GPU pricing of cross-currency interest rate derivatives under a FX volatility skew model

Duy Minh Dang
Department of Computer Science
University of Toronto, Toronto, Canada
dmdang@cs.toronto.edu

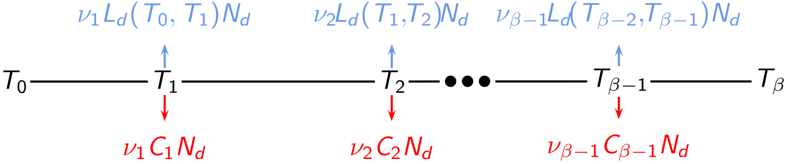
Joint work with Christina Christara and Ken Jackson

6th World Congress of the Bachelier Finance Society
Toronto, Canada, June 22 – 26, 2010

Outline

PRDC swaps

- **Long-dated** swaps ($\geq$ 30 years);
- Two currencies (domestic and foreign) and the foreign exchange (FX) rate
- Funding leg: domestic LIBOR payments (from the *investor*)
- Structured leg: FX-linked PRDC coupons (from the *issuer*)



- $C_\alpha = \min\left(\max\left(c_f \dfrac{s(T_\alpha)}{F(0, T_\alpha)} - c_d, b_f\right), b_c\right)$
  - $s(T_\alpha)$: the spot FX-rate at time T_α;
  - $F(0, T_\alpha) = \dfrac{P_f(0, T_\alpha)}{P_d(0, T_\alpha)} s(0)$
  - c_d, c_f: domestic and foreign coupon rates; b_f, b_c : a cap and a floor
- In the standard case ($b_f = 0$ and $b_c = \infty$), C_α is a **call option on the spot FX rate**

$$C_\alpha = h_\alpha \max(s(T_\alpha) - k_\alpha, 0), \quad h_\alpha = \frac{c_f}{f_\alpha}, k_\alpha = \frac{f_\alpha c_d}{c_f}$$

PRDC swaps (cont.)

- A PRDC swap are portfolio of long dated FX options
  - stochastic interest rates
  - effects of FX volatility skew (log-normal vs. local vol/stochastic vol.)

  $\Rightarrow$ multi-factor models (≥ 3), calibration difficulties
- Moreover, the swap usually contains some optionality:
  - knockout
  - FX Target Redemption (FX-TARN)
  - Bermudan cancelable
- Popular pricing approaches:
  - Monte-Carlo
  - PDEs

This talk is about

- Pricing of PRDC swaps (FX-IR exotics) on GPUs via a PDE approach
- Three-factor model with a FX local volatility function (V. Piterbarg, 2006)
- Bermudan cancelable feature
- Impact of FX volatility skew

Bermudan cancelable PRDC swaps

The issuer has the right to cancel the swap at **any** of the times $\{T_\alpha\}_{\alpha=1}^{\beta-1}$

- **Observations**: terminating a swap at T_α is the same as
  i. continuing the underlying swap, and
  ii. entering into the offsetting swap at $T_\alpha \Rightarrow$ the issuer has a long position in an associated offsetting Bermudan swaption
- **Pricing**:
  - Over each period: dividing the pricing of a cancelable PRDC swap into
    i. the pricing of the underlying PRDC swap (a "vanilla" PRDC swap), and
    ii. the pricing of the associated offsetting Bermudan swaption
  - Computations: over each period, 2 model-dependent PDEs to solve on separate GPUs, one for the PRDC coupons, one for the "option" in the swaption
  - Across each date: apply jump conditions and exchange information

Outline

The pricing model

Consider the following model under domestic risk neutral measure (V. Piterbarg, 2006)

$$
\begin{aligned}
\frac{ds(t)}{s(t)} &= (r_d(t) - r_f(t))dt + \gamma(t, s(t))dW_s(t), \\
dr_d(t) &= (\theta_d(t) - \kappa_d(t) r_d(t))dt + \sigma_d(t) dW_d(t), \\
dr_f(t) &= (\theta_f(t) - \kappa_f(t) r_f(t) - \rho_{fs}(t)\sigma_f(t)\gamma(t, s(t)))dt + \sigma_f(t) dW_f(t)
\end{aligned}
$$

