Affine Recursion Problem and a general framework for Adjoint Methods for calculating sensitivities for financial instruments

Jörg Kienitz[1] Nikolai Nowaczyk[2]

[1]Deutsche Postbank AG
Quantitative Analysis Group

[2]Uni Regensburg
Fakultät für Mathematik

MathFinance Conference 26 - 27 March 2012

Outline

Outline

What is the "Adjoint Method"?

- originally a technique to calculate certain quantities in fluid dynamics
- 2006: application to finance introduced by Giles and Glasserman $\rightarrow$ *Risk Quant of the Year 2007*
- efficient evaluation of pathwise sensitivities in Monte Carlo simulation
- 2009: Leclerc, Liang, Schneider applied the "adjoint method" to calculate the Delta and the Vega of Bermuda-style options

Finding a General Framework

- What is the ”Adjoint Method”?

Finding a General Framework

- What is the ”Adjoint Method”?
  - More a calculational trick than a sharp abstract notion.
  - Appears in lots of different forms and applications.

Finding a General Framework

- What is the ”Adjoint Method”?
  - More a calculational trick than a sharp abstract notion.
  - Appears in lots of different forms and applications.
  - Can one grasp the Adjoint Method formally?

Finding a General Framework

- What is the "Adjoint Method"?
  - More a calculational trick than a sharp abstract notion.
  - Appears in lots of different forms and applications.
  - Can one grasp the Adjoint Method formally?
- Find a problem P such that
  - all common problems Q, which can be solved by the "adjoint method", are instances of P
  - P can be formally stated as a mathematical problem.
  - Solving P can be formalized by a mathematical theorem.
  - There exists a mathematical theorem, which formalizes the notion of solving P by the "adjoint method".
- Develop an algorithm to practically solve P.

Outline

Affine Recursion Problem Statement

Definition

Assume we are given the data $N, m, q \in \mathbb{N}$ and for any $n = 1, \ldots, N-1$

$$A_1 \in \mathbb{R}^{m\times q} \quad D(n) \in \mathbb{R}^{m\times m} \quad C(n) \in \mathbb{R}^{m\times q} \quad v \in \mathbb{R}^{1\times m}$$

Assume that $A(n) \in \mathbb{R}^{m\times q}$ is a sequence of matrices satisfying the *forward recursion*

$$\forall 1 \leq n \leq N-1 : A(n+1) = D(n)A(n) + C(n), \quad A(1) = A_1.$$

The calculation of

$$w := vA(N) \in \mathbb{R}^{1\times q}$$

is called the *Affine Recursion Problem* or just an *ARP*.

Solution using Forward Method

Theorem (forward method)

Any ARP is uniquely solvable.

Solution using Forward Method

Theorem (forward method)

Any ARP is uniquely solvable. In fact

$$w = vA(N) = \sum_{j=1}^{N-1} v\Big(\prod_{k=1}^{j-1} D(N-k)\Big) C(N-j) + v\left(\prod_{j=1}^{N-1} D(N-j)\right) A_1 .$$

Implementation of the Forward Method

Implementation

```
function [w A] = forward_method(A1, D, C, v)
    [m, q, N] = size(C);          %calculate dimensions
    N = N+1;
    A = zeros(m,q,N);             %initialize A
    A(:,:,1)=A1;
    for n = 1:N-1                 %run forward recursion
        A(:,:,n+1) = D(:,:,n)*A(:,:,n) + C(:,:,n);
    end
    w = v * A(:,:,N);             %calculate result
end
```

Implementation of the Forward Method

Implementation

```
function [w A] = forward_method(A1, D, C, v)
    [m, q, N] = size(C);        %calculate dimensions
    N = N+1;
    A = zeros(m,q,N);           %initialize A
    A(:,:,1)=A1;
    for n = 1:N-1               %run forward recursion
        A(:,:,n+1) = D(:,:,n)*A(:,:,n) + C(:,:,n);
    end
    w = v * A(:,:,N);           %calculate result
end
```

Computational cost: $\mathcal{O}(Nqm^2)$ resp. $\mathcal{O}(m^4)$

Solution using Adjoint Method

Theorem (Adjoint method)

There exists an *adjoint sequence* of vectors $V(n) \in \mathbb{R}^{m\times 1}$ and a *total adjoint sequence* $\overline{V}(n) \in \mathbb{R}^{q\times 1}$, which calculate the result vector of the ARP.

