



# Stochastic Grid Bundling Method

## *GPU Acceleration*

Delft University of Technology - Centrum Wiskunde & Informatica

Álvaro Leitao Rodríguez and Cornelis W. Oosterlee  
London - December 17, 2015

# Outline

- 1 Introduction
- 2 Basket Bermudan Options
- 3 Stochastic Grid Bundling Method
- 4 Continuation value computation: new approach
- 5 Parallel GPU Implementation
- 6 Results
- 7 Conclusions

# Introduction and Motivation

- Originally introduced by Jain and Oosterlee in 2013.
- Multi-dimensional early-exercise option contracts.
- Increase the dimensionality.
- SGBM becomes very expensive.
- Solution: parallelization of the method.
- GPU computing (GPGPU).

# Basket Bermudan Options

- Right to exercise at a set of times:  
 $t \in [t_0 = 0, \dots, t_m, \dots, t_M = T]$ .
- $d$ -dimensional underlying process:  $\mathbf{S}_t = (S_t^1, \dots, S_t^d) \in \mathbb{R}^d$ .
- Intrinsic value of the option:  $h_t := h(\mathbf{S}_t)$ .
- The value of the option at the terminal time  $T$ :

$$V_T(\mathbf{S}_T) = \max(h(\mathbf{S}_T), 0).$$

- The conditional continuation value  $Q_{t_m}$ , i.e. the discounted expected payoff at time  $t_m$ :

$$Q_{t_m}(\mathbf{S}_{t_m}) = D_{t_m} \mathbb{E} [V_{t_{m+1}}(\mathbf{S}_{t_{m+1}}) | \mathbf{S}_{t_m}].$$

- The Bermudan option value at time  $t_m$  and state  $\mathbf{S}_{t_m}$ :

$$V_{t_m}(\mathbf{S}_{t_m}) = \max(h(\mathbf{S}_{t_m}), Q_{t_m}(\mathbf{S}_{t_m})).$$

- Value of the option at the initial state  $\mathbf{S}_{t_0}$ , i.e.  $V_{t_0}(\mathbf{S}_{t_0})$ .

# Basket Bermudan options scheme

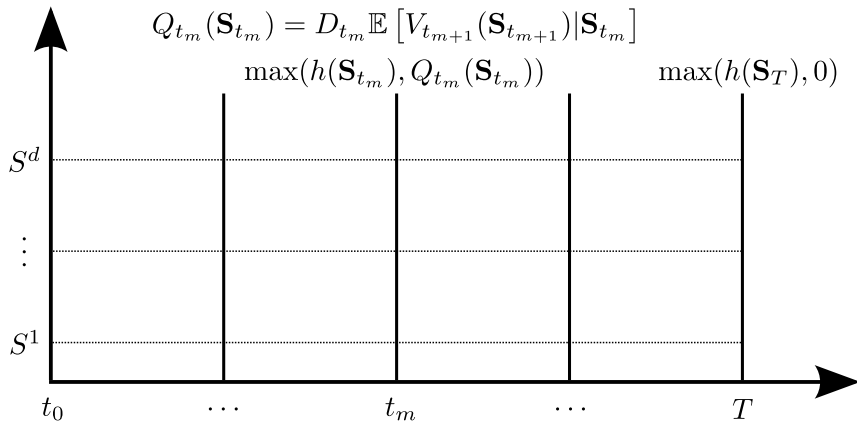


Figure: d-dimensional Bermudan option

# Stochastic Grid Bundling Method

- Simulation and regression-based method.
- Forward in time: Monte Carlo simulation.
- Backward in time: Early-exercise policy computation.
- Step I: Generation of stochastic grid points

