CPU及びGPUのローカル・ボラティリティFXバスケットオプション

Jacques du Toit* & Isabel Ehrlichy†

要約

私たちは10ファクターのローカル・ボラティリティモデルによって決定される，10個の外国為替レートのバスケットオプションを研究しています。私たちはモンテカルロシミュレーションを用いてオプションの価格を設定し，高品質のIntel CPU と NVIDIA GPU 上でアルゴリズムの高性能実装を開発しています。私たちは以下のパフォーマンスの数字を得ています。

	価格	実行時間(ms)	加速
CPU	0.5892381192	7,557.72	1.0倍
GPU	0.5892391192	698.28	10.83倍

98% 信頼区間の幅はオプションプライスの0.05%です。アルゴリズムが単精度で安定して正確に動くことを確認しました。これにより両方のプラットフォームでパフォーマンスが２倍に改善されました。最終的にGPUテクスチャメモリで試してGPU実行時間を153ms（ミリ秒）に短縮しました。これにより倍精度の価格と比べて2.60e-7の誤差の価格が得られます。

1 概要

金融市場では取引が可能な新しいデリバティブやオプションが増加しています。これらのオプションは今まで以上に複雑なため、価格決定の閉形式解がすぐには利用できないことがよくあり，適正価格を得るのに近似に頼らざるをえません。

閉形式解なしで正しいオプションの価格を決定できる一つの方法は，モンテカルロシミュレーションを使用することですが，これは価格決定の手法としては多くの場合好まれません。計算要求が多い（したがって時間がかかる）という事実があるためです。大量の並列コンピュータハードウェアの出現（AVX搭載のマルチコアCPU，GPU，Intel Many Integrated Coreとしても知られるIntel Xeon Phi[1]あるいはMIC ）はモンテカルロ法の新たな関心を引き起こしました。 これらの手法がハードウェアに理想的に適しているためです。多くの金融商品について，モンテカルロ法は金融機関の活動に役立てるのに十分速く正確な価格を提供することができます。

この論文では従来から労力を要する商品，すなわちマルチファクターのローカル・ボラティリティモデルによって決定されるFXバスケットオプションを研究します。

その目的は２つのコンピュータプラットフォーム上でこの商品のパフォーマンスを評価することです：それらは高品質のIntel CPU と NVIDIA GPU です。特に，速度と精度の両方について混

* ニューメリカルアルゴリズムズグループ

† インペリアルカレッジロンドン

[1] Xeon Phi は Intel Many Integrated Core (MIC)技術向けの製品名です。

合精度のプログラミングの影響を研究しています。多くの専門家は倍精度でほとんどの計算を行っており，モンテカルロシミュレーションでさえも倍精度で行っています。私たちの目的は安定性や精度と同様に混合精度のパフォーマンスの利点を数値化することです。

2 バスケットオプションと解析的近似

バスケットオプションはすぐに利用できる閉形式解をもたないオプションの絶好の例です。さらに，利用できる近似法はいくつかの重要な欠点があります。バスケットオプションはペイオフがN個の資産（原資産と呼ばれる）の加重和の値に依存している多次元のデリバティブです。この加重和はバスケットオプションの分布が未知であることを意味します。バスケットオプションの価格の近似を見つけるために論文の中で多くの試みがなされてきました。大まかにこれらは２つのグループに分けることができます:それらは一定のボラティリティを想定するモデルとボラティリティスマイルと調和しようとしているモデルです。最初のグループのモデルは一般的に分析的に扱いやすい分布をもつバスケットオプションの分布を近似しようとします。Levy [11] は原資産価格プロセスが幾何学的ブラウン運動に従っているとき算術平均の分布は対数正規分布によってうまく近似されることを示唆しています。その他の近似にはMilevsky & Posner [12]が提案した逆ガンマ分布あるいはジョンソン分布族の一つが含まれています。これらの近似は多くの場合バスケットオプションの価格決定には適していません。観察されたマーケットスマイルを得て再現するということができないためです。さらにこれらの近似から得られた価格は信頼性がないことが示されています:近似はバスケットの真の分布をより扱いやすい分布へ適合させることに基づいており，従って原資産間の相関効果を得ることができません。相関効果はバスケットオプションを有益にするものであるためこれは重要な欠点です。完全に相関性のある市場ではバスケットの価格決定はN個のコールオプションの価格決定と等しいです。しかしながらその他の場合では資産価値の下落を防いでいるうちにバスケットの価格はN個のコールオプションの合計よりも低くなります。最終的にこれらの近似は厳密に正となる重みづけを必要とします。これはモンテカルロシミュレーションでは問題にはなりません。