- $r_i(t), i = d, f$: domestic and foreign interest rates with mean reversion rate and volatility functions $\kappa_i(t)$ and $\sigma_i(t)$
- $s(t)$: the spot FX rate (units domestic currency per one unit foreign currency)
- $W_d(t), W_f(t)$, and $W_s(t)$ are correlated Brownian motions with $dW_d(t)dW_s(t) = \rho_{ds}dt$, $dW_f(t)dW_s(t) = \rho_{fs}dt$, $dW_d(t)dW_f(t) = \rho_{df}dt$
- Local volatility function $\gamma(t, s(t)) = \xi(t)\left(\frac{s(t)}{L(t)}\right)^{\varsigma(t)-1}$
  - $\xi(t)$: relative volatility function
  - $\varsigma(t)$: constant elasticity of variance (CEV) parameter
  - $L(t)$: scaling constant (e.g. the forward FX rate $F(0, t)$)

The 3-D pricing PDE

Over each period of the tenor structure, we need to solve two PDEs of the form

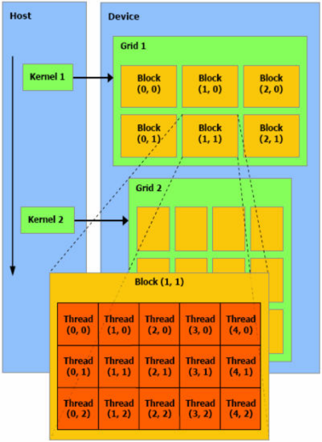
$$
\begin{aligned}
\frac{\partial u}{\partial t} + \mathcal{L}u \equiv & \frac{\partial u}{\partial t} + (r_d - r_f)s\frac{\partial u}{\partial s} \\
& + \Big(\theta_d(t) - \kappa_d(t)r_d\Big)\frac{\partial u}{\partial r_d} + \Big(\theta_f(t) - \kappa_f(t)r_f - \rho_{fS}\sigma_f(t)\gamma(t, s(t))\Big)\frac{\partial u}{\partial r_f} \\
& + \frac{1}{2}\gamma^2(t, s(t))s^2\frac{\partial^2 u}{\partial s^2} + \frac{1}{2}\sigma_d^2(t)\frac{\partial^2 u}{\partial r_d^2} + \frac{1}{2}\sigma_f^2(t)\frac{\partial^2 u}{\partial r_f^2} \\
& + \rho_{dS}\sigma_d(t)\gamma(t, s(t))s\frac{\partial^2 u}{\partial r_d \partial s} \\
& + \rho_{fS}\sigma_f(t)\gamma(t, s(t))s\frac{\partial^2 u}{\partial r_f \partial s} + \rho_{df}\sigma_d(t)\sigma_f(t)\frac{\partial^2 u}{\partial r_d \partial r_f} - r_d u = 0
\end{aligned}
$$

- Derivation: multi-dimensional Itô's formula
- Boundary conditions: Dirichlet-type "stopped process" boundary conditions (M. Dempster and J. Hutton, 1997)
- Backward PDE: the change of variable $\tau = T_{end} - t$
- Difficulties: high-dimensionality, cross-derivative terms

Outline

GPU overview and CUDA programming model

- GPU architecture: set of independent streaming multiprocessors
  - scalar processors
  - multi-threaded instruction unit (I/U)
  - shared memory
- CUDA programming environment
  - Host code on CPU, CUDA code on GPU (*device*)
  - Functions that run on GPUs are called *kernels*
  - Many copies of a kernel (*threads*) are executed concurrently
  - Single Instruction Multiple Threads - SIMT
- CUDA thread organization
  - A kernel is executed by a *grid* (2- or 3-D), which contain *threadblocks* (1-, 2- or 3-D)
  - Threads in the same threadblock can
    - Share data through the shared memory
    - Synchronize their executions
  - Threads from different blocks operate independently