$$\{\mathbf{S}_{t_0}(n), \dots, \mathbf{S}_{t_M}(n)\}, \quad n = 1, \dots, N.$$

- Step II: Option value at terminal time  $t_M = T$

$$V_{t_M}(\mathbf{S}_{t_M}) = \max(h(\mathbf{S}_{t_M}), 0).$$

## Stochastic Grid Bundling Method (II)

- Backward in time,  $t_m$ ,  $m \leq M$ ;
- Step III: Bundling into  $\nu$  non-overlapping sets or partitions

$$\mathcal{B}_{t_{m-1}}(1), \dots, \mathcal{B}_{t_{m-1}}(\nu)$$

- Step IV: Parameterizing the option values

$$Z(\mathbf{S}_{t_m}, \alpha_{t_m}^\beta) \approx V_{t_m}(\mathbf{S}_{t_m}).$$

- Step V: Computing the continuation and option values at  $t_{m-1}$

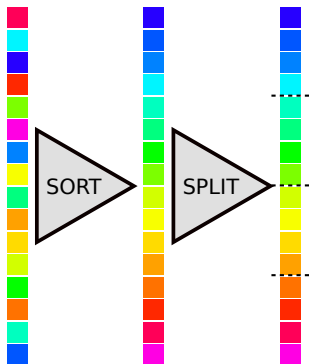
$$\hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \mathbb{E}[Z(\mathbf{S}_{t_m}, \alpha_{t_m}^\beta) | \mathbf{S}_{t_{m-1}}(n)].$$

The option value is then given by:

$$\hat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max(h(\mathbf{S}_{t_{m-1}}(n)), \hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n))).$$

# Bundling

- Original: Iterative process (K-means clustering).
- Problems: Too expensive (time and memory) and distribution.
- New technique: Equal-partitioning. Efficient for parallelization.
- Two stages: sorting and splitting.





# Parametrizing the option value

- Basis functions  $\phi_1, \phi_2, \dots, \phi_K$ .
- In our case,  $Z(\mathbf{S}_{t_m}, \alpha_{t_m}^\beta)$  depends on  $\mathbf{S}_{t_m}$  only through  $\phi_k(\mathbf{S}_{t_m})$ :

$$Z(\mathbf{S}_{t_m}, \alpha_{t_m}^\beta) = \sum_{k=1}^K \alpha_{t_m}^\beta(k) \phi_k(\mathbf{S}_{t_m}).$$

- Computation of  $\alpha_{t_m}^\beta$  (or  $\hat{\alpha}_{t_m}^\beta$ ) by least squares regression.
- The  $\alpha_{t_m}^\beta$  determines the early-exercise policy.
- The continuation value:

$$\begin{aligned} \hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) &= D_{t_{m-1}} \mathbb{E} \left[ \left( \sum_{k=1}^K \hat{\alpha}_{t_m}^\beta(k) \phi_k(\mathbf{S}_{t_m}) \right) \mid \mathbf{S}_{t_{m-1}} \right] \\ &= D_{t_{m-1}} \sum_{k=1}^K \hat{\alpha}_{t_m}^\beta(k) \mathbb{E} [\phi_k(\mathbf{S}_{t_m}) \mid \mathbf{S}_{t_{m-1}}]. \end{aligned}$$

# Basis functions

- Choose  $\phi_k$ : the expectations  $\mathbb{E} [\phi_k(\mathbf{S}_{t_m}) | \mathbf{S}_{t_{m-1}}]$  should be easy to calculate.
- The intrinsic value of the option (payoff),  $h(\cdot)$ , is usually an important and useful basis function.
- For  $\mathbf{S}_t$  following a GBM: expectations available analytically for geometric and arithmetic basket Bermudan options.
- Other underlying models: new proposed approach based on the *discrete characteristic function*.

