should cost the buyer at any time t as a function of the price S of the underlying asset at time t.

Types of European options

The "something to do" distinguishes types of European options:

Three examples:

- A **Call Option** (C) gives the holder the right to **buy** the underlying asset S from the seller at expiration time T for a certain Strike price K.
- A **Put Option** (P) gives the holder the right to **sell** the underlying asset S to the seller at time T for the strike price K.
- A **Bet Option** (Digital Call Option/Cash or nothing option) (B) gives the holder a lump sum B from the seller if at expiration time the price of S is K or more.

The Put-Call-parity: $V^P(S,t) = V^C(S,t) - S + Ke^{-r(T-t)}$ means that computing both call and put is somewhat of a waste of time.

Black-Scholes Model for valuing Options

Suppose that we have a European option (whose value $V(S,t)$ depends only on S and t). No matter what type (call, put, bet or other), the *Black-Scoles model* is the following partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \text{ for } (S,t) \in \Omega_\infty \tag{1}$$

Where

- $\Omega_\infty = (0,\infty) \times (0,T)$, and $V: \ (S,t) \in \bar{\Omega}_\infty \to \mathcal{R}, \ V \in \mathcal{C}^{2,1}(\Omega_\infty)$
- σ is the volatility of the underlying asset
- T is the expiration time
- r is the interest rate

The type enters in the Terminal condition setting the value $V(S,T)$ depending on things like

- K is the exercise price
- B is the bet amount

Black-Scholes Model for valuing Options

Suppose that we have a European option (whose value $V(S,t)$ depends only on S and t). No matter what type (call, put, bet or other), the *Black-Scoles model* is the following partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \text{ for } (S,t) \in \Omega_\infty \quad (1)$$

Where

- $\Omega_\infty = (0,\infty) \times (0,T)$,
  and $V: \ (S,t) \in \bar{\Omega}_\infty \to \mathcal{R}, \ V \in \mathcal{C}^{2,1}(\Omega_\infty)$
- σ is the volatility of the underlying asset
- T is the expiration time
- r is the interest rate

The type enters in the Terminal condition setting the value $V(S,T)$ depending on things like

- K is the exercise price
- B is the bet amount

Black-Scholes Model for valuing Options

Suppose that we have a European option (whose value $V(S,t)$ depends only on S and t). No matter what type (call, put, bet or other), the *Black-Scoles model* is the following partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \text{ for } (S,t) \in \Omega_\infty \qquad (1)$$

Where

- $\Omega_\infty = (0,\infty) \times (0,T)$,
  and $V: \ (S,t) \in \bar{\Omega}_\infty \to \mathcal{R}, \ V \in \mathcal{C}^{2,1}(\Omega_\infty)$
- σ is the volatility of the underlying asset
- T is the expiration time
- r is the interest rate

The type enters in the Terminal condition setting the value $V(S,T)$ depending on things like

- K is the exercise price
- B is the bet amount

Terminal and Boundary Conditions

- $V(S,T) = \kappa(S)$ where $\kappa^{C/P/B}(S)$ is given by:

$$\kappa^C(S) = \max\{S-K, 0\} \text{ for the } \textit{call option}$$
$$\kappa^P(S) = \max\{K-S, 0\} \text{ for the } \textit{put option}$$
$$\kappa^B(S) = \begin{cases} B & \text{for } S-K \geq 0 \\ 0 & \text{for } S-K < 0 \end{cases} \quad \text{for the } \textit{bet option}$$

If $S=0$ (bancruptcy) the value is the back-discounted payoff at time T:

- $V(0,t) = \kappa(0)e^{-r(T-t)}$ (Bancruptcy condition)

For numerical computations it is convenient to have a bounded computational domain $S \in (0, S_{max})$. Boundary conditions can be derived for $S \to \infty$ and then "moved" to $S_{\max} >> K$:

- $$V(S_{\max}, t) = \begin{cases} V^C(S_{\max}, t) \simeq S_{\max} - Ke^{-r(T-t)} & \textit{(call option)} \\ V^P(S_{\max}, t) \simeq 0 & \textit{(put option)} \\ V^B(S_{\max}, t) \simeq Be^{-r(T-t)} & \textit{(bet option)} \end{cases}$$

Terminal and Boundary Conditions

- $V(S,T) = \kappa(S)$ where $\kappa^{C/P/B}(S)$ is given by:

$$\begin{aligned}
\kappa^C(S) &= \max\{S-K,0\} \text{ for the } \textit{call option} \\
\kappa^P(S) &= \max\{K-S,0\} \text{ for the } \textit{put option} \\
\kappa^B(S) &= \begin{cases} B & \text{for } S-K \geq 0 \\ 0 & \text{for } S-K < 0 \end{cases} \quad \text{for the } \textit{bet option}
\end{aligned}$$

If $S=0$ (bancruptcy) the value is the back-discounted payoff at time T:

- $V(0,t) = \kappa(0)e^{-r(T-t)}$ (Bancruptcy condition)

For numerical computations it is convenient to have a bounded computational domain $S \in (0, S_{max})$. Boundary conditions can be derived for $S \to \infty$ and then "moved" to $S_{\max} >> K$:

- $V(S_{\max},t) = \begin{cases} V^C(S_{\max},t) \simeq S_{\max} - Ke^{-r(T-t)} & \textit{(call option)} \\ V^P(S_{\max},t) \simeq 0 & \textit{(put option)} \\ V^B(S_{\max},t) \simeq Be^{-r(T-t)} & \textit{(bet option)} \end{cases}$

