Quantras

Heavy Parallelization of Alternating Direction Schemes in Multi-Factor Option Valuation Models

Cris Doloc, Ph.D.

Quantras

INTRO

WHO

- Ex-physicist – Ph.D. in Computational Physics - Applied TN Plasma (10 yrs)
- Working for the last 15 years in HFT / MM

WHAT

- Working on Option valuation for proprietary trading
  - ➢ big consumer of cutting-edge tech: Big Data + Fast Compute
- Exploring new products: multi-asset derivatives
- Presentation last year on Lattice models → exploring "new" techniques for M-Fact

WHY

Goals:

- Understand the nature of the "beast", i.e. the **Problem**
- Explore "new" methods to solve the problem ← Physics & Engineering: ADI/E

Quantras

OUTLINE

1. Understanding the PROBLEM:
   - ❑ *well-posed(?)*
   - ❑ *current methods*
   - ❑ *limitations*
   - ❑ *needs*
2. Alternating Direction Schemes:
   - ❑ *Intro*
   - ❑ *classification*
   - ❑ *comparison*
3. CUDA implementation of ADI: *goal → heavy parallelization*
4. Practical applications of ADI: *Volatility models, Swaps, TF, APOs*
5. Conclusions

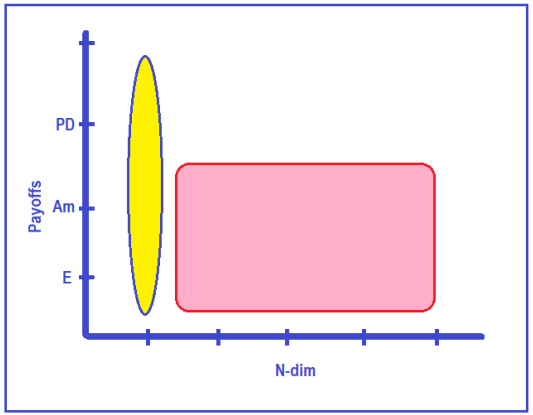
Quantras

1. Understanding the PROBLEM

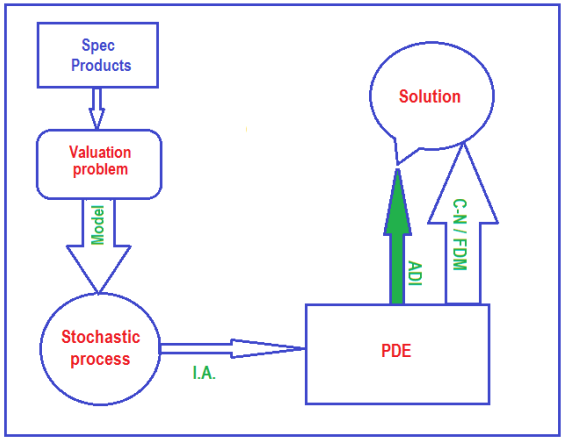
A. Statement & Context:

Problem: pricing complex financial derivatives that have more than one underlying asset and different types of payoff functions. 2 dimensions: # underlying assets+ payoff form

Payoffs: - single / double barrier
- time windows barrier
- exercise style: European / American / Bermudan



Model:



Quantras

Examples

a. Black-Scholes: valuing a single asset European-style option

$$dS_t = rS_t dt + \sigma S_t dW_t$$

$$\frac{\partial V}{\partial t^\star} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - D)S\frac{\partial V}{\partial S} - rV = 0$$

Solution - analytic

b. **Heston**: stochastic volatility model

$$\begin{cases} dS_t & = rS_t dt + S_t\sqrt{V_t}dW_t^1, \\ dV_t & = \kappa(\eta - V_t)dt + \sigma\sqrt{V_t}dW_t^2 \\ dW_t^1 dW_t^2 & = \rho dt, \end{cases}$$

$$\frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + \sigma sv\rho\frac{\partial^2 u}{\partial s\partial v} + \frac{1}{2}vs^2\frac{\partial^2 u}{\partial s^2} + \kappa(\eta - v)\frac{\partial u}{\partial v} + rs\frac{\partial u}{\partial s} + \frac{\partial u}{\partial t} - ru = 0.$$