## Estimating the option value

- SGBM has been developed as *duality-based method*.
- Provide two estimators (confidence interval).
- *Direct estimator* (high-biased estimation):

$$\widehat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max \left( h(\mathbf{S}_{t_{m-1}}(n)), \widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) \right),$$

$$\mathbb{E}[\widehat{V}_{t_0}(\mathbf{S}_{t_0})] = \frac{1}{N} \sum_{n=1}^N \widehat{V}_{t_0}(\mathbf{S}_{t_0}(n)).$$

- *Path estimator* (low-biased estimation):

$$\widehat{\tau}^*(\mathbf{S}(n)) = \min \{ t_m : h(\mathbf{S}_{t_m}(n)) \geq \widehat{Q}_{t_m}(\mathbf{S}_{t_m}(n)), m = 1, \dots, M \},$$
$$v(n) = h(\mathbf{S}_{\widehat{\tau}^*(\mathbf{S}(n))}),$$

$$\underline{V}_{t_0}(\mathbf{S}_{t_0}) = \lim_{N_L} \frac{1}{N_L} \sum_{n=1}^{N_L} v(n).$$

# SGBM - schematic algorithm

---

**Data:**  $S_{t_0}, X, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$

Pre-Bundling (only in k-means case).

Generation of the grid points (Monte Carlo). Step I.

Option value at terminal time  $t = M$ . Step II.

**for** *Time*  $t = (M - 1) \dots 1$  **do**

    Bundling. Step III.

**for** *Bundle*  $\beta = 1 \dots \nu$  **do**

        Exercise policy (Regression). Step IV.

        Continuation value. Step V.

        Direct estimator. Step V.

Generation of the grid points (Monte Carlo). Step I.

Option value at terminal time  $t = M$ . Step II.

**for** *Time*  $t = (M - 1) \dots 1$  **do**

    Bundling. Step III.

**for** *Bundle*  $\beta = 1 \dots \nu$  **do**

        Continuation value. Step V.

        Path estimator. Step V.

## Cont. value computation: new approach

- More generally applicable. More involved models or options.
- First discretize, then derive the *discrete* characteristic function.

$$S_{t_{m+1}}^1 = S_{t_m}^1 + \mu_1(\mathbf{S}_{t_m})\Delta t + \sigma_1(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^1,$$

$$S_{t_{m+1}}^2 = S_{t_m}^2 + \mu_2(\mathbf{S}_{t_m})\Delta t + \rho_{1,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^1 + L_{2,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^2,$$

...

$$S_{t_{m+1}}^d = S_{t_m}^d + \mu_d(\mathbf{S}_{t_m})\Delta t + \rho_{1,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^1 + L_{2,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^2 + \dots + L_{d,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}_{t_{m+1}}^d,$$

- By definition, the  $d$ -variate discrete characteristic function:

$$\begin{aligned}\psi_{\mathbf{S}_{t_{m+1}}}(u_1, u_2, \dots, u_d | \mathbf{S}_{t_m}) &= \mathbb{E} \left[ \exp \left( \sum_{j=1}^d iu_j S_{t_{m+1}}^j \right) | \mathbf{S}_{t_m} \right] \\ &= \mathbb{E} \left[ \exp \left( \sum_{j=1}^d iu_j \left( S_{t_m}^j + \mu_j(\mathbf{S}_{t_m})\Delta t + \sigma_j(\mathbf{S}_{t_m}) \sum_{k=1}^j L_{k,j} \Delta\tilde{W}_{t_{m+1}}^k \right) \right) | \mathbf{S}_{t_m} \right] \\ &= \exp \left( \sum_{j=1}^d iu_j (S_{t_m}^j + \mu_j(\mathbf{S}_{t_m})\Delta t) \right) \cdot \prod_{k=1}^d \left( \mathbb{E} \left[ \exp \left( \sum_{j=k}^d iu_j L_{k,j} \sigma_j(\mathbf{S}_{t_m}) \Delta\tilde{W}_{t_{m+1}}^k \right) \right] \right) \\ &= \exp \left( \sum_{j=1}^d iu_j (S_{t_m}^j + \mu_j(\mathbf{S}_{t_m})\Delta t) \right) \cdot \prod_{k=1}^d \left( \psi_{\mathcal{N}(0, \Delta t)} \left( \sum_{j=k}^d u_j L_{k,j} \sigma_j(\mathbf{S}_{t_m}) \right) \right),\end{aligned}$$

