Pricing of cross-currency interest rate derivatives on Graphics Processing Units

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

Joint work with Christina Christara and Ken Jackson

Workshop on Parallel and Distributed Computing in Finance
IEEE International Parallel & Distributed Processing Symposium
Atlanta, USA, April 19 – 23, 2010

PRDC swaps

- Long-dated swaps ($\geq$ 30 years)
- Two currencies: domestic and foreign (unit zero-coupon bond prices P_d and P_f)
- PRDC coupons in exchange for domestic LIBOR payments (*funding leg*)
- Two parties: the *issuer* (pays PRDC coupons) and the *investor* (pays LIBOR)
- PRDC coupon and LIBOR rates are applied on the domestic currency principal N_d

Tenor structure: $T_0 < T_1 < \ldots < T_{\beta-1} < T_\beta, \quad \nu_\alpha \equiv \nu(T_{\alpha-1}, T_\alpha) = T_\alpha - T_{\alpha-1}$
At each of the times T_α, $\alpha = 1, \ldots, \beta - 1$, the issuer

- receives $\nu_\alpha N_d L_d(T_{\alpha-1}, T_\alpha)$, where $L_d(T_{\alpha-1}, T_\alpha) = \dfrac{1 - P_d(T_{\alpha-1}, T_\alpha)}{\nu(T_{\alpha-1}, T_\alpha) P_d(T_{\alpha-1}, T_\alpha)}$
- pays PRDC coupon amount $\nu_\alpha N_d C_\alpha$, where the coupon rate C_α has the structure

$$C_\alpha = \min\left(\max\left(c_f \frac{s(T_\alpha)}{f_\alpha} - c_d, b_f\right), b_c\right)$$

  - $s(T_\alpha)$: the spot FX-rate at time T_α
  - f_α: scaling factor, usually is set to the forward FX rate $F(0, T_\alpha) = \dfrac{P_f(0, T_\alpha)}{P_d(0, T_\alpha)} s(0)$
  - 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

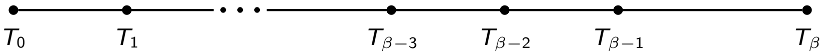
$$C_\alpha = h_\alpha \max(s(T_\alpha) - k_\alpha, 0), \quad h_\alpha = \frac{c_f}{f_\alpha}, k_\alpha = \frac{f_\alpha c_d}{c_f}$$

Bermudan cancelable PRDC swaps

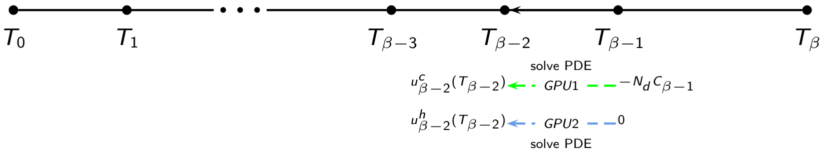
The issuer has the right to cancel the underlying swap at *any* of the times $\{T_\alpha\}_{\alpha=1}^{\beta-1}$ *after* the occurrence of any exchange of fund flows scheduled on that date.

- Observation: 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 framework: dividing the pricing of a Bermudan 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
- Notations
  - $u_\alpha^c(t)$ and $u_\alpha^f(t)$: value at time t of the coupon and the LIBOR part scheduled after T_α, respectively
  - $u_\alpha^h(t)$: value at time t of the offsetting Bermudan swaption that has only the dates $\{T_{\alpha+1}, \ldots, T_{\beta-1}\}$ as exercise opportunities
  - $u_\alpha^e(t)$: value at time t of all fund flows in the offsetting swap scheduled after T_α
  - $u_{\beta-1}^h(T_{\beta-1}) = u_{\beta-1}^e(T_{\beta-1}) = 0$
  - Note: $u_\alpha^h(T_\alpha)$ is the **"hold value"** and $u_\alpha^e(T_\alpha)$ is the **"exercise value"** of the option at time T_α