Terminal and Boundary Conditions

- $V(S,T) = \kappa(S)$ where $\kappa^{C/P/B}(S)$ is given by:

$$\begin{aligned}
\kappa^C(S) &= \max\{S-K,0\} \text{ for the } \textit{call option} \\
\kappa^P(S) &= \max\{K-S,0\} \text{ for the } \textit{put option} \\
\kappa^B(S) &= \begin{cases} B & \text{for } S-K \geq 0 \\ 0 & \text{for } S-K < 0 \end{cases} \text{ for the } \textit{bet option}
\end{aligned}$$

If $S=0$ (bancruptcy) the value is the back-discounted payoff at time T:

- $V(0,t) = \kappa(0)e^{-r(T-t)}$ (Bancruptcy condition)

For numerical computations it is convenient to have a bounded computational domain $S \in (0, S_{max})$. Boundary conditions can be derived for $S \to \infty$ and then "moved" to $S_{\max} >> K$:

- $$V(S_{\max},t) = \begin{cases} V^C(S_{\max},t) \simeq S_{\max} - Ke^{-r(T-t)} & \textit{(call option)} \\ V^P(S_{\max},t) \simeq 0 & \textit{(put option)} \\ V^B(S_{\max},t) \simeq Be^{-r(T-t)} & \textit{(bet option)} \end{cases}$$

Properties of the solution

The Black-Scholes PDE is a standard convection-diffusion equation and can be transformed smoothly into the heat equation:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \text{ for } (x, \tau) \in \omega_\infty = (-\infty, \infty) \times (0, T) \tag{2}$$

which is wellposed with only a reasonable initial condition (smooth transformation of the terminal condition from BS).

Note 1: The terminal conditions for the call, put and bet options have **singularities in the first, first and zero'th derivative** respectively. This means "numerical trouble".

Note 2: Numerical solution of the heat equation version of BS gives the same problems (singular initial condition) as the convection-diffusion version plus **additional problems since also the left boundary condition must be approximated**.
Hence numerical solution of the heat equation version is discouraged. The heat version is only for theoretical purposes.

Properties of the solution

The Black-Scholes PDE is a standard convection-diffusion equation and can be transformed smoothly into the heat equation:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \text{ for } (x, \tau) \in \omega_\infty = (-\infty, \infty) \times (0, T) \tag{2}$$

which is wellposed with only a reasonable initial condition (smooth transformation of the terminal condition from BS).

Note 1: The terminal conditions for the call, put and bet options have **singularities in the first, first and zero'th derivative** respectively. This means "numerical trouble".

Note 2: Numerical solution of the heat equation version of BS gives the same problems (singular initial condition) as the convection-diffusion version plus **additional problems since also the left boundary condition must be approximated**.
Hence numerical solution of the heat equation version is discouraged. The heat version is only for theoretical purposes.

Properties of the solution

The Black-Scholes PDE is a standard convection-diffusion equation and can be transformed smoothly into the heat equation:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \text{ for } (x, \tau) \in \omega_\infty = (-\infty, \infty) \times (0, T) \tag{2}$$

which is wellposed with only a reasonable initial condition (smooth transformation of the terminal condition from BS).

Note 1: The terminal conditions for the call, put and bet options have **singularities in the first, first and zero'th derivative** respectively. This means "numerical trouble".

Note 2: Numerical solution of the heat equation version of BS gives the same problems (singular initial condition) as the convection-diffusion version plus **additional problems since also the left boundary condition must be approximated**.
Hence numerical solution of the heat equation version is discouraged. The heat version is only for theoretical purposes.

Visualization of solution - Put-Call parity

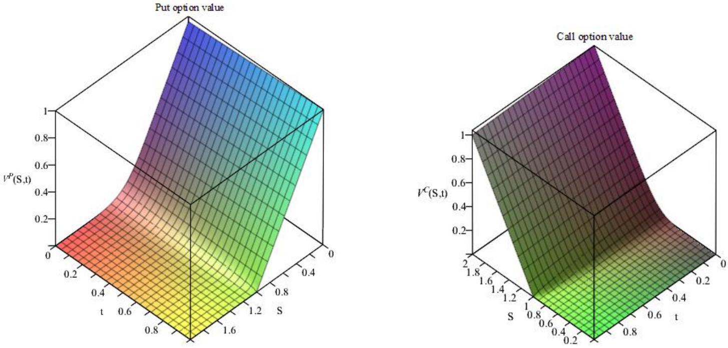


Figure: Exact solution $V^P(S,t)$ for put (left) and call (right) option, with $T = 1$, $K = 1$, $\sigma = 0.2$ and $r = 0.04$. Recall the put-call parity: $V^P(S,t) = V^C(S,t) - S + Ke^{-r(T-t)}$.

From here on, we shall stick to the call and the bet options.

Visualization of solution - Call and Bet

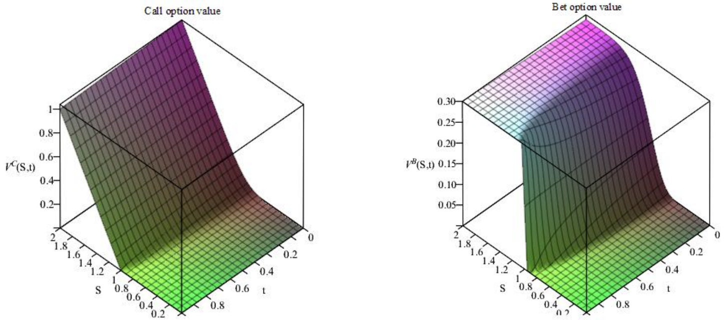
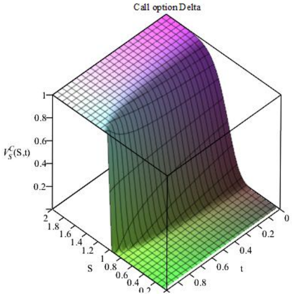


Figure: Exact solution $V(S,t)$ for call (left) and bet (right) option, with $T = 1$, $K = 1$, $\sigma = 0.2$, $r = 0.04$ and $B = 0.3$.