Discretization

- Space: Second-order central finite differences on uniform mesh
- Time: ADI timestepping from τ_{m-1} to τ_m (Hundsdorfer and Verwer, 2003)

Phase 1: $$\mathbf{v}_0 = \mathbf{u}^{m-1} + \Delta\tau(\mathbf{A}^{m-1}\mathbf{u}^{m-1} + \mathbf{g}^{m-1}),$$

$$\underbrace{(\mathbf{I} - \frac{1}{2}\Delta\tau\mathbf{A}_i^m)}_{\widehat{\mathbf{A}}_i^m}\mathbf{v}_i = \underbrace{\mathbf{v}_{i-1} - \frac{1}{2}\Delta\tau\mathbf{A}_i^{m-1}\mathbf{u}^{m-1} + \frac{1}{2}\Delta\tau(\mathbf{g}_i^m - \mathbf{g}_i^{m-1})}_{\widehat{\mathbf{v}}_i}, \quad i = 1, 2, 3,$$

Phase 2: $$\widetilde{\mathbf{v}}_0 = \mathbf{v}_0 + \frac{1}{2}\Delta\tau(\mathbf{A}^m\mathbf{v}_3 - \mathbf{A}^{m-1}\mathbf{u}^{m-1}) + \frac{1}{2}\Delta\tau(\mathbf{g}^m - \mathbf{g}^{m-1}),$$

$$(\mathbf{I} - \frac{1}{2}\Delta\tau\mathbf{A}_i^m)\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{v}}_{i-1} - \frac{1}{2}\Delta\tau\mathbf{A}_i^m\mathbf{v}_3, \quad i = 1, 2, 3,$$

$$\mathbf{u}^m = \widetilde{\mathbf{v}}_3.$$

- $\mathbf{A}_0^m$: matrix of all mixed derivatives terms; $\mathbf{A}_i^m, i = 1, \ldots, 3$: matrices of the second-order spatial derivative in the s-, r_d-, and r_s- directions, respectively
- $\mathbf{g}_i^m, i = 0, \ldots, 3$: vectors obtained from the boundary conditions
- $\mathbf{A}^m = \sum_{i=0}^3 \mathbf{A}_i^m$; $\mathbf{g}^m = \sum_{i=0}^3 \mathbf{g}_i^m$

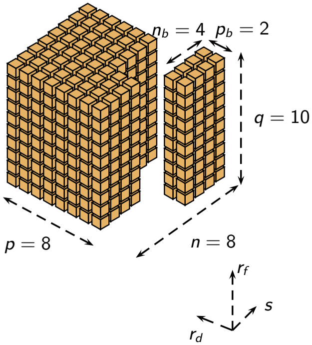
Parallel algorithm overview

- Focus on the parallelism within one timestep via a parallelization of the ADI scheme
- With respect to the CUDA implementation, the two phases of the ADI scheme are essentially the same $\Rightarrow$ focus on describing Phase 1.
- Main steps of Phase 1:
  - Step a.1: computes the matrices $\mathbf{A}_i^m$, $i = 0, 1, 2, 3$, the matrices $\widehat{\mathbf{A}}_i^m$, $i = 1, 2, 3$, the matrix-vector multiplications $\mathbf{A}_i^m \mathbf{u}^{m-1}$, $i = 0, 1, 2, 3$, and the vector $\mathbf{v}_0$;
  - Step a.2: computes $\widehat{\mathbf{v}}_1$ and solves $\widehat{\mathbf{A}}_1^m \mathbf{v}_1 = \widehat{\mathbf{v}}_1$;
  - Step a.3: computes $\widehat{\mathbf{v}}_2$ and solves $\widehat{\mathbf{A}}_2^m \mathbf{v}_2 = \widehat{\mathbf{v}}_2$;
  - Step a.4: computes $\widehat{\mathbf{v}}_3$ and solves $\widehat{\mathbf{A}}_3^m \mathbf{v}_3 = \widehat{\mathbf{v}}_3$;
- Steps a.2, a.3, and a.4: inherently parallelizable (block-diagonal, with tridiagonal blocks)
- Step a.1: $\mathbf{A}_i^m \mathbf{u}^{m-1}$, $i = 0, 1, 2, 3$, is more difficult to parallelize efficiently