## Cont. value computation: new approach

- Joint moments of the product:

$$\begin{aligned} M_{\mathbf{S}_{t_{m+1}}} &= \mathbb{E} \left[ \left( S_{t_{m+1}}^1 \right)^{c_1} \left( S_{t_{m+1}}^2 \right)^{c_2} \cdots \left( S_{t_{m+1}}^d \right)^{c_d} \mid \mathbf{S}_{t_m} \right] \\ &= (-i)^{c_1+c_2+\cdots+c_d} \left[ \frac{\partial^{c_1+c_2+\cdots+c_d} \psi_{\mathbf{S}_{t_{m+1}}}(\mathbf{u} \mid \mathbf{S}_{t_m})}{\partial u_1^{c_1} \partial u_2^{c_2} \cdots \partial u_d^{c_d}} \right]_{\mathbf{u}=0}, \end{aligned}$$

- So, if the basis functions are the product of asset processes:

$$\phi_k(\mathbf{S}_{t_m}) = \left( \prod_{\delta=1}^d S_{t_m}^{\delta} \right)^{k-1}, \quad k = 1, \dots, K,$$

- However, this approx. is, in general, worse than analytic.
- Feasible thank to the GPU implementation: time steps  $\uparrow\uparrow$ .

# Parallel SGBM on GPU

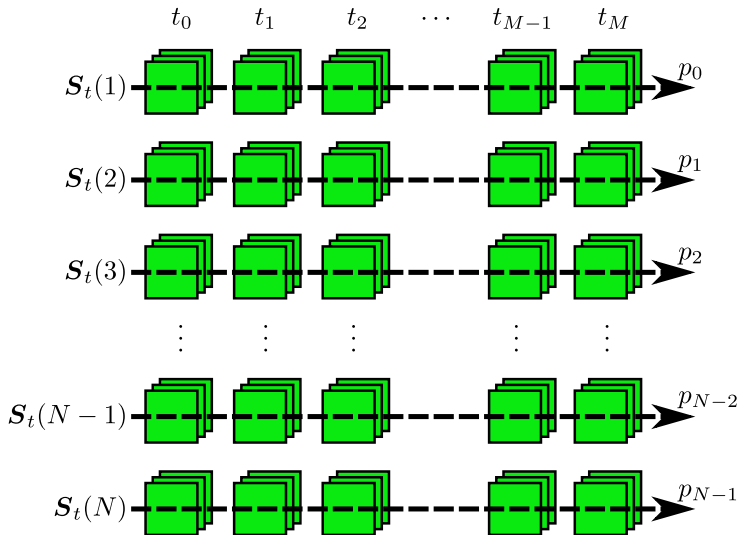
- NVIDIA CUDA platform.
- Parallel strategy: two parallelization stages:
  - Forward: Monte Carlo simulation.
  - Backward: Bundles → Opportunity of parallelization.
- Novelty in early-exercise option pricing methods.
- Two implementations → K-means vs. Equal-partitioning:
  - K-means: sequential parts.
  - K-means: transfers between CPU and GPU cannot be avoided.
  - K-means: all data need to be stored.
  - K-means: Load-balancing.
  - Equal-partitioning: fully parallelizable.
  - Equal-partitioning: No transfers.
  - Equal-partitioning: efficient memory use.

# Parallel SGBM on GPU - Forward in time

- One GPU thread per Monte Carlo simulation.
- Random numbers “on the fly”: cuRAND library.
- Compute intermediate results:
  - Expectations.
  - Intrinsic value of the option.
  - Equal-partitioning: sorting criterion calculations.
- Intermediate results in the registers: fast memory access.
- Original bundling: all the data still necessary.