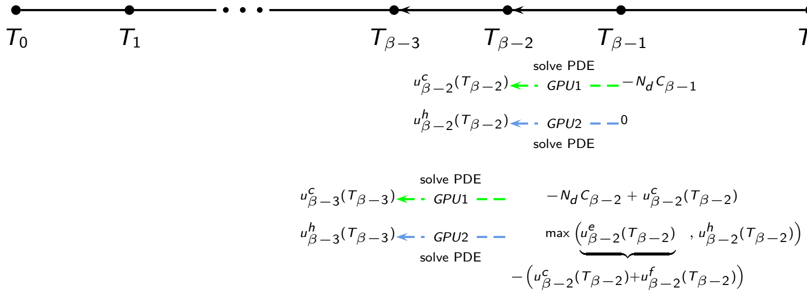
Backward pricing algorithm



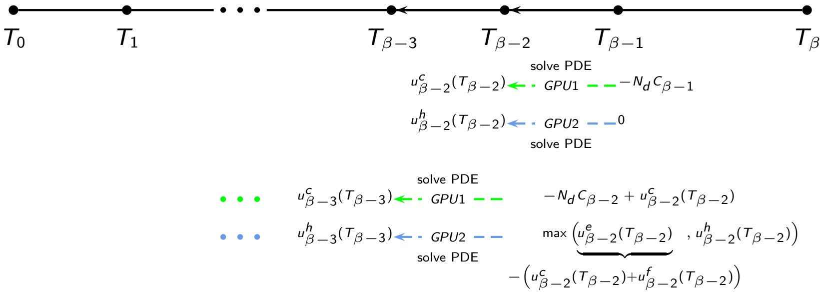
Backward pricing algorithm



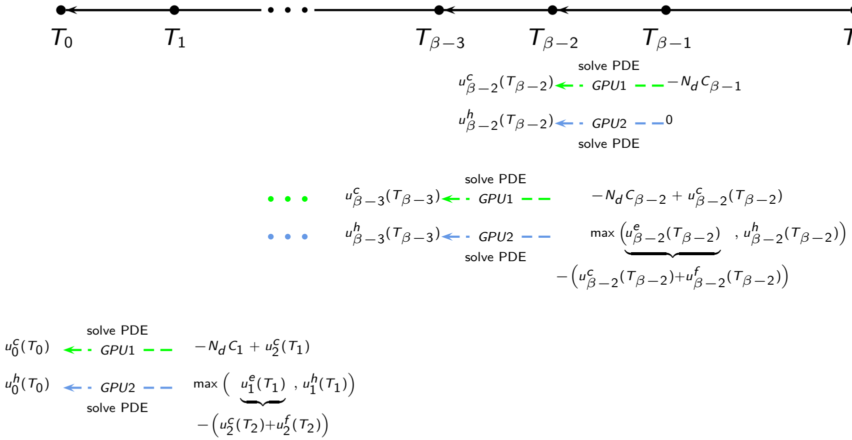
Backward pricing algorithm



Backward pricing algorithm



Backward pricing algorithm



- $u^f_\alpha(T_\alpha)$: obtained by the "fixed notional" method, not by solving a PDE
- Price of the underlying PRDC swap: $u^f_0(T_0) + u^c_0(T_0)$
- Price of the Bermudan cancelable PRDC swap: $(u^f_0(T_0) + u^c_0(T_0)) + u^h_0(T_0)$

The pricing model

Consider the following model under <u>domestic</u> risk neutral measure

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

- $r_i(t), i = d, f$: 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

Let $u = u(s, r_d, r_f, t)$ be the value of a security at time t, with a terminal payoff measurable with respect to the σ-algebra at maturity time T_{end} and without intermediate payments. On $\mathbb{R}_+^3 \times [T_{start}, T_{end})$, u satisfies the PDE

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

- Derivation: Multi-dimensional Itô's formula
- Boundary conditions: Dirichlet-type "stopped process" boundary conditions
- Backward PDE: the change of variable $\tau = T_{end} - t$
- Difficulties: High-dimensionality, cross-derivative terms

Discretization

- Space: Second-order central finite differences on uniform mesh
- Time: ADI technique based on Hundsdorfer and Verwer (HV) approach
  - $\mathbf{u}^m$: the vector of approximate values
  - $\mathbf{A}_0^m$: 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
  - $\mathbf{A}^m = \sum_{i=0}^{3} \mathbf{A}_i^m$; $\mathbf{g}^m = \sum_{i=0}^{3} \mathbf{g}_i^m$

Timestepping HV scheme from time t_{m-1} to t_m:

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:

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

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

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

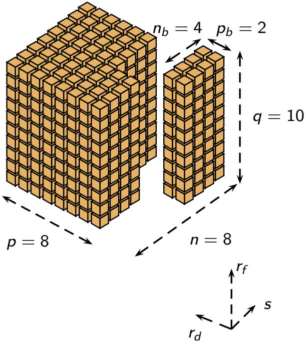
Parallel algorithm overview

- Focus on the parallelism within one timestep via a parallelization of the HV scheme
- With respect to the CUDA implementation, the two phases of the HV scheme are essentially the same. Hence, we focus on describing the parallelization of the first phase.
- 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 products $\mathbf{A}_i^m \mathbf{u}^{m-1}$, $i = 0, 1, 2, 3$, 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 are inherently parallelizable (block-diagonal, with tridiagonal blocks)
- Step a.1, on the other hand, the computation of the products $\mathbf{A}_i^m \mathbf{u}^{m-1}$ is more difficult to parallelize efficiently.