Step a.1: matrix-vector mult. and matrix construction

Computational grid partitioning

- 3-D computational grid of size $n \times p \times q \Rightarrow$ 3-D blocks of size $n_b \times p_b \times q$
- each 3-D block consists of q tiles (2-D blocks of size $n_b \times p_b$)



Assignment of gridpoints

- invoke a grid of multiple $n_b \times p_b$ threadblocks
- each 3-D block is assigned to a 2-D threadblock
- each threadblock does a q-iteration loop, processing an $n_b \times p_b$ tile at each iteration

At each iteration, each threadblock

- loads its data from the global memory to the shared memory
- computes the respective entries of matrix-vector multiplications $\mathbf{A}_i^m \mathbf{u}^{m-1}$ and the respective row of the matrices $\widehat{\mathbf{A}}_i^m$
- copies new rows/values from the shared memory to the global memory

Steps a.2/a.3/a.4: independent tridiagonal solves

- Motivated by the block structure of the tridiagonal matrices $\widehat{\mathbf{A}}_i^m = \mathbf{I} - \frac{1}{2}\Delta\tau\mathbf{A}_i^m$
- Based on the parallelism arising from independent tridiagonal solutions, rather than the parallelism within each one
- Assign each tridiagonal system to one of the threads
- **Example:** $\underbrace{\left(\mathbf{I} - \frac{1}{2}\Delta\tau\mathbf{A}_1^m\right)}_{\widehat{\mathbf{A}}_1^m}\mathbf{v}_1 = \underbrace{\mathbf{v}_0 - \frac{1}{2}\Delta\tau\mathbf{A}_1^{m-1}\mathbf{u}^{m-1} + \frac{1}{2}\Delta\tau(\mathbf{g}_1^m - \mathbf{g}_1^{m-1})}_{\widehat{\mathbf{v}}_1}$
    1. Partition $\widehat{\mathbf{A}}_1^m$ and $\widehat{\mathbf{v}}_1$ into pq independent $n \times n$ tridiagonal systems
    2. Assign each tridiagonal system to one of pq threads.
    3. Use multiple 2-D threadblocks of identical size $r_t \times c_t$

Outline

Market Data

- Two economies: Japan (domestic) and US (foreign)
- Initial spot FX rate: $s(0) = 105$
- Interest rate curves, volatility parameters, correlations:

$$P_d(0,T) = \exp(-0.02 \times T) \quad \sigma_d(t) = 0.7\% \quad \kappa_d(t) = 0.0\%$$

$$P_f(0,T) = \exp(-0.05 \times T) \quad \sigma_f(t) = 1.2\% \quad \kappa_f(t) = 5.0\%$$

$$\rho_{df} = 25\%$$

$$\rho_{dS} = -15\%$$

$$\rho_{fS} = -15\%$$

- Local volatility function:

period (years)	$(\xi(t))$	$(\varsigma(t))$	period (years)	$(\xi(t))$	$(\varsigma(t))$
(0 0.5]	9.03%	-200%	(7 10]	13.30%	-24%
(0.5 1]	8.87%	-172%	(10 15]	18.18%	10%
(1 3]	8.42%	-115%	(15 20]	16.73%	38%
(3 5]	8.99%	-65%	(20 25]	13.51%	38%
(5 7]	10.18%	-50%	(25 30]	13.51%	38%

- Truncated computational domain:

$$\{(s, r_d, r_f) \in [0,S] \times [0,R_d] \times [0,R_f]\} \equiv \{[0,305] \times [0,0.06] \times [0,0.15]\}$$

Specification

Bermudan cancelable PRDC swaps

- Principal: N_d (JPY); Maturity: 30 years
- Details: paying annual PRDC coupon, receiving annual JPY LIBOR