Approximations can be found with standard finite difference schemes on standard laptop PC's with maximal absolute errors of 0.0001 for put and call and 0.001 for bet. Such errors are not visible to the naked eye.

Visualization of Delta $\Delta = \frac{\partial V}{\partial S}$ - Call and Bet



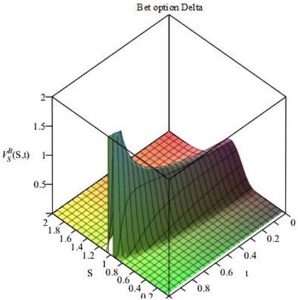


Figure: Exact Delta $\frac{\partial V}{\partial S}$ for call (left) and bet (right) option, with $T = 1$, $K = 1$, $\sigma = 0.2$, $r = 0.04$ and $B = 0.3$.

Visualization of Gamma $\Gamma = \frac{\partial^2 V}{\partial S^2}$ - Call and Bet

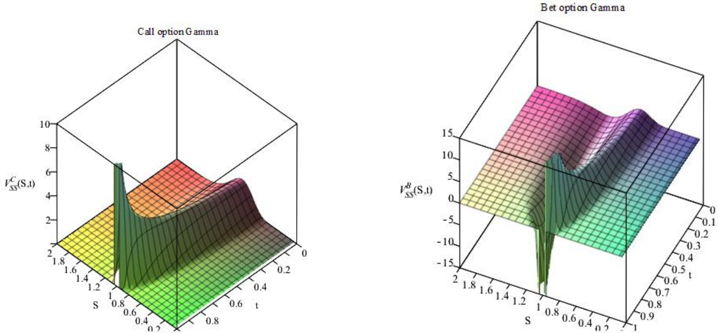


Figure: Exact Gamma $\frac{\partial^2 V}{\partial S^2}$ for call (left) and bet (right) option, with $T = 1$, $K = 1$, $\sigma = 0.2$, $r = 0.04$ and $B = 0.3$.

3D visualization of error - FE, Call and Bet

Invisible errors on solution plots may be visualized on error plots:

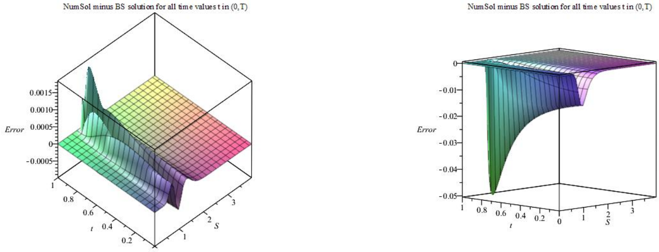
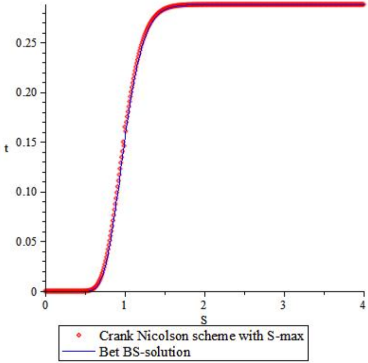


Figure: Typical example of error function for call (left) and bet (right) option, with $T = 1$, $K = 1$, $\sigma = 0.2$, $r = 0.04$ and $B = 0.3$ (StdCase), when solved with a standard explicit Euler method (FE=BtCS).

Bet error $\simeq$ 30 times Call error.
Call error resembles call Gamma in structure.
Bet error resembles bet Delta in structure.

2D ($t = 0$) visualization of solution - CN, Bet

Invisible 3D-errors may be visible in 2D with coarse step sizes ($\Delta S = h \simeq 0.01, \Delta t = k \simeq 0.05$) when "zooming in" on $S = K$:



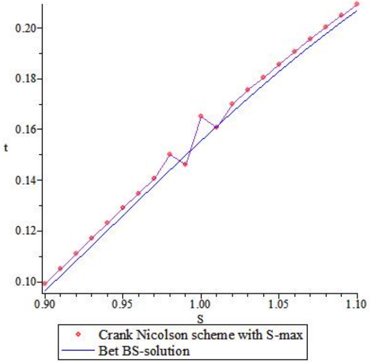


Figure: Typical example of solution at $t = 0$ for all S (left) and in a small S-interval around K (right) for bet option in StdCase, when solved with a standard Crank Nicolson method (CN=CtCS).

FE-convergence of maximal error - Call and Bet

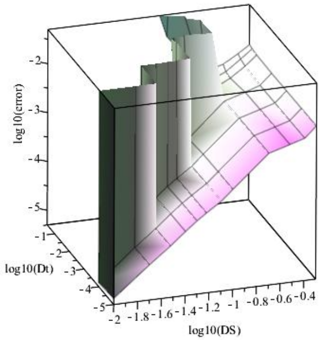


Figure: Maximal error for StdCase at time $t = 0$ for call (left) and bet (right) option, when solved with a standard explicit Euler method (FE).

FE is conditionally convergent with order: $e = \mathcal{O}(DS^2 + Dt)$.
Observed order: $e_{\text{call}} = \mathcal{O}(DS^2)$, $e_{\text{bet}} = \mathcal{O}(DS^1)$.

CN-convergence of maximal error - Call and Bet

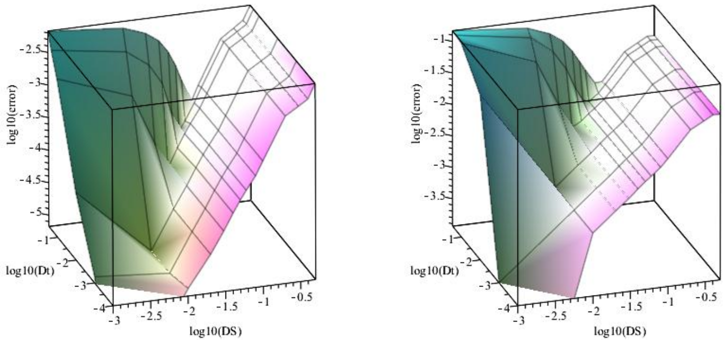


Figure: Maximal error for StdCase at time $t = 0$ for call (left) and bet (right) option, when solved with a standard implicit Crank Nicolson method (CN).