二つ目のグループは観察されたマーケットスマイル（smile）を得ようとするモデルから構成されています。これらの例には特に Brigo & Mercurio [2]の対数正規混合モデル，Huynh [8]の Edgeworth 級数展開モデルと Xu & Zheng[13]のローカル・ボラティリティ・ジャンプ拡散モデルが含まれています。これらのモデルから生じる近似の多くは多数の資産を扱う際に役にたちません。これは主にモデルの構築や調整に必要な計算量に起因しています。例えば，対数正規混合モデルは一般に原資産の数の増加とともに急速に（急激に）増加する高い密度を必要とします。従ってモデルの作成は非常に費用がかかります。繰り返しますが，これらの近似は全て重みが正のままである必要があります。

近似の論文とは別に，スマイルを得ようとする資産ダイナミクス（場合によってはスマイルダイナミクス）の急速に成長しているより一般的なモデルがあります。これらのモデルのうち最もよく知られているのはおそらくヘストンモデル（期間構造をもつ），ローカル・ボラティリティモ

デルそして確率ローカル・ボラティリティモデルです。これらは資産動向の一般モデルであり通常は近似式を導き出すには非常に複雑です。オプション価格を得るためにはこれらのモデルは（疑似）モンテカルロシミュレーションかPDE技術かその他の手法（例えば半解析的公式や恒等変換）のどれかによって数値的に処理される必要があります。以前からこれらの数値手法のコストはこのような一般的資産モデルがバスケットオプションの価格決定には使用されてこなかったことを意味していました。パフォーマンスの結果はモンテカルロが現在はこれらのモデルによって決定されるバスケットに対する実行可能な価格決定の手法であることを示唆しています。

3 ローカル・ボラティリティFXバスケットオプション

私たちが重点を置くモデルはローカル・ボラティリティモデルです。これは実際に広く使用される柔軟性のあるモデルで,アービトラージフリー補間法と対になる場合にアービトラージフリーコールオプション価格を生じさせることができます。そのため外国為替市場での慣行と調和し続けるため Dupire [4] の研究に少し修正を加えて進めています。
バスケットコールオプションはN 個の相関資産の加重和から成り立っており、以下の式で与えられる価格となります。

$$\mathrm{C} = e^{-r_d T}\mathbb{E}(B_T - K)^+ \tag{1}$$

ここで r_d は国内のリスクフリー金利で、$K > 0$ は行使価格です。B_T は以下の式で与えられます。

$$B_T = \sum_{i=1}^{N} w^{(i)} S_T^{(i)} \tag{2}$$

ここで $S^{(i)}$は i 番目（$i = 1,\dots,N$）の原資産の値（外国為替レート）を示します。また$w^{(i)}S$ は $\sum_{i=1}^{N} w^{(i)} = 1$ となる一連の重みです。 ローカル・ボラティリティモデルでは各原資産の変化は確立微分方程式（SDE）によって決定されます。

$$\frac{dS_t{}^{(i)}}{S_t{}^{(i)}} = \left(r_d - r_f^{(i)}\right)dt + \sigma^{(i)}\left(S_T^{(i)}, t\right)\mathrm{d}W_t{}^{(i)} \tag{3}$$

ここで$r_f^{(i)}$ は i 番目の通貨ペアの海外のリスクフリー金利です。$(W_t)_{t\geq 0}$ は$\langle W^{(i)}, W^{(j)}\rangle_t = \rho^{(i,j)}t$ となる相関 N次元ブラウン運動で， $\rho^{(i,j)}$ は $i,j = 1,\dots,N$ の場合の相関係数を示します。上記の(3)の関数 $\sigma^{(i)}$ は i 番目の資産のローカル・ボラティリティと呼ばれ，通常未知です。$\sigma^{(i)}$ は時間と現在の資産レベルの両者に依存するため，ローカル・ボラティリティモデルは観測されたマーケットスマイルを得ることができると予想する人がいるかもしれません。そして実際にこれが事実であると示すことができます。Dupire [4]に従って，$C_{BS}(S,K,T,r_f,r_d,\theta)$ で価格 S の原資産の満期 T と行使価格 K のコールオプションのブラック・ショールズ価格を示すと