Year	PRDC coupon	JPY LIBOR
1	$\max(c_f \frac{s(1)}{F(0,1)} - c_d, 0)N_d$	$L_d(0,1)N_d$
...	...	...
29	$\max(c_f \frac{s(29)}{F(0,29)} - c_d, 0)N_d$	$L_d(28,29)N_d$

- Leverage levels

level	low	medium	high
c_f	4.5%	6.25%	9.00%
c_d	2.25%	4.36%	8.10%

- The issuer has the right to cancel the swap on each of $\{T_\alpha\}_{\alpha=1}^{\beta-1}$, $\beta = 30(y)$

Architectures

- Host: Xeon running at 2.0GHz; Device: a NVIDIA Tesla S870 (four Tesla C870 GPUs, each has 16 multi-processors with 8 processors running at 1.35GHz, and 16 KB of shared memory)
- The tile sizes are chosen to be $n_b \times p_b \equiv 16 \times 4$ and $r_t \times c_t \equiv 16 \times 4$

Prices and convergence

leverage (c_d/c_f)	m (t)	n (s)	p (r_d)	q (r_f)	underlying swap value (%)	underlying swap change	underlying swap ratio	cancelable swap value (%)	cancelable swap change	cancelable swap ratio
low (50%)	4	24	12	12	-11.1510			11.2936		
	8	48	24	24	-11.1205	3.0e-4		11.2829	1.1e-4	
	16	96	48	48	-11.1118	8.6e-5	3.6	11.2806	2.3e-5	4.4
	32	192	96	96	**-11.1094**	2.4e-5	3.7	**11.2801**	5.8e-6	4.0
medium (70%)	4	24	12	12	-12.9418			13.6638		
	8	48	24	24	-12.7495	1.9e-3		13.8012	1.3e-3	
	16	96	48	48	-12.7033	4.6e-4	4.1	13.8399	3.9e-4	3.5
	32	192	96	96	**-12.6916**	1.2e-4	3.9	**13.8507**	1.1e-4	3.6
high (90%)	4	24	12	12	-11.2723			19.3138		
	8	48	24	24	-11.2097	6.2e-4		19.5689	2.5e-3	
	16	96	48	48	-11.1932	1.4e-4	3.8	19.6256	5.6e-4	4.4
	32	192	96	96	**-11.1889**	4.3e-5	3.8	**19.6402**	1.4e-4	3.8

Computed prices and convergence results for the underlying swap and cancelable swap with the FX skew model

Parallel speedup

m (t)	n (s)	p (r_d)	q (r_f)	**underlying swap (one Tesla C870)** value (%)	CPU time (s.)	GPU time (s.)	speed up
4	24	12	12	-11.1510	2.10	0.89	2.4
8	48	24	24	-11.1205	31.22	2.53	12.3
16	96	48	48	-11.1118	492.51	23.68	20.8
32	192	96	96	-11.1094	7870.27	356.12	**22.1**

m (t)	n (s)	p (r_d)	q (r_f)	**cancelable swap (two Tesla C870)** value (%)	CPU time (s.)	GPU time (s.)	speed up
4	24	12	12	11.2936	4.35	0.89	4.9
8	48	24	24	11.2828	63.98	2.53	25.2
16	96	48	48	11.2806	1016.33	23.68	42.9
32	192	96	96	11.2802	15796.95	356.12	**44.3**

Computed prices and timing results for the underlying swap and cancelable swap for the low-leverage case

FX skew impact - underlying swap

leverage (c_d/c_f)	Prices		
	FX skew	Log-normal	FX skew - log-normal
low (50%)	-11.1094	-9.0128	-2.0966
medium (70%)	-12.6916	-9.6773	-3.0143
high (90%)	-11.1889	-9.8538	-1.3351

- Prices are more negative (i.e. profits)
  - The issuer takes a short position in low strike FX call options.
  - Skew $\nearrow$ the implied volatility of low-strike options $\Rightarrow$ $\searrow$ value of the underlying.