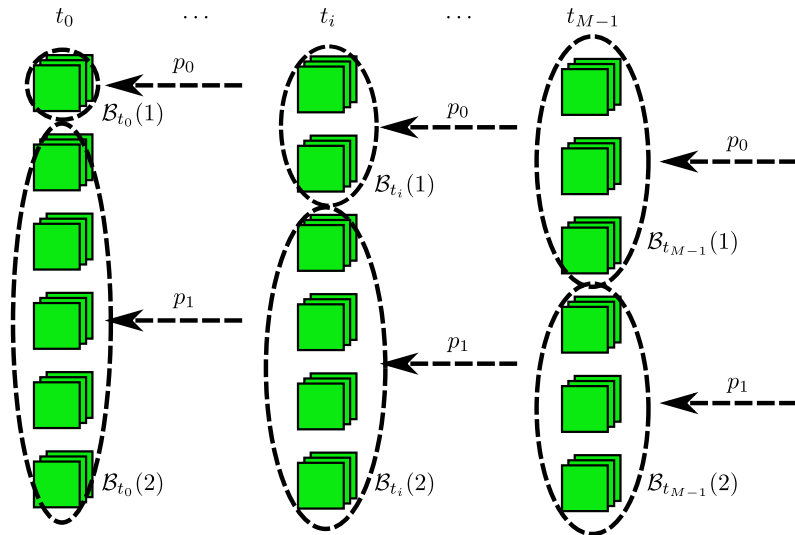
# Parallel SGBM on GPU - Forward in time



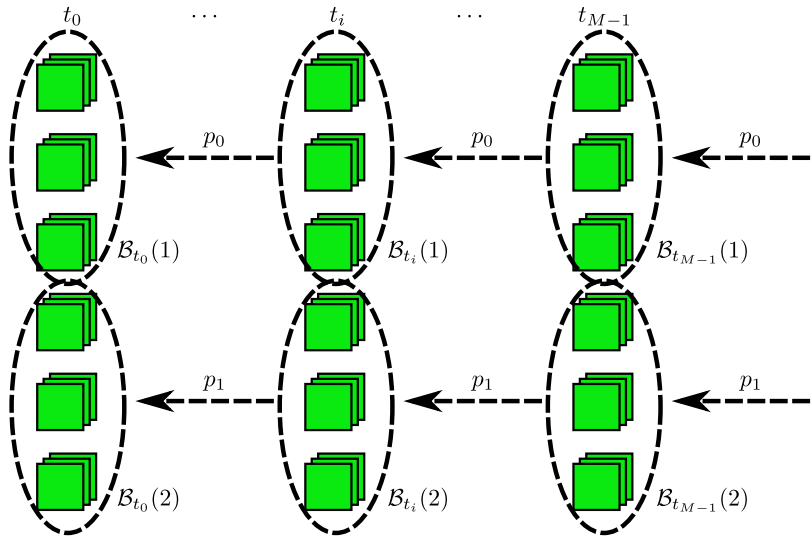
# Parallel SGBM on GPU - Backward in time

- One parallelization stage per exercise time step.
- Sort w.r.t bundles: efficient memory access.
- Parallelization in bundles.
- Each bundle calculations (option value and early-exercise policy) in parallel.
- All GPU threads collaborate in order to compute the continuation value.
- Path estimator: One GPU thread per path (the early-exercise policy is already computed).
- Final reduction: Thrust library.