以下のようになります。

$$C_{BS} = C_{BS}(S,K,T,r_f,r_d,\theta) = Se^{-r_f{}^T}N(d_1) - Ke^{-r_d{}^T}N(d_2) \tag{4}$$

ここでθ はボラティリティを示します。$d_{1,2}$ は以下の式で表されます。

$$d_{1,2} = \frac{\ln(S/K) + \left(r_d - r_f \pm \frac{1}{2}\theta^2\right)T}{\theta\sqrt{T}} \tag{5}$$

そして $N(x)$ は $x \in \mathbb{R}$ で評価される累積正規分布関数を示します。(3)のローカル・ボラティリティ$\sigma(K,T)$ が以下のDupire 式で与えられると示すことができます(上付き文字(i)を省略)。

$$\sigma^2(K,T) = \frac{\theta^2 + 2T\theta\theta_T + 2\left(r_T^d - r_T^f\right)KT\theta\theta_K}{(1 + Kd_1T\theta_K)^2 + K^2T\theta(\theta_{KK} - d_1T\theta_K^2)} \tag{6}$$

(6)の式中のθ はブラック・ショールズインプライド・ボラティリティと呼ばれており，市場で観測可能な任意及び固定の C_{BS}，S，K，T，r_f ，r_d についての(4)の等式を生み出す固有の値として定義されています。インプライド・ボラティリティ θ は K と T （他の変数と同様）の陰関数と見なされます。Dupire 式は$\theta \equiv \theta(K,T) \in C^{2,1}$ を想定します。そのため対応するインプライド・ボラティリティ $\theta(K,T)$ を見つけるために異なる行使価格 Kと満期日（tenors：満期）T について市場で観測されたクォート（quote） C_{BS} を用います。この関数はボラティリティ・サーフェスまたはスマイルとして参照されます。

(6) は非常に多くのインプライド・ボラティリティ・クォートがあることを意味します。しかし市場は異なる行使価格と満期日のわずかなコールオプションのみを取引します。従ってタスクは(6)で使用できるかなり小さな個別クォートから $C^{2,1}$ 関数を作ることです。外国為替市場で業務を行う際にアービトラージフリー手法でこれを行うことは決して容易ではありません。この10年以上これはずっと多くの研究のテーマでした（例えばAndreasen & Huge [1], Kahalé[9], Fengler [6] 及び Glaser & Heider[7] を参照）。市場実務家がとってきた共通の手法はアービトラージフリー補間の細部を無視し単に三次スプラインを使用することです。これは私たちが採用してきた方法で簡単ではありますが,計算という観点からはこの手法はハードウェアプラットフォームがより優れた補間法を用いてどのように実行するかを示すには十分複雑です。

3.1 モンテカルロスキーム

SDE (3) を解くために，私たちは対数過程（log process）でEuler-Maruyama スキームを使用します。バスケットオプションの満期が $n_T\Delta t = T$ となるようにそれぞれの長さΔt のn_T タイムステップ（時間刻み幅）を使用します。

$t_\tau = \tau\Delta t$ （ $\tau = 1,\dots,n_T$ ）では，(3) の離散化バージョンは以下になります。

$$\log\left(\frac{S_{t_{\tau+1}}^{(i)}}{S_0^{(i)}}\right) - \log\left(\frac{S_{t_\tau}^{(i)}}{S_0^{(i)}}\right) = \left(r_d - r_f^{(i)} - \frac{1}{2}\sigma^{(i)}\left(S_{t_\tau}^{(i)}, t_\tau\right)^2\right)\Delta t + \sigma^{(i)}\left(S_{t_\tau}^{(i)}, t_\tau\right)\sqrt{\Delta t}d_iZ_r \tag{7}$$