Solution using Adjoint Method

Theorem (Adjoint method)

There exists an *adjoint sequence* of vectors $V(n) \in \mathbb{R}^{m\times 1}$ and a *total adjoint sequence* $\overline{V}(n) \in \mathbb{R}^{q\times 1}$, which calculate the result vector of the ARP. In fact

$$w = vA(N) = \overline{V}(1)^t + V(1)^t A_1$$

These sequences satisfy the *backward recursions*

$$\forall 1 \leq n \leq N-1 : V(n) = D(n)^t V(n+1),$$
$$V(N) := v^t$$
$$\forall 1 \leq n \leq N-1 : \overline{V}(n) = C(n)^t V(n+1) + \overline{V}(n+1),$$
$$\overline{V}(N) := 0$$

Implementation of the Adjoint Method

Implementation

```
function w = adjoint_method(A1, D, C, v)
    [m, q, N] = size(C);      %calculate dimensions
    N = N+1;
    V=v.';                    %initialize V
    Vbar = zeros(q,1);        %initialize Vbar
    for n = N-1:-1:1          %run backward recursion
        Vbar = Vbar + C(:,:,n).' * V;
        V = D(:,:,n).' * V;
    end
    w = Vbar.'+ V.' * A1;         %calculate result
end
```

Implementation of the Adjoint Method

Implementation

```
function w = adjoint_method(A1, D, C, v)
    [m, q, N] = size(C);      %calculate dimensions
    N = N+1;
    V=v.';                    %initialize V
    Vbar = zeros(q,1);        %initialize Vbar
    for n = N-1:-1:1          %run backward recursion
        Vbar = Vbar + C(:,:,n).' * V;
        V = D(:,:,n).' * V;
    end
    w = Vbar.'+ V.' * A1;         %calculate result
end
```

Computational cost: $\mathcal{O}(Nqm)$ resp. $\mathcal{O}(m^3)$

Conclusion

- ARP is formulated in an abstract Linear Algebra setting.
- ARP can be solved by the Forward Method or the Adjoint Method.
- Both methods yield the same exact result, a closed form representation of the solution and a recursive sequence to calculate it.
- Forward method requires a matrix-matrix recursion and is therefore of complexity $\mathcal{O}(m^4)$, Adjoint method works with matrix-vector recursions and is of complexity $\mathcal{O}(m^3)$.

Outline

Setup in Stochastical Analysis

- Underlying (e.g. Libor) is modelled as a diffusion

$$d\tilde{X}_t = a(\tilde{X}_t, t, \sigma)dt + b(\tilde{X}_t, t, \sigma)dW(t) \in \mathbb{R}^m.$$

- Choose some *maturity* $T > 0$, a *payoff function* $g \in \mathcal{C}^2$ and define
  1. the expected *payoff* $\tilde{p} := \mathrm{E}[g(\tilde{X}_T)] \in \mathbb{R}_{\geq 0}$,
  2. the *Delta* $\tilde{\Delta} := \nabla_{\tilde{X}_0}(\tilde{p}) \in \mathbb{R}^{1\times m}$,
  3. the *Gamma* $\tilde{\Gamma} := \mathrm{Hess}_{\tilde{X}_0}(\tilde{p}) \in \mathbb{R}^{m\times m}$,
  4. the *Vega* $\tilde{\mathcal{V}} := \nabla_{\sigma}(\tilde{p}) \in \mathbb{R}^{1\times q}$.

Numerical Approximation

Choose

1. any *time grid* $0 = T_1 < \ldots < T_N = T$
2. an *approximation sequence*

$$X(n+1) := F_n(X(n), \sigma), \qquad X(1) := \tilde{X}_{T_1} = \tilde{X}_0,$$

   with an *evolution map* F_n, usually (Log-)Euler scheme.
3. obtain approximation $\tilde{X}_{T_n} \approx X(n)$.
4. Approximate $\tilde{p}, \tilde{\Delta}, \tilde{\Gamma}, \tilde{\mathcal{V}}$ by

$$p := g(X(N)), \qquad \Delta := \nabla_{X(1)}(p) \in \mathbb{R}^{1\times m},$$
$$\Gamma := \mathrm{Hess}_{X(1)}(p) \in \mathbb{R}^{m\times m}, \qquad \mathcal{V} := \nabla_{\sigma}(p) \in \mathbb{R}^{1\times q}.$$

5. Calculate using Monte Carlo Simulation.

Reduction to ARP

General Strategy

1. Differentiate the *evolution equation* $X(n+1) = F_n(X(n), \sigma)$ with respect to $X(1)$ or σ.
2. Obtain recursive formulae for Δ, Γ, $\mathcal{V}$.
3. Formulate calculation of Greeks by ARPs.
4. Solve ARPs with general theory.

Important Applications

- Underlying SDE given by the Libor Market Model.
- Payoff function is defined by a Bermudan Swaption.
- Payoff function is defined by a Trigger Swap.