CN is unconditionally convergent with order: $e = \mathcal{O}(DS^2 + Dt^2)$.
Observed order in buble: $e_{\text{call}} = \mathcal{O}(DS^2 + Dt^2)$, $e_{\text{bet}} = \mathcal{O}(DS^1 + Dt^?)$.
Observed order outside buble: $e_{\text{call}} = \mathcal{O}(DS^2)$, $e_{\text{bet}} = \mathcal{O}(DS^1)$.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_0-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps.
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_s-optimization - Optimal location of $S = K$ with respect to element boundaries
- Rannacher time stepping - Reduced time step size for the first few steps
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_α-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_α-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps.
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_α-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps.
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler $(\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t))$ and Crank-Nicolson $(\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2))$ deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_α-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps.
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

Numerical issues

- Micro trading means a need for very precise and very fast numerical solutions.
- Standard finite difference methods may deliver the required precision but maybe not at an acceptable cost.
- Explicit and implicit Euler ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t)$) and Crank-Nicolson ($\mathcal{O}(\Delta S^2) + \mathcal{O}(\Delta t^2)$) deliver only $\mathcal{O}(\Delta S^2)$ for put and call and $\mathcal{O}(\Delta S)$ for bet, and very slow if any convergence in Δt within the computational capacity.
- Hence "shortcuts" are needed i.e. more advanced methods.

Examples of shortcuts:

- K_α-optimization - Optimal location of $S = K$ with respect to element boundaries.
- Rannacher time stepping - Reduced time step size for the first few steps.
- Mesh grading - Using smaller step sizes ΔS close to $S = K$ where the error is the biggest.

K_α-optimization

The error for given stepsizes depends significantly on the location of K in the element that it belongs to. Say

$$K = (s_h + \alpha_h)h \text{ for } s_h \in \mathcal{N} \text{ and } 0 \leq \alpha_h < 1.$$

We say: K is in α_h-position in element number s_h.
Given K, s_h and α_h are uniquely (but in a complex way) determined by h.
To control the error, we must first control and then optimize α_h:
Force $\alpha_h \to \alpha$ (α user provided). Now $\tilde{s}_h = \frac{K-\alpha h}{h}$ is no longer integer.
Force $s_h \to s = \lceil \frac{K-\alpha h}{h} \rceil$ which is integer and

$$s = \lceil \frac{K-\alpha h}{h} \rceil = \lceil \frac{K-\alpha_h h}{h} + \frac{(\alpha_h - \alpha)h}{h} \rceil = s_h + \lceil \alpha_h - \alpha \rceil.$$

But $-1 < \alpha_h - \alpha < 1 \Rightarrow 0 \leq \lceil \alpha_h - \alpha \rceil \leq 1$ so $s_h \leq s \leq s_h + 1$.
Hence K lies in the same or one later element, i.e. the same or slightly smaller step size $\tilde{h}$ is induced:

$$K = (s+\alpha)\tilde{h} \text{ i.e. } \tilde{h} = \frac{K}{s+\alpha} = \frac{K}{\lceil \frac{K-\alpha h}{h} \rceil + \alpha}.$$

Hence we compute with a slightly smaller step size than requested.

Error as function of α - CN, Call

We compute with a fine mesh with step sizes $h \simeq 0.03$, $k \simeq 0.001$ and α in $[0, 1)$ with $\Delta\alpha = 0.025$:

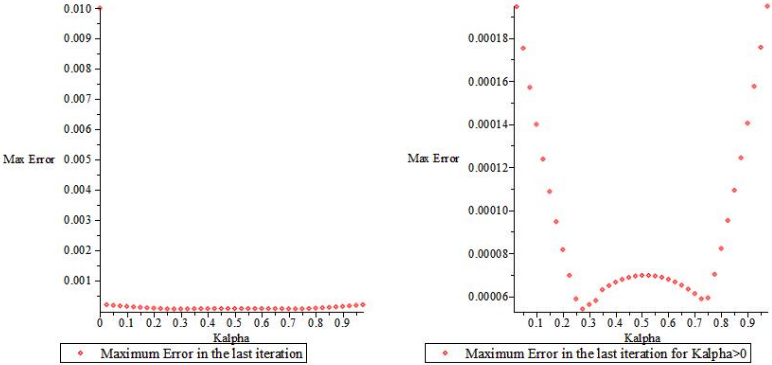


Figure: Maximal error with CN for call in StdCase at time $t = 0$ as function of $\alpha \in [0, 1[$. Left with, right without $\alpha = 0$.

$K_\alpha = 0.275$ is the optimal α.

Is K_α stable for different interest rates? - CN, Call

Now we consider stability of K_α (the optimal α) for changing interest rates (r) for the fine mesh.