ここでZ_r は標準の正規乱数の N ×1 列ベクトルで，d_i は$N \times N$ の相関行列のコレスキー分解の i 番目の行です。言い換えれば $d_i d_j{}^T = \rho^{(i,j)}$（$i,j = 1,\dots,N$）です。
ローカル・ボラティリティ $\sigma^{(i)}\left(S_{t_\tau}^{(i)}, t_\tau\right)$ は点 $\left(S_{t_\tau}^{(i)}, t_\tau\right)$（$\tau = 1,\dots,n_T$）のインプライド・ボラティリティ（及びデリバティブ）を必要とする（6）によって計算されます。この計算は各タイムステップでサンプルパスごとに実行される必要があります。そして計算の大部分を占めます。

3.2 インプライド・ボラティリティ・サーフェスの構築

完全を期すために，私たちはどのようにインプライド・ボラティリティ・サーフェスを構築するかを簡単に説明します。 あるいはより正確にサーフェスを表すスプラインを補間する方法について説明します。これはモンテカルロシミュレーションの前に行われる設定プロセスの一部で，これらの計算は実行時間には反映されません。
前述のとおり，ローカル・ボラティリティ関数$\sigma^{(i)}$ は市場で観察されたインプライド・ボラティリティ･サーフェスから計算される必要があります。通常外国為替市場は満期ごとに５つの｢デルタ」についてクォートを与えます。クォートは行使価格ではなく「デルタ」で与えられます。外国為替市場には独自の慣習とプライスクォート手法があります:例えば４つの異なるデルタタイプがあります。どのデルタを使用するかを決定する方法とデルタから行使価格への変換方法についてのさらなる詳細についてはClark [3]を参照下さい。
デルタから行使価格へ市場クォートを変換すると滑らかなインプライド･ボラティリティ･サーフェスを構築するのにそれらを使用することができます。このため私たちは三次スプラインを使用します。市場インプライド･ボラティリティ･サーフェスは満期の集まりから成り立っていて，満期に５つの異なる行使価格で与えられる５つの異なるインプライド･ボラティリティ・クォートがあることを思い出してください。最初のステップは各満期で５つのクォートを通して行使価格の方向に三次スプラインをフィットさせます。
各満期は次のように処理されます。特定の満期を決めます。５つの時間 0 から時間 T まで区分的単調エルミート多項式をフィットさせます。各多項式はこの満期でクォートの一つを通ります。各満期のクォートは異なる行使価格なので，前もってフィットされた三次スプラインで値を補間する必要があります。 今回の満期と以前の満期との間の全てのモンテカルロタイムステップrΔt に関してインプライド･ボラティリティ値を５つの行使価格それぞれで補間するために私たちはエルミート多項式を使用します。
最後にこれらの rΔt のそれぞれについて三次スプラインは計算された５つのインプライド・ボラティリティ値を通して行使価格方向にフィットされます。これらのスプラインの終点の勾配は推定され，インプライド・ボラティリティの左側及び右側（行使価格方向の）のテイルを含めるようインプライド・ボラティリティサーフェスの線形外挿関数を形成します。
各満期についてこのプロシージャを繰り返すことにより行使価格方向に三次スプラインの列ができます。各モンテカルロタイムステップ rΔt（r $= 1,\dots,n_T$）ごとに一つが生成され、これは(6)の θ，θ_K, θ_{KK} の計算に使用されます。基本的に同じプロシージャが θ_T の計算に使用されます。

正確なスプラインデリバティブが計算されますが有限差分近似は使用されませんのでご注意ください。どちらかというと暫定的な補間技術と外挿技術と併せて実際の市場データ（アービトラージフリーは保証されていない)を使用するのでモデルのアービトラージの存在を示す負のローカル・ボラティリティが生じます。これはアービトラージフリー補間法，例えばFengler [6]を用いて修正することができます。

4 モデルの実装

主要な計算負荷は各タイムステップのそれぞれのサンプルパスについての Dupire式(6)の計算から成りたっています。このために私たちはいくつかのデリバティブと同様にインプライド・ボラティリティ θ を必要とします。それらは２組の３次スプライン（１つはθ，θ_K,θ_{KK} ，もう一つはθ_T）から計算されます。そのためモンテカルロの各タイムステップはN個の原資産それぞれについて２つの３次スプラインと関係しています。