In all cases ARP solution coincides with existing theory.

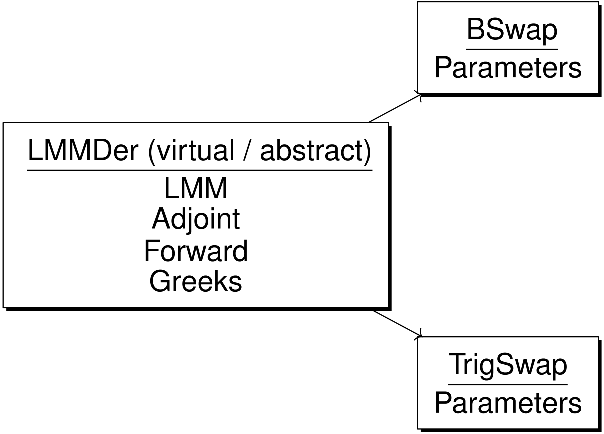
Bermudan Swaption / Trigger Swap

- Tenor structure

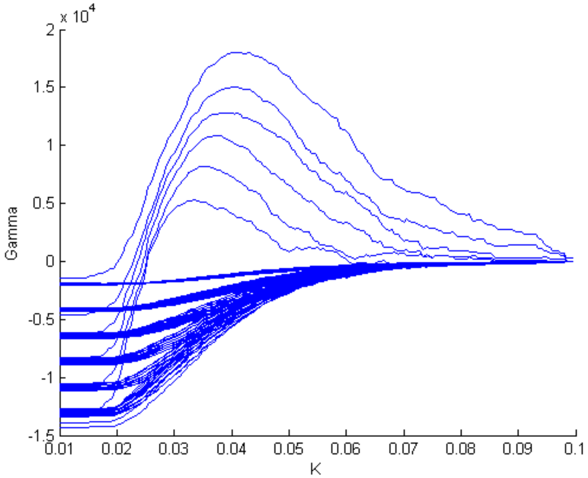
$$0 = T_1 < \ldots < T_e < \ldots < T_{m+1}, \quad \tau_i := T_{i+1} - T_i \triangleq 1/2 \text{ year}$$

- Bermudan Swaption: Grants the holder the right (but not the obligation) to enter into a swap at T_r, $r = e, \ldots, m$
- Trigger Swap: Is triggered the first time $e \leq \tau \leq m$, where $L(\tau) > K$. Holder receives some spread rate s on $[T_e, T_\tau[$ and enters into a swap on $[T_\tau, T_{m+1}]$ with predetermined fixed rate κ.
- Discounted payoffs fail to be $\mathcal{C}^2$ ($\leadsto$ smooth approximation).

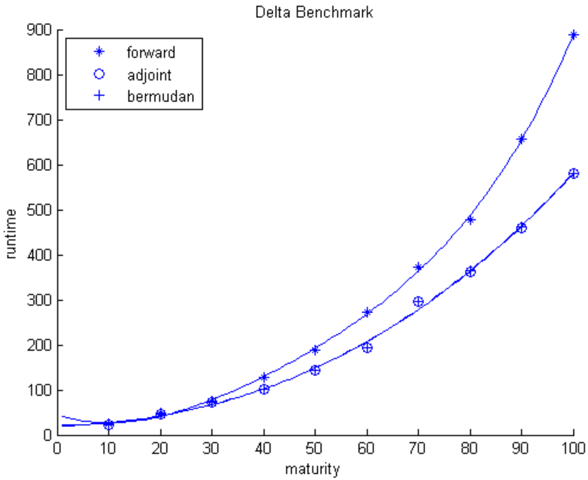
Object Oriented Programming (OOP)



Bermudan Swaption: Strike Rate K vs. Gamma



Benchmark for Δ



forward method: $\mathcal{O}(m^4)$, adjoint method $\mathcal{O}(m^3)$

Outline

Conclusion

1. Principle of ”Ajoint Method” can be formalized and solved as a problem formulated in the setting of abstract Linear Algebra.
2. Existing theory can be obtained as a special case.
3. This method of abstraction suggests a way towards a convenient object oriented implementation.
4. OOP Implementation is particularly useful when analysing a large number of derivatives on the same underlying.

Further information

More details can be found

- in the preprint of the paper on SSRN
- in the upcoming book ”Financial Modelling: Theory, Implementation and Practice” (Wiley Finance Series)

Further information

More details can be found

- in the preprint of the paper on SSRN
- in the upcoming book ”Financial Modelling: Theory, Implementation and Practice” (Wiley Finance Series)

Thank’s for your attention.