Phase 1 - Step a.1: Overview

Grid partitioning/assignment of gridpoints to threads

- computational grid of size $n \times p \times q$ is partitioned into 3-D blocks of size $n_b \times p_b \times q$, each of which can be viewed as consisting of q 2-D blocks, referred to as *tiles*, of size $n_b \times p_b$.
- A grid of $\texttt{ceil}(n/n_b) \times \texttt{ceil}(p/p_b)$ threadblocks is invoked, each of which consists of an $n_b \times p_b$ array of threads.
- Each threadblock does a q-iteration loop, processing an $n_b \times p_b$ tile at each iteration, i.e. each thread does a q-iteration loop, processing one gridpoint at each iteration



Computation details of a threadblock at each iteration

- loads from the global memory to its shared memory the components of $\mathbf{u}^{m-1}$ corresponding to a tile, and the associated halo values;
- computes the respective rows of matrices $\mathbf{A}_i^m$ and $\widehat{\mathbf{A}}_i^m$, and respective entries of $\mathbf{A}_i^m \mathbf{u}^{m-1}$ and $\mathbf{v_0}$
- copies new rows and new values from the shared memory to the global memory

Phase 1 - Step a.1: Computation of $\mathbf{v_0}$

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.

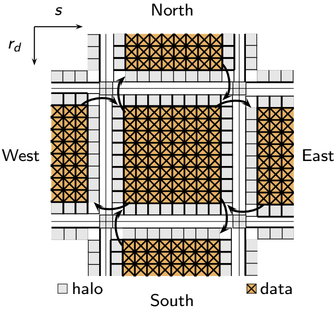


Figure: An example of $n_b \times p_b = 8 \times 8$ tiles with halos.

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)

Phase 1 - Steps a.2/a.3/a.4: 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
- When solved in one direction, the data are partitioned with respect to the other two
- Assign each tridiagonal system to one of the threads
- **Example:** $\underbrace{(\mathbf{I} - \frac{1}{2}\Delta\tau\mathbf{A}_1^m)}_{\widehat{\mathbf{A}}_1^m}\mathbf{v}_1 = \underbrace{\mathbf{v}_0 - \frac{1}{2}\Delta\tau\mathbf{A}_1^{m-1}\mathbf{u}^{m-1} + \frac{1}{2}\Delta\tau(\mathbf{g}_1^m - \mathbf{g}_1^{m-1})}_{\widehat{\mathbf{v}}_1},$
  i. Partition $\widehat{\mathbf{A}}_1^m$ and $\widehat{\mathbf{v}}_1$ into pq independent $n \times n$ tridiagonal systems
  ii. Assign each tridiagonal system to one of pq threads.
  iii. Use multiple 2-D threadblocks of identical size $r_t \times c_t$, i.e. a 2-D grid of threadblocks of size $\texttt{ceil}(\frac{p}{r_t}) \times \texttt{ceil}(\frac{q}{c_t})$ is invoked.
- **Memory coalescence:** fully achieved for the tridiagonal solves in the r_d- and r_f-directions, but not in the s-direction.

  Could be improved by renumbering gridpoints between steps of the first phase.

Market Data

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

$$P_d(0,T) = \exp(-0.02 \times T) \quad \sigma_d(t) = 0.7\% \quad \kappa_d(t) = 0.0\% \qquad \rho_{df} = 25\%$$

$$P_f(0,T) = \exp(-0.05 \times T) \quad \sigma_f(t) = 1.2\% \quad \kappa_f(t) = 5.0\% \qquad \rho_{dS} = -15\% \quad \rho_{fS} = -15\%$$

- 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); Settlement/Maturity dates: 23 Apr. 2010/23 Nov. 2040
- Details: paying annual PRDC coupon, receiving JPY LIBOR

Year	coupon (FX options)	funding leg
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 level

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

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

Architectures

- Xeon running at 2.0GHz host system with a NVIDIA Tesla S870 (four Tesla C870 GPUs, 16 multi-processors, each containing 8 processors running at 1.35GHz, and 16 KB of shared memory)
- The tile sizes are chosen to be $n_b \times p_b \equiv 16 \times 4$ (for Step a.1), and $r_t \times c_t \equiv 16 \times 4$ (for Steps a.2, a.3, a.4), which appears to be optimal on Tesla C870.

Prices and convergence

leverage	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	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	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	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

				underlying swap (one Tesla C870)			
m (t)	n (s)	p (r_d)	q (r_f)	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

				cancelable swap (two Tesla C870)			
m (t)	n (s)	p (r_d)	q (r_f)	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

Summary and future work

Summary

- GPU-based algorithm for pricing exotic cross-currency interest rate derivatives under a FX local volatility skew model via a PDE approach, with strong emphasis on Bermudan cancelable PRDC swaps
- The parallel algorithm is based on
  i. partitioning the pricing of cancelable PRDC swaps into two entirely independent pricing subproblems in each period of the tenor structure
  ii. efficient parallelization on GPUs of the HV ADI scheme at each timestep for the efficient solution of each of these subproblems
- Results indicate speedup of 44 with two Tesla C870, for the cancellable swap.

Ongoing projects

- Exotic features: knockout, FX-TARN (interesting)
- GPU-based parallel methods for pricing multi-asset American options (penalty + ADI)

Future work

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

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