モデルの実装の見地から,外国為替市場と株式市場との主な違いは前者の市場クォートが相対的に少ないという点です:前述のとおり，流動的な株式商品は満期ごとに15以上のクォートがあるのに対して外国為替市場は満期ごとに５つのクォートしかありません。これは株式商品に使用されるスプラインが外国為替商品で使用されるスプラインよりもはるかに大きいことを意味します。例えば外国為替モデルでは一つの原資産に対する全てのスプラインデータは約100KB記憶域しか必要としません。スプラインの多くはL1キャッシュ（あるいはGPUの共有メモリ）にフィットできます。株式モデルは逆にキャッシュに影響するより多くの記憶域を必要とします

１つの実装のオプションは,次のサンプルパスに進む前に全てのタイムステップを通してサンプルパスを処理することです。なぜなら全てのスプラインが基本的にキャッシュにフィットするためです。しかしながらこの手法は株式資産を扱う際には（特にGPU上では）機能しません。このため特定のモンテカルロタイムステップのスプラインをキャッシュに入れることを選択し全てのサンプルパスを次のタイムステップへ進めることを選びます。これによりメインメモリへの多くのトラフィックの行き来が生じます。しかし多くの浮動小数点演算があるためこのトラフィックを効果的に秘匿する必要があります。

(7)からEuler-Maruyamaタイムステップを行っているとき各資産を個別に処理できることにご注意ください。全てのN個の資産が同時に必要となるのは利益(2)が計算されるときだけです。

4.1 GPU 実装

GPU コードはCUDA で書かれています。正規乱数はGPU向けNAG数値計算ルーチンの MRG32k3a 乱数生成器を使用してGPUメモリで生成されています。これはデバッグを非常に軽減するので順序付けは生成器の（L'Ecuyer [5]参照）標準逐次アルゴリズム[2] の順序と一致させるよう選択さ

[2] NAG GPU 生成器は元の順番を維持できます。あるいはピークパフォーマンス用に順序を変更された数を返します。変更された順番は速いです。詳細についてはGPUドキュメントに関するNAG数値計算ルーチンを参照下さい。

れています。計算の剰余は２つのカーネルに分かれます。一つは全てのサンプルパスを時間 0 から時間 T へ進めます。もう一つはペイオフを計算します。すべての３次スプラインと入力データはGPUメモリにコピーされます。サンプルパスのカーネルでは各スレッドブロックは一つの資産のみ処理します（しかし資産ごとに複数のスレッドブロックがあります）。カーネルの開始では相関行列（上記の(7)参照）のコレスキー分解の対応する行d_i が共有メモリ[3] に取り込まれます。各タイムステップで，対応する３次スプラインは共有メモリに取り込まれ，全てのサンプルパスを次のタイムステップに進めるのに使用されます。

4.2 CPU 実装

CPU上でできるだけ多くAdvanced Vector Extension (AVX)ベクトルユニットを使用したことを確認するため，CPU コードはC言語で書かれておりかなりの手間がかけられています。私たちは完全にベクトル化され，完全に並列化された，標準逐次 MRG32k3a アルゴリズムと同じ順序で数値を返す MRG32k3a の実装を行いました。パス計算について私たちは２つの手法を試しました。一つ目は基本的にGPU実装と同じです：全てのサンプルパスは一つのタイムステップから次のタイムステップへ進みます。基本的な関数の組み合わせ(Intel Compiler Documentation[10]を参照)と手作業のループアンローリングを通して全てのサンプルパスを通るループをベクトル化するのは可能です。そのためカーネルには全てのタイムステップの外側のループがあり，サンプルパスを通る完全にベクトル化され並列化された内側のループがあります。
二つ目の手法は２つのループを交換します：外側のループは全てのサンプルパスを通り，内側のループは全てのタイムステップを通ります。この発想は外側のループをベクトル化することです。いくつかの異なるオプション[4] を試しましたが コンパイラにこれをさせることはできませんでした。結局，最初の手法がより速いです。
手作業のループアンローリングのためCPUアルゴリズムは10個の原資産（研究用に選択した問題サイズ）をもつバスケットに対してハードコードされており，そのため汎用コードではありません。GPUコードはどんな数の原資産にも対処します。さらに，各ベクトルユニットが一度に４つの倍精度データを処理できますがサンプルパスループのベクトル化は倍精度の実行時間を43%削減しました。