Quantras

Numerical Methods to solve the problem

[1] Monte-Carlo simulations:

- most popular
- embarrassingly parallel
- computationally expensive
- very hard to adapt to more "exotic" payoffs styles

[2] Numerical integration techniques:

- involve a transform to the Fourier domain
- expectation is calculated as an integral of the probability density function
- analytical forms are commonly used for calibration of path-independent options

[3] **Finite Difference Models**:

- commonly used for early exercise options
- more accurate and generic method than MC
- "***curse of dimensionality***" for higher dimensions

Quantras

CURSE OF DIMENSIONALITY

• When using a PDE approach, each stochastic factor → spatial variable in the PDE. There is a "***curse of dimensionality***" associated with high-dimensional PDEs, therefore pricing of such derivatives as PDEs is very challenging.

• When stochastic processes in the pricing model are correlated (very common) the resulting PDE, possesses ***mixed spatial derivatives*** which makes solving the associate problems numerically even more challenging.

Quantras

B. The use of FDM to price complex derivatives:

• The FDM is used to approximate the solution of the PDE that describes the time evolution of a derivative that is contingent on one or more underlying variables. The theory of FDM for one or two space variables is reasonably well developed, but it has some serious limitations (Duffy)

• The generic one-factor PDE has the form:

$$\frac{\partial u}{\partial t} = a(x,t)\frac{\partial^2 u}{\partial x^2} + b(x,t)\frac{\partial u}{\partial x} + c(x,t)u + f(x,t), \quad 0 < x < \infty, \quad 0 < t < T$$

• In order to completely specify the problem one needs to define (a) initial and (b) boundary conditions:

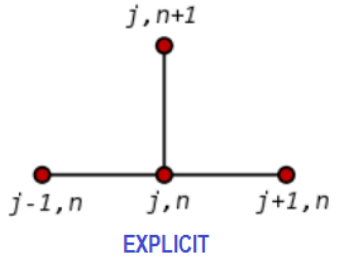
$$u(x,0) = f(x), \quad 0 < x < \infty$$

• The PDE is a generic form of the **time-dependent**, inhomogeneous **convection-diffusion-reaction** eq.

Quantras

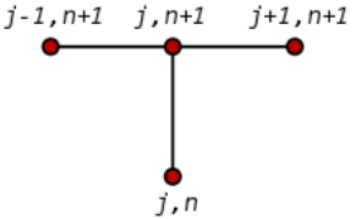
Discretization schemes:

- Explicit: ***Cond. Stable***
  - FWD diff in time
  - Central diff in state



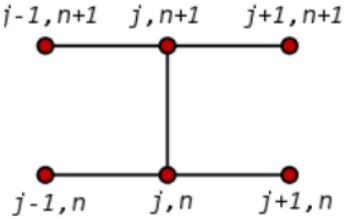
EXPLICIT

- Implicit : ***Always Stable***
  - BKW diff in time
  - Central diff in state



IMPLICIT

- Crank-Nicholson : ***Always Stable***
  - Central diff in time
  - Central diff in state



CRANK-NICHOLSON

Quantras

Importance of Boundary conditions:

• Usually the PDE is defined on a semi-infinite interval.

• In order for the problem to be **well-posed** in the Hadamard sense (solutions exist, are unique and depend continuously on the initial conditions) it requires some change of variable to normalize it to (0, 1) – ***Fichera*** theory

• This PDE models a variety of problems from fluid dynamics, to computational finance. The diffusion and convections terms determine whether boundary conditions need to be specified, and if so, where on the boundary they should be imposed.

• An example of numerical issue is the "**boundary layer behavior**": the diffusion term is small and the convection is large. In this scenario the FDM techniques will produce oscillating / spurious solutions – static instability. Example: small volatility and high interest rates….

Quantras

C. Limitations of the classic FDM approach:

• The standard von Neumann stability analysis fails to predict the infamous ***spurious oscillation*** problem. Hedging applications that use TVs calculated via FDM will run the risk of being inaccurate at values in the payoff function where this is not smooth (strike, expiration).