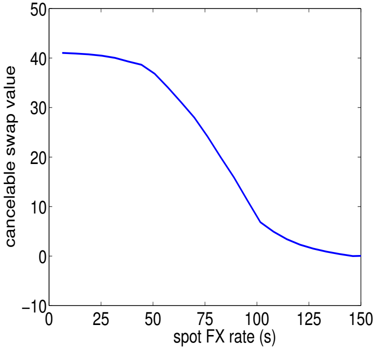
FX skew impact - cancelable swap

leverage (c_d/c_f)	Prices		
	FX skew	Log-normal	FX skew - log-normal
low (50%)	11.2801	13.3128	-2.0327
medium (70%)	13.8507	16.8985	-3.0478
high (90%)	19.6402	22.9523	-3.3121

Prices are less positive (i.e. profits)

- $s <$ forward FX rate: concave down (neg. gamma) $\Leftrightarrow$ short FX option positions
- $s >$ forward FX rate: concave up (pos. gamma) $\Leftrightarrow$ long FX option positions
- Skew impact:
  - higher vol. for low strikes (short pos. $\Rightarrow\searrow$ prices)
  - lower vol. for high strikes (long pos. $\Rightarrow\searrow$ prices)

$\Rightarrow$ both concave up and down parts are valued lower



$T_5 = 5$

forward FX $= 90.3$

Outline

Summary and future work

Summary

- GPU-based pricing of FX-IR exotics under a FX local volatility skew model via a PDE approach
- Speedup of 44 with two Tesla C870 for the cancelable swap
- Significant impact of FX skew

Future work

- Modeling: stochastic models/regime switch for the volatility of the spot FX rate, multi-factor models for the short rates
- Numerical methods: non-uniform/adaptive grids, higher-order ADI schemes
- Parallelization: extension to multi-GPU platforms

Related projects

- Exotic features: knockout, FX Target Redemption (TARN)
- Multi-asset American options
  - Penalty approach $\Rightarrow$ nonlinear PDE
  - GPU-based parallel Approximate Matrix Factorization for the solution of the linear system arising at each Newton iteration

Thank you!

1. D. M. Dang, C. C. Christara, K. R. Jackson and A. Lakhany (2009)
   A PDE pricing framework for cross-currency interest rate derivatives
   Available at http://ssrn.com/abstract=1502302

2. D. M. Dang (2009)
   Pricing of cross-currency interest rate derivatives on Graphics Processing Units
   Available at http://ssrn.com/abstract=1498563

3. D. M. Dang, C. C. Christara and K. R. Jackson (2010)
   GPU pricing of exotic cross-currency interest rate derivatives with a foreign exchange volatility skew model
   Available at http://ssrn.com/abstract=1549661

4. D. M. Dang, C. C. Christara and K. R. Jackson (2010)
   Parallel implementation on GPUs of ADI finite difference methods for parabolic PDEs with applications in finance
   Available at http://ssrn.com/abstract=1580057

More at http://ssrn.com/author=1173218

Step a.1: Forward Euler step

During the kth iteration, each threadblock

1. loads from the global memory into its shared memory the old data (vector $\mathbf{u}^{m-1}$) corresponding to the $(k+1)$st tile, and the associated halos (in the s- and r_d-directions), if any,
2. computes and stores new values for the kth tile using data of the $(k-1)$st, kth and $(k+1)$st tiles, and of the associated halos, if any,
3. copies the newly computed data of the kth tile from the shared memory to the global memory, and frees the shared memory locations taken by the data of the $(k-1)$st tile, and associated halos, if any, so that they can be used in the next iteration.

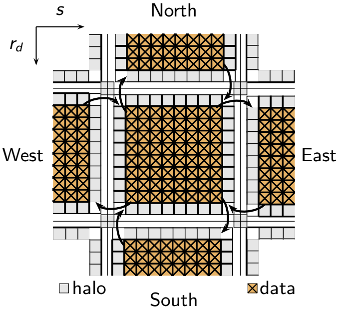


Figure: An example of $n_b \times p_b = 8 \times 8$ tiles with halos.

Memory coalescing: fully coalesced loading for interior data of a tile and halos along the s-direction (North and South), but not for halos along the r_d-direction (East and West)