[3] 私たちは正規乱数を相関するため CuBLAS (dtrmm) を用いて調査しましたがサンプルパスカーネルの内側では乱数の相関よりも遅いことがわかりました。
[4] 基本的関数，様々なループの手動でのアンローリング，if文の再構築と様々なコンパイラプログラム

5 結果

私たちは純粋な倍精度コードから始め,徐々に様々な部分の単精度の計算と入れ替えて精度と速度における影響をみてきました。最新のプロセッサは単精度の計算を倍精度の計算よりも少なくとも２倍速く処理します。実際に計算カードのNVIDIAのケプラー級数では倍精度のパフォーマンスよりも３倍高い単精度のパフォーマンスが得られます。多くの金融実務家は全ての計算を倍精度でおこなっていることを認めていますが､何故なのかはっきりとした認識をもっていないようです。これはモンテカルロコードでさえも事実です。モンテカルロ積分は本質的にランダムプロシージャであり答えに関する推定値や信頼区間を与えるにすぎないことを考えると,倍精度の余分な精度は何か意味があるというのは疑わしいようです。実務家の間で懸念されていることは数値的安定性についてと混合精度アルゴリズムが労力を正当化するだけの十分な加速を提供するかどうかということです。これらは私たちが調査し始めている課題です。

5.1 テストプログラム

私たちは10個の通貨（等しく重みづけされたEURUSD, AUDUSD, GBPUSD, USDCAD, USDJPY, USDBRL, USDINR, USDMYR, USDRUB 及び USDZAR）で１年のバスケットオプションを検討しました。
スポットレートは2012年９月時点の市場データから得られ, 相関行列が推定されました。海外と国内両方の金利と行使価格は簡単にするためにゼロとしました。
モンテカルロシミュレーションでは360タイムステップをもつ150,000のサンプルパスを使用しました。シミュレーションは倍精度で4.12GBとなる5億4000万個の乱数を必要とします。

5.2 ハードウェアとコンパイラ

以下は, 私たちが使用した様々な計算プラットフォームです：

1. CPU: Intel Xeon E5-2670, 2.6GHz の基底周波数で稼働し計算負荷時に3.3GHzまで増加可能な20MB L3 キャッシュ搭載の8コアプロセッサ。本プロセッサはAVX命令をサポート。256ビットベクトルユニット（4つの倍精度データ）。130GB メモリ。ハイパースレッディングは無効。Intel compiler icc version 12.1.0 を以下のフラグを指定して使用。

```
-O3 -xHost -openmp -g -restrict
```

2. GPU: 6GB RAM 搭載のNVIDIA K20Xm。GPU クロックレート700MHz。ECC はオフ。メモリ幅約200GB/s。gcc version 4.4.6 を使用。また nvcc version 5.0.0.2.1221 のCUDA toolkit 5.0 を以下のフラグを指定して使用。

```
-O2 -Xptxas -v --generate-code arch=compute_35,code=sm_35
```

備考： GPU のタイミングには全てのオーバーヘッドとメモリ転送が含まれます。[5]

5.3 ベース倍精度の結果とスケーリングの数字

表１は両方のプラットフォームの倍精度の結果を示しています。信頼区間は価格の約0.05%です。

	価格	信頼区間の幅	実行時間(ms)	加速	乱数生成実行時間(ms)
CPU	0.5892381192	3.195e-4	7,557.72	1.0倍	1,144.85
GPU	0.5892391192	3.195e-4	698.28	10.83倍	125.14

表１： 倍精度の結果（価格, 98%信頼区間の幅, 全体の実行時間, CPUに関する加速, 乱数生成の実行時間）

最後に, 表２でCPU のスケーリングの数字を示します。CPU実行時間はスレッドの数が倍になると半分になり, 非常に強いスケーリングを示しています。

CPU スレッド	1	2	4	8
実行時間(ms)	57,354.13	30,319.64	14,953.17	7,557.72

表２：CPUのスケーリングの数字

5.4 混合精度

単精度で実行した計算（例えば IEEE 32bit 浮動小数点を使用）は約1e-7の精度があります。特に,単精度と倍精度で同じ計算を実行した場合２つの結果には約1e-7の相対誤差が予想されます。もし誤差が著しくこの数字よりも悪い場合は単精度では計算が不安定であることを示唆しているか,もしくは倍精度が軽減に役立つ丸め誤差がかなり積み重なっていることを示唆しています。混合精度プログラミングの影響を分析するため,NVIDIAからの新しいケプラー級数では倍精度より単精度のパフォーマンスが３倍高いため私たちは最初にGPUコードに注目しています。精度とパフォーマンスへの影響をみるため私たちは次々にアルゴリズムの様々な部分を倍精度から単精度へ変更しました。その結果は表３で示されています。以下がその変更です：