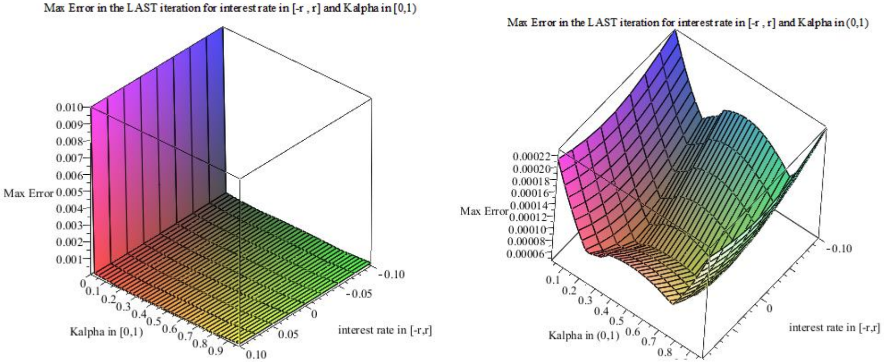


Figure: Maximal error with CN for call in StdCase at time $t = 0$ as function of $r \in [-0.1, 0.1]$ and $\alpha \in [0, 1[$. Left with, right without $\alpha = 0$.

Full stability with r: $K_\alpha = 0.275$ for $r > 0$. $K_\alpha = 0.725$ for $r < 0$.

K_α and its stability wrt r - CN, Bet

We compute with the fine mesh and α in $[0, 1)$ with $\Delta\alpha = 0.025$ and $r \in [-0.1, 0.1]$:

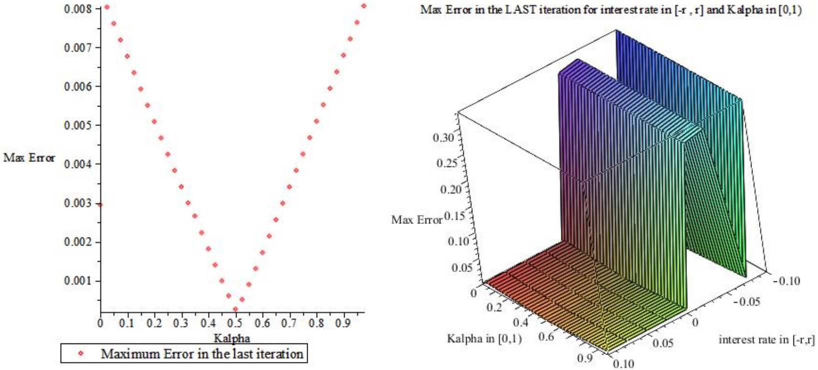


Figure: Maximal error with CN for bet in StdCase at time $t = 0$ as function of $\alpha \in [0, 1[$ (left) and $r \in [-0.1, 0.1]$ and $\alpha \in [0, 1[$ (right).

Full stability with r: $K_\alpha = 0.500$ is the optimal α for $r \geq 0$.

Is K_α stable for different volatilities? - CN, Call

Now we consider stability of K_α with respect to volatility (σ) between 0.1 and 0.4 in the standard case for the fine mesh.

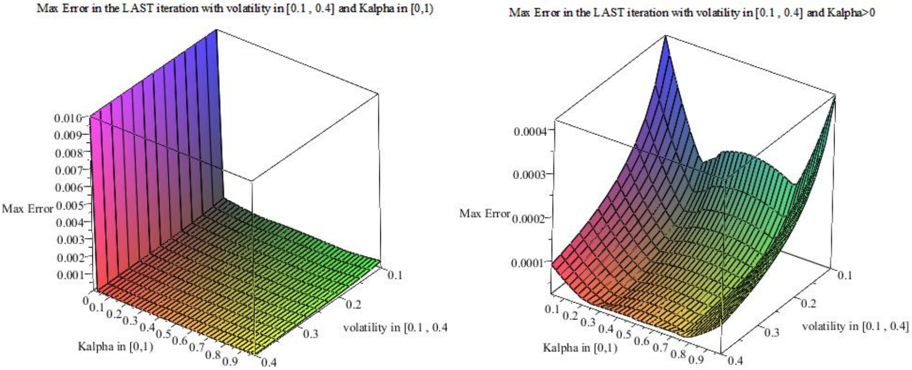


Figure: Maximal error with CN for call in StdCase at time $t = 0$ as function of $\sigma \in [0.1, 0.4]$ and $\alpha \in [0, 1[$. Left with, right without $\alpha = 0$.

Full stability with σ: $K_\alpha = 0.275$ is the optimal α.

Is K_α stable for different volatilities? - CN, Bet

We consider stability of K_α with respect to volatility (σ) between 0.1 and 0.4 in the standard case for the fine mesh.

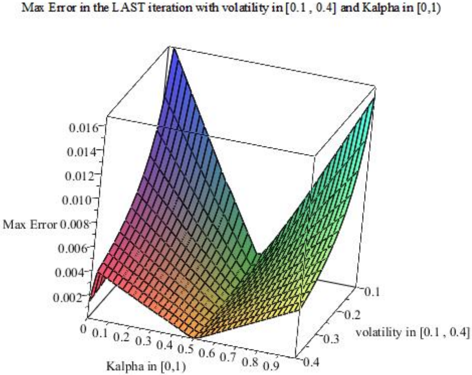


Figure: Maximal error with CN for bet in StdCase at time $t = 0$ as function of $\sigma \in [0.1, 0.4]$ and $\alpha \in [0, 1[$.

Full stability with σ: $K_\alpha = 0.500$ is the optimal α.

Conclusion on K_α-optimization

- By optimizing the position of the strike price relative to the element end points the error may be reduced substancially (more than 300 times in the worst cases).
- The optimal position depends on the option type but not on the various parameters and is
  - $K_\alpha = 0.275$ for call and put options with $r > 0$.
    $\alpha \in [0.2, 0.8]$ give errors less than the double of the minimal.
  - $K_\alpha = 0.500$ for bet options with $r > 0$.
    $\alpha \in [0.4, 0.6]$ give errors less than the double of the minimal.
- The price of adjusting the S-stepsize to fit the optimal α is negligible $\mathcal{O}(1)$ and the adjustment is done a priori to the solution, and hence can be built into any existing code.

Having the strike price midway between nodal points was considered by Tavella et al (2999) and Pooley et al (2003) in [2, 4]. Finding the optimal α is novel.