# Parallel SGBM on GPU - Backward in time



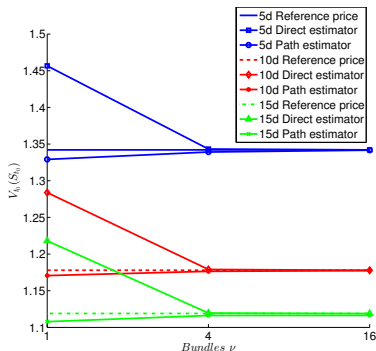
# Parallel SGBM on GPU - Backward in time



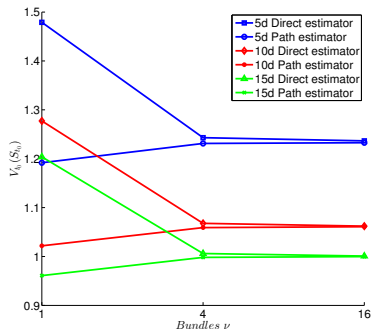
# Results

- Accelerator Island system of Cartesius Supercomputer.
  - Intel Xeon E5-2450 v2.
  - NVIDIA Tesla K40m.
  - C-compiler: GCC 4.4.7.
  - CUDA version: 5.5.
- Geometric and arithmetic basket Bermudan put options:  
 $\mathbf{S}_{t_0} = (40, \dots, 40) \in \mathbb{R}^d$ ,  $X = 40$ ,  $r_t = 0.06$ ,  
 $\sigma = (0.2, \dots, 0.2) \in \mathbb{R}^d$ ,  $\rho_{ij} = 0.25$ ,  $T = 1$  and  $M = 10$ .
- Basis functions:  $K = 3$ .
- Multi-dimensional Geometric Brownian Motion.
- Euler discretization,  $\delta t = T/M$ .

# Equal-partitioning: convergence test



(a) Geometric basket put option



(b) Arithmetic basket put option

Figure: Convergence with equal-partitioning bundling technique. Test configuration:  $N = 2^{18}$  and  $\Delta t = T/M$ .

# Speedup

<b>Geometric basket Bermudan option</b>						
	k-means			equal-partitioning		
	$d = 5$	$d = 10$	$d = 15$	$d = 5$	$d = 10$	$d = 15$
C	604.13	1155.63	1718.36	303.26	501.99	716.57
CUDA	35.26	112.70	259.03	8.29	9.28	10.14
Speedup	17.13	10.25	6.63	36.58	54.09	70.67

<b>Arithmetic basket Bermudan option</b>						
	k-means			equal-partitioning		
	$d = 5$	$d = 10$	$d = 15$	$d = 5$	$d = 10$	$d = 15$
C	591.91	1332.68	2236.93	256.05	600.09	1143.06
CUDA	34.62	126.69	263.62	8.02	11.23	15.73
Speedup	17.10	10.52	8.48	31.93	53.44	72.67

**Table:** SGBM total time (s) for the C and CUDA versions. Test configuration:  $N = 2^{22}$ ,  $\Delta t = T/M$  and  $\nu = 2^{10}$ .

## Speedup - High dimensions

<b>Geometric basket Bermudan option</b>						
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	337.61	476.16	620.11	337.06	475.12	618.98
CUDA	4.65	6.18	8.08	4.71	6.26	8.16
Speedup	72.60	77.05	76.75	71.56	75.90	75.85

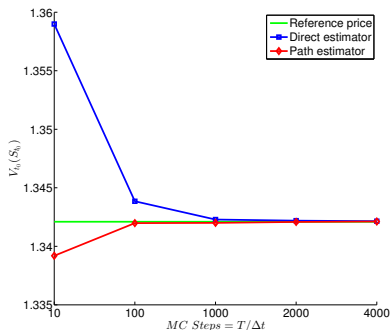
  

<b>Arithmetic basket Bermudan option</b>						
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	993.96	1723.79	2631.95	992.29	1724.60	2631.43
CUDA	11.14	17.88	26.99	11.20	17.94	27.07
Speedup	89.22	96.41	97.51	88.60	96.13	97.21

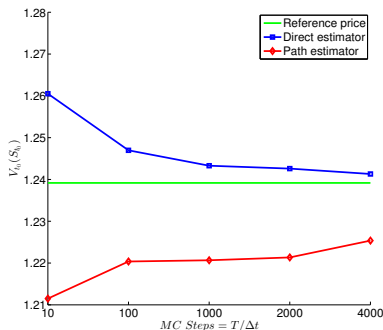
**Table:** SGBM total time (s) for a high-dimensional problem with equal-partitioning. Test configuration:  $N = 2^{20}$  and  $\Delta t = T/M$ .