Dp　元の倍精度のコード。

1. 単精度の正規乱数を使用します。数字はパス計算カーネルに読み込まれるとすぐに倍精度に変換されます。そのため加速は速い乱数生成器のみに起因します。相対誤差は単精度アルゴリズムからの予想値と同等です。誤差はステップ Sp や 4 の総合誤差とそれほど変わりません。

[5] CUDA 実行時間の初期化とコンテキスト作成のコストは除外されています。

ステップ	価格	増分の相対誤差	全体の相対誤差	実行時間(ms)	加速
Dp	0.5892391192		0.0	698.21	1.00倍
1	0.5892389949	2.11e-7	2.11e-7	637.60	1.10倍
2	0.5892389969	3.39e-9	2.08e-7	425.41	1.64倍
3	0.5892389787	3.09e-8	2.38e-7	402.70	1.73倍
Sp	0.5892388787	1.70e-7	4.08e-7	347.32	2.01倍
4	0.5892392722	6.68e-7	2.60e-7	153.82	4.54倍

表３：GPU コードの混合精度の変更（各ステップと前のステップと比較した相対誤差，各ステップと倍精度コードと比較した相対誤差，実行時間，倍精度と比較した加速）

2. 前述のステップに加えて，単精度でインプライド・ボラティリティ・スプラインの計算を行います。スプラインの計算は計算の大部分を占めており，これらを単精度に切り替えることにより大幅に実行時間を短縮しています。
3. 前述のステップに加えて，単精度で Dupire式 (6)を計算します。おそらく実行時間は以外にもそれほど削減しません。

Sp シミュレーション全体（乱数，パス計算カーネル，ペイオフカーネル）は単精度で行われます。

4. 各資産のローカル・ボラティリティ・サーフェスは361時間点と400空間点の格子上で事前に計算されます。これらの事前に計算されたサーフェスはパス計算カーネルで双線形補間をもつレイヤテクスチャを用いて抽出されます。

計算全体を倍精度から単精度へ移行することにより2.01倍のかなりの加速がもたらされますが，理論上の3倍の加速よりはかなり少ないです。なぜ達成された数字が理論上の数字とかなり異なるかは不明です。

精度に関して，単精度のコードには4e-7の相対誤差があります（倍精度の答えと比較して）。これはアルゴリズムが単精度で安定していることと大幅な累積の丸め誤差がないことを示唆しています。

CPU上で単精度での計算全体を実行した結果は表４に示されています。CPUコードに対して2倍以上の加速が得られたのは様々なレベルのキャッシュでのデータフィッティングに起因しています。

	価格	相対誤差	実行時間(ms)	加速	乱数生成 実行時間(ms)
CPU (sp)	0.5892389162	3.45e-7	3,312.53	2.28倍	957.92

表４：CPUの完全な単精度コードの結果（価格，倍精度の結果と比較した相対誤差、倍精度の結果、全体の実行時間、倍精度と比較した実行時間の加速、乱数生成器のみの実行時間）

単精度のオプション価格は値が追加される順番が異なるため２つのプラットフォームでは異なります。これは異なる並列アーキテクチャの単精度のコードでも予想されます。

5.5 GPU テクスチャメモリ

表３の最終ステップのパフォーマンスの数字を考慮し，GPUテクスチャメモリを用いた実験を簡単に説明しましょう。テクスチャはGPUメモリ内の１次、２次あるいは３次の配列で，ハードウェアテクスチャ参照ユニットからアクセスされます。テクスチャは通常コンピュータゲームなどのグラフィックスアプリケーションで使用されますが，それらはローカル・ボラティリティモデルにも非常によく適合しています。ハードウェアテクスチャユニットによりGPUプログラムはランダムポイントでテクスチャを抽出することができます:テクスチャユニットは入力座標を取り込み，テクスチャの座標に変換し，テクスチャから座標に最も近い要素を見つけ，そして入力座標でテクスチャの値を近似するために最も近い要素の間で線形補間を実行します。たとえばもしテクスチャとして２次元のローカル・ボラティリティ・サーフェスを設定する場合，GPUプログラムは株価パス上の各点でローカル・ボラティリティ・サーフェスを抽出することができ，現在の株価でローカル･ボラティリティを近似するためにテクスチャユニットは４つの最も近いローカル・ボラティリティ要素間の双線形補間を実行します。