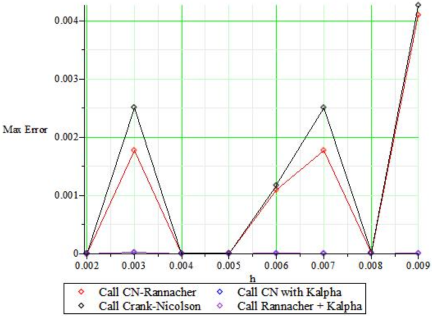
Rannacher time stepping

Rannacher (1984) considered in [3] a start up process for the Crank Nicolson method with non smooth initial value condition:
The first few timesteps in CN is replaced by a number of smaller implicit Euler (BE=FtCS) steps to take advantage of the L-stability of BE (no oscillations).
Giles et al (2006) showed in [1] that replacing the first CN timestep by 4 BE quarter-steps works better than replacing the first two CN timesteps by 4 BE half-steps. Hence we consider the 4 quarter-step version.
We compare 4 methods

- CN
- CN with Rannacher time stepping
- CN with K_α-optimization
- CN with K_α-optimization and Rannacher time stepping

All with parameters $T = 2$, $K = 1$, $B = 0.3$, $r = 0.05$, $\gamma = 0$, $\sigma = 0.2$ and $S_{max} = 5$. We consider stepsizes $\Delta S = h \in [0.002, 0.1]$ and $\Delta t = k = 5h$.

Comparing the Methods - CN, Call



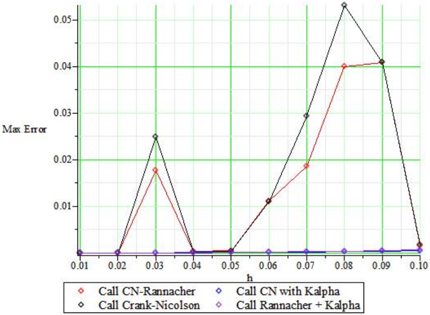


Figure: Maximal error with 4 versions of CN for call at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.009]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

Clearly CN with K_α-opt. and CN with K_α-opt. and Rannacher time stepping are the most interesting, and are considered alone next:

Comparing the two best methods - CN, Call

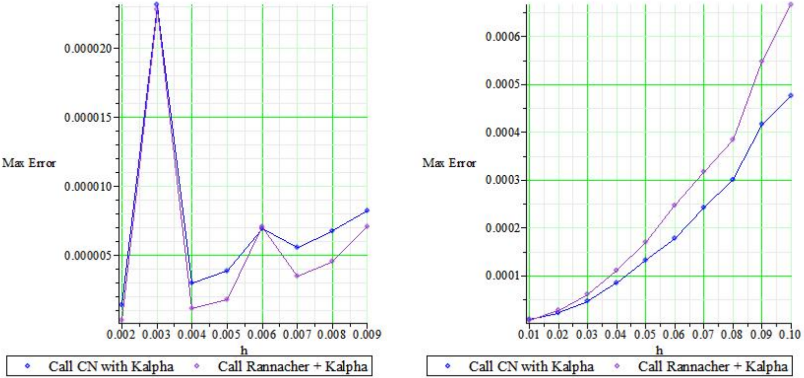
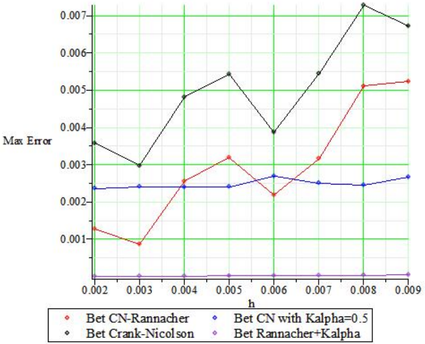


Figure: Maximal error with 2 versions of CN for call at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.009]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

CN with K_α-optimization is best for coarse meshes.
CN with K_α-opt. and Rannacher time stepping is best for fine meshes.

Comparing the Methods - CN, Bet



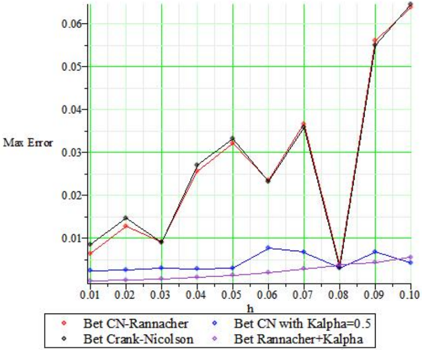


Figure: Maximal error with 4 versions of CN for bet at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.009]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

CN with K_α-optimization and Rannacher time stepping is best for both coarse and fine meshes but most for fine meshes.

Conclusions on K_{alpha} and Rannacher time stepping

- "Vanilla" CN is the worst of the 4 methods considered for all stepsizes.
- CN with Rannacher time stepping improves (but only slightly) over vanilla CN.
- CN with K_α-optimization is better than the previous two, except for fine meshes for the call option where CN with Rannacher time stepping is better.
- CN with K_α-optimization and Rannacher time stepping is better than the previous three, except for coarse meshes for the call option where CN with K_α-optimization is better.

CN with K_α-optimization and Rannacher time stepping (CNRK) is the overall winner.

Conclusions on K_{alpha} and Rannacher time stepping

- "Vanilla" CN is the worst of the 4 methods considered for all stepsizes.
- CN with Rannacher time stepping improves (but only slightly) over vanilla CN.
- CN with K_α-optimization is better than the previous two, except for fine meshes for the call option where CN with Rannacher time stepping is better.
- CN with K_α-optimization and Rannacher time stepping is better than the previous three, except for coarse meshes for the call option where CN with K_α-optimization is better.