# Cont. value computation: New approach



(a) Geometric basket put option



(b) Arithmetic basket put option

Figure: CEV model convergence,  $\gamma = 1.0$ . Test configuration:  $N = 2^{16}$ ,  $\nu = 2^{10}$  and  $d = 5$ .

## Cont. value computation: New approach

<b>Geometric basket Bermudan option</b>				
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000291	0.029395	0.276030	1.342147
SGBM PE	0.000274	0.029322	0.275131	1.342118

<b>Arithmetic basket Bermudan option</b>				
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000289	0.029089	0.267943	1.241304
SGBM PE	0.000288	0.028944	0.267214	1.225359

**Table:** CEV option pricing. Test configuration:  $N = 2^{16}$ ,  $\Delta t = T/4000$ ,  $\nu = 2^{10}$  and  $d = 5$ .

# Conclusions

- Efficient parallel GPU implementation.
- Extend the SGBM's applicability: Increasing dimensionality.
- New bundling technique.
- More general approach to compute the continuation value.
- Future work:
  - Explore the new CUDA 7 features: cuSOLVER (QR factorization).
  - CVA calculations.

# References



Shashi Jain and Cornelis W. Oosterlee.

The Stochastic Grid Bundling Method: Efficient pricing of Bermudan options and their Greeks.

*Applied Mathematics and Computation*, 269:412–431, 2015.



Álvaro Leitao and Cornelis W. Oosterlee.

GPU Acceleration of the Stochastic Grid Bundling Method for Early-Exercise options.

*International Journal of Computer Mathematics*, 92(12):2433–2454, 2015.

# Acknowledgements



Thank you for your attention

## Appendix

- Geo. basket Bermudan option - Basis functions:

$$\phi_k(\mathbf{S}_{t_m}) = \left( \left( \prod_{\delta=1}^d S_{t_m}^{\delta} \right)^{\frac{1}{d}} \right)^{k-1}, \quad k = 1, \dots, K,$$

- The expectation can directly be computed as:

$$\mathbb{E} [\phi_k(\mathbf{S}_{t_m}) | \mathbf{S}_{t_{m-1}}(n)] = \left( P_{t_{m-1}}(n) e^{\left( \bar{\mu} + \frac{(k-1)\bar{\sigma}^2}{2} \right) \Delta t} \right)^{k-1},$$

where,

$$P_{t_{m-1}}(n) = \left( \prod_{\delta=1}^d S_{t_{m-1}}^{\delta}(n) \right)^{\frac{1}{d}}, \quad \bar{\mu} = \frac{1}{d} \sum_{\delta=1}^d \left( r - q_{\delta} - \frac{\sigma_{\delta}^2}{2} \right), \quad \bar{\sigma}^2 = \frac{1}{d^2} \sum_{p=1}^d \left( \sum_{q=1}^d C_{pq}^2 \right)^2$$

## Appendix

- Arith. basket Bermudan option - Basis functions:

$$\phi_k(\mathbf{S}_{t_m}) = \left( \frac{1}{d} \sum_{\delta=1}^d S_{t_m}^{\delta} \right)^{k-1}, \quad k = 1, \dots, K.,$$

- The summation can be expressed as a linear combination of the products:

$$\left( \sum_{\delta=1}^d S_{t_m}^{\delta} \right)^k = \sum_{k_1+k_2+\dots+k_d=k} \binom{k}{k_1, k_2, \dots, k_d} \prod_{1 \leq \delta \leq d} \left( S_{t_m}^{\delta} \right)^{k_{\delta}},$$

- And the expression for Geometric basket option can be applied.