テクスチャは金融工学では使用が限定されています。実務家は通常２つの懸念を挙げます：一つ目はテクスチャが単精度データでしか作用しないことと，２つめはハードウェアユニットにより実行される線形補間が低い精度であることです。NVIDIA GPUのテクスチャユニットは（変換された）入力座標の端数部分を表すために8ビットの固定精度を使用しています[6]。そのため端数部分は $2^{-8} \approx 0.004$ の解をもちます。10.001と10.003 の２つの（変換された）入力座標は両者とも同じ値に補間されます。

テクスチャメモリの予想される効果を調べるため,私たちは361時間点と400空間点をもつ座標上で各資産についてローカル・ボラティリティ・サーフェスを計算しました。これらの計算は倍精度で行われ，計算結果は単精度に変換されています。これらの値はレイヤテクスチャ[7] にコピーされ,各資産のボラティリティ･サーフェスはハードウェアテクスチャを用いて抽出されています。この結果表３の最終ステップですばらしい加速が報告されています。この手法の精度は完全な単精度演算と一致しています。

空間と時間の両方で座標のサイズを変えましたが､実行時間について何も変化が観察されませんでした。座標のサイズを小さくすると精度が悪くなるだけでした。一方座標のサイズを361x400よりも大きくしても精度は改善しないようでした。どのくらいの大きさの座標が十分なのかは問題に依存しますが,一般にテクスチャを使用したい実務家は小さい座標よりも大きい座標を選ぶ必要があります。

---

[6] わずか 1 を表すことができる 9 番目の bit があります。

[7] レイヤテクスチャとは簡単に言えばテクスチャの配列です。

6 謝辞

PSG クラスタへのアクセスを提供いただいたNVIDIA社 に感謝致します。

参考文献

[1] ANDREASEN, J., HUGE, B. (2011) Volatility interpolation. Risk Magazine, March 76-79.

[2] BRIGO, D., MERCURIO, F. (2000) Displaced and Mixture Di_usions for Analytically-Tractable Smile Models. Mathematical Finance-Bachelier Congress 2000, Springer : Berlin.

[3] CLARK, I. (2011) Foreign Exchange Option Pricing. John Wiley & Sons Ltd , Chichester.

[4] DUPIRE, B. (1994) Pricing with a Smile. Risk , 7 (1) 18-20.

[5] L'ÉCUYER, P. (1999) Good parameters and implementations for combined multiple recursive random number generators. Operations Research, 47 (1)159-164.

[6] FENGLER, M. (2009) Arbitrage free smoothing of the implied volatility surface. Quan-titative Finance, 9 (4) 417-428.

[7] GLASER, J., HEIDER, P. (2012) Arbitrage free approximation of call price surfaces and input data risk. Quantitative Finance, 12 (1) 61-73.

[8] HUYNH, C. (1994) Back to Baskets. Risk , 7 (5), 59-61.

[9] KAHALÉ, N. (2004) An arbitrage free interpolation of volatilities. Risk, 17 102-106.

[10] INTEL CORPORATION. Intel C++ Compiler XE 13.1 User and Reference Guide. Available online on the Intel website.

[11] LEVY, E. (1992). Pricing European Average Rate Currency Options. Journal of International Money and Finance, 11 (5), 474-491.

[12] MILEVSKY, M., POSNER, S. (1998). A Closed-form Approximation for Valuing Basket Options. The Journal of Derivatives, Summer 1998, 54-61.

[13] XU, G., ZHENG, H. (2010) Basket Options Valuation for a Local Volatility Jump-Diffusion Model with Asymptotic Expansion Method. Insurance : Mathematics and Economics , 47, 415-422.