CN with K_α-optimization and Rannacher time stepping (CNRK) is the overall winner.

Conclusions on K_{alpha} and Rannacher time stepping

- "Vanilla" CN is the worst of the 4 methods considered for all stepsizes.
- CN with Rannacher time stepping improves (but only slightly) over vanilla CN.
- CN with K_α-optimization is better than the previous two, except for fine meshes for the call option where CN with Rannacher time stepping is better.
- CN with K_α-optimization and Rannacher time stepping is better than the previous three, except for coarse meshes for the call option where CN with K_α-optimization is better.

CN with K_α-optimization and Rannacher time stepping (CNRK) is the overall winner.

Conclusions on K_{alpha} and Rannacher time stepping

- "Vanilla" CN is the worst of the 4 methods considered for all stepsizes.
- CN with Rannacher time stepping improves (but only slightly) over vanilla CN.
- CN with K_α-optimization is better than the previous two, except for fine meshes for the call option where CN with Rannacher time stepping is better.
- CN with K_α-optimization and Rannacher time stepping is better than the previous three, except for coarse meshes for the call option where CN with K_α-optimization is better.

CN with K_α-optimization and Rannacher time stepping (CNRK) is the overall winner.

Conclusions on K_{alpha} and Rannacher time stepping

- "Vanilla" CN is the worst of the 4 methods considered for all stepsizes.
- CN with Rannacher time stepping improves (but only slightly) over vanilla CN.
- CN with K_α-optimization is better than the previous two, except for fine meshes for the call option where CN with Rannacher time stepping is better.
- CN with K_α-optimization and Rannacher time stepping is better than the previous three, except for coarse meshes for the call option where CN with K_α-optimization is better.

CN with K_α-optimization and Rannacher time stepping (CNRK) is the overall winner.

Mesh grading

Most of the error is located close to $S = K$. Tangman et al suggests in [6] the following grading function:

$$S(x) = K + \frac{1}{b}\sinh(c_1(1-x) + c_2 x) \text{ with } \begin{cases} c_1 = \operatorname{arcsinh}(-bK) \\ c_2 = \operatorname{arcsinh}(b(S_{max} - K)) \end{cases}.$$

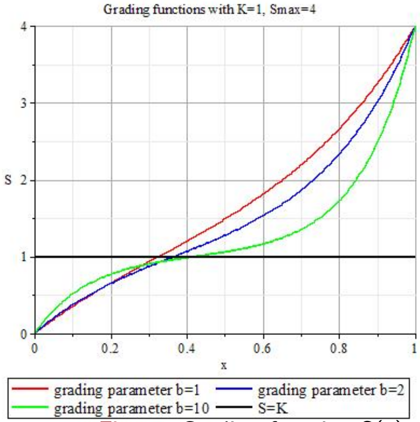


Figure: Grading function S(x)

Comparing the methods - CNRK with grading, Call

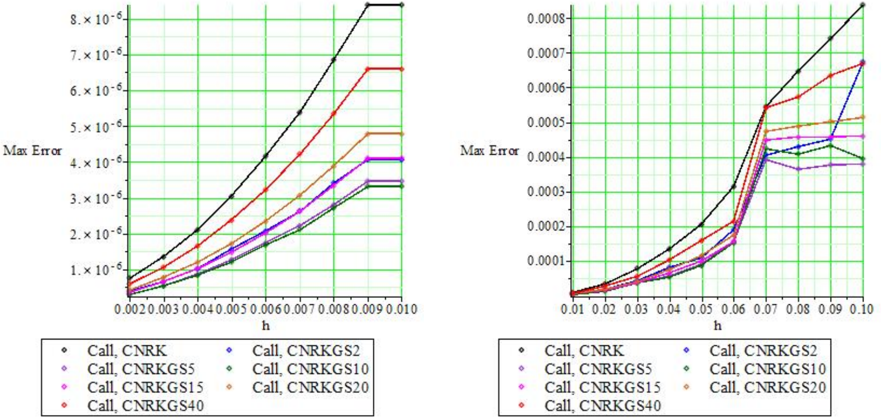


Figure: Maximal error with 6 versions ($b = 0$, 2, 5, 10, 15, 20, 40) of CN with mesh grading for call at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.01]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

Conclusions on V for CNRK with grading - Call

- CNRK with mesh grading with grading parameter $b \simeq 10$ is significantly better than CNRK and CNRK with mesh grading with other grading parameter values.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter $b \simeq 10$.

Now consider how well we recover the greeks:

Conclusions on V for CNRK with grading - Call

- CNRK with mesh grading with grading parameter $b \simeq 10$ is significantly better than CNRK and CNRK with mesh grading with other grading parameter values.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter $b \simeq 10$.

Now consider how well we recover the greeks:

Conclusions on V for CNRK with grading - Call

- CNRK with mesh grading with grading parameter $b \simeq 10$ is significantly better than CNRK and CNRK with mesh grading with other grading parameter values.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter $b \simeq 10$.

Now consider how well we recover the greeks:

Recovering Delta$= \frac{\partial V}{\partial S}$ - CNRK with grading, Call

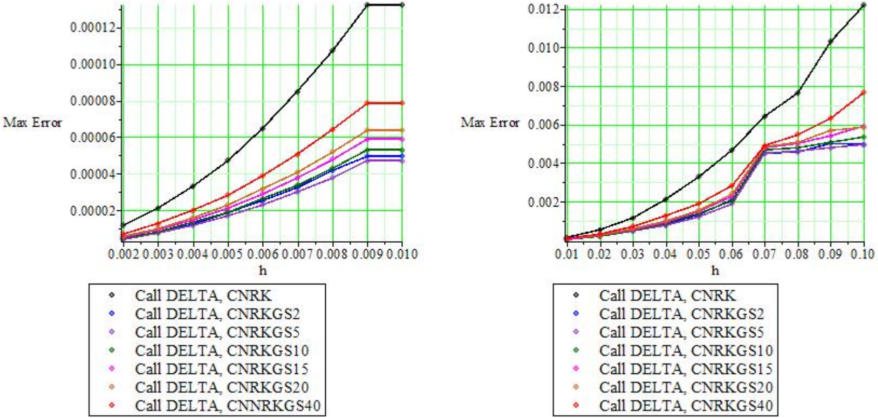


Figure: Maximal error in Delta with 6 versions ($b = 0, 2, 5, 10, 15, 20, 40$) of CN with mesh grading for call at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.01]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

Recovering Gamma= $\frac{\partial^2 V}{\partial S^2}$ - CNRK with grading, Call

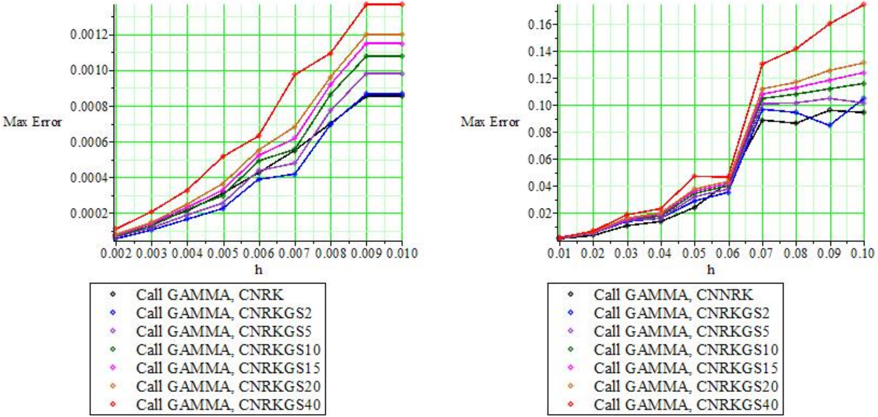


Figure: Maximal error in Delta with 6 versions ($b = 0, 2, 5, 10, 15, 20, 40$) of CN with mesh grading for call at time $t = 0$ as function of h. Fine meshes ($h \in [0.002, 0.01]$) to the left and coarse meshes ($h \in [0.01, 0.1]$) to the right.

Conclusions on V, $\Delta(V)$ and $\Gamma(V)$ for CNRK with grading - Call

- CNRK with mesh grading with the optimal grading parameter b is significantly better than CNRK with mesh grading with grading parameter values far from the optimal value (including CNRK corresponding to $b = 0$).
- The optimal mesh grading parameter is
  $b \simeq 10$ for recovering the solution V.
  $b \simeq 5$ for recovering the Delta $\frac{\partial V}{\partial S}$.
  $b \simeq 2$ for recovering the Gamma $\frac{\partial^2 V}{\partial S^2}$.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter depending on the what is recovered.

Conclusions on V, $\Delta(V)$ and $\Gamma(V)$ for CNRK with grading - Call

- CNRK with mesh grading with the optimal grading parameter b is significantly better than CNRK with mesh grading with grading parameter values far from the optimal value (including CNRK corresponding to $b = 0$).
- The optimal mesh grading parameter is
  $b \simeq 10$ for recovering the solution V.
  $b \simeq 5$ for recovering the Delta $\frac{\partial V}{\partial S}$.
  $b \simeq 2$ for recovering the Gamma $\frac{\partial^2 V}{\partial S^2}$.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter depending on the what is recovered.

Conclusions on V, $\Delta(V)$ and $\Gamma(V)$ for CNRK with grading - Call

- CNRK with mesh grading with the optimal grading parameter b is significantly better than CNRK with mesh grading with grading parameter values far from the optimal value (including CNRK corresponding to $b = 0$).
- The optimal mesh grading parameter is
  $b \simeq 10$ for recovering the solution V.
  $b \simeq 5$ for recovering the Delta $\frac{\partial V}{\partial S}$.
  $b \simeq 2$ for recovering the Gamma $\frac{\partial^2 V}{\partial S^2}$.

So the overall winner as the best Crank-Nicolson method is Crank-Nicolson with K_α-optimization, Rannacher time stepping and mesh grading with a grading parameter depending on the what is recovered.

Future works

- Analytical proof for K_α-optimization.
- Compare the methods with respect to their orders of convergence.
- Compare CN with K_α-optimization, Rannacher time stepping and grading on the Greeks for the bet option.

Future works

- Analytical proof for K_α-optimization.
- Compare the methods with respect to their orders of convergence.
- Compare CN with K_α-optimization, Rannacher time stepping and grading on the Greeks for the bet option.

Future works

- Analytical proof for K_α-optimization.
- Compare the methods with respect to their orders of convergence.
- Compare CN with K_α-optimization, Rannacher time stepping and grading on the Greeks for the bet option.

References:

- M. B. Giles, R. Carter, Convergence analysis of Crank-Nicolson and Rannacher time-marching, (2006).
- D. M. Pooley, K. R. Vetzal, and P. A. Forsyth, Convergence remedies for non-smooth payoffs in option pricing, *Journal of Computational Finance*, **6.4**, 25–40, (2003).
- R. Rannacher, Finite element solution of diffusion problems with irregular data, *Numerische Mathematik 43*, 309–327, (1984).
- D. Tavella, C. Randall, Pricing Financial Instruments: The Finite Difference Method, *Wiley series in financial engineering*, (2000).
- P. Wilmott, S. Howison and J. Dewynne, The Mathematics of Financial Derivatives, *Cambridge University Press*, (1995).
- D. Y. Tangman, A. Gopaul and M. Bhuruth, Numerical pricing op options using high-order compact finite difference schemes, *Journal of Computational and Applied Mathematics 218*, 270–280, (2008).