• The ***second-order accuracy*** is not preserved when using non uniform meshes. And sometimes uniform, meshes are not sufficient to approximate correctly the solution in a boundary layer with complex payoffs. To improve this stability issues one uses extrapolation (last year presentation…)

• The PDEs in computational finance tend to be defined in a semi-infinite interval in space. But of course, from a numerical perspective the FDM must be defined on a finite interval and to this end most methods ***truncate*** the semi-infinite interval. Fichera theory allows for space variable changes that will normalize the PDE to the unit interval (0, 1).

$$y = \frac{x}{x + \alpha}$$

• Implicit schemes are stable but we ***must solve a matrix system*** at each time level to arrive at a solution. This could be very expensive computationally. The multi-dimensional PDEs are almost intractable from a numerical point of view, b/c the "***curse of dimensionality***".

Quantras

D. Need for Stability – Accuracy – Fast computation:

• What are the necessary and sufficient conditions for a given FD scheme to be a "good" approximation for a PDE?

•Three criteria must be satisfied:

- Unconditional stability
- Convergence
- Fast computation

• **Stability** - requires that the FD solution does not exponentially grow per time step.

• **Convergence** - the solution to which it converges actually is the desired solution.

• **Fast compute capabilities** – use of GPUs

Quantras

2. Alternating Direction Schemes

• The **Alternating Direction Implicit (ADI)** method is a FD method for solving PDEs.

• It is most notably used to solve the *diffusion equation* in two or more dimensions. It is an example of an *operator splitting* method.

• ADI is a mixture of an implicit explicit scheme, and it can be seen as “the best of both worlds" where it has the stability characteristics from the implicit scheme, but also limits the computation time of matrix inversion. This is done by splitting the matrix per direction.

• **Operator Splitting Schemes**
A technique used to decompose unwieldy systems of PDEs into simpler sub-problems and treat them individually using specialized numerical algorithms. The main idea is to take advantage of decomposition properties in the structure of the problem to be solved.

Quantras

A. How it works:

$$\frac{\partial u}{\partial \tau} = \mathcal{L}u(x_1, x_2, x_3, \tau) \equiv \sum_{i,j=1}^{3} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{3} c_i \frac{\partial u}{\partial x_i} + c_0 u \qquad u(x_1, x_2, x_3, 0) = u_0(x_1, x_2, x_3)$$

- The ADI scheme splits a time step into sub-steps.
- ADI starts by doing one fully explicit step for the portion of the difference operator that corresponds to the mixed derivative matrix.
- Then it does a correction sub-step such that one direction is done implicitly and the rest still explicitly. In the next sub-step the next direction is corrected implicitly. The process is repeated until each sub-step was treated implicitly once.
- In every sub-step only one direction matrix needs to be inverted, which usually has small bandwidth. Therefore, the implicit steps are done fast because these matrices are sparse and the non zero entries are close to the diagonal.
- ADI is a ***predictor-corrector*** scheme → the **intermediate** steps are often referred to as **correction** steps.
- One could treat every direction implicitly, but to enhance computational time it is better to choose only directions that result in sparse matrices. The mixed derivative matrix, is not sparse and is therefore could not treated in an implicit fashion.

Quantras

B. Classification of ADI schemes:

• **Douglas**

$$\begin{cases} Y_0 = U_{n-1} + \Delta t\, F(t_{n-1}, U_{n-1}), \\ Y_j = Y_{j-1} + \theta\, \Delta t\, (F_j(t_n, Y_j) - F_j(t_{n-1}, U_{n-1})), \quad j = 1, 2, \ldots, k \\ U_n = Y_k. \end{cases}$$

• The parameter $\theta > 0$ defines the scheme.

• In the ADI scheme the forward Euler predictor(explicit) step is followed by k implicit but unidirectional corrector steps, whose purpose is to stabilize the predictor step.

• Special instances of well-known ADI schemes:

- **Douglas-Rachford**, with $F_0 = 0$ and $\theta = 1$
- **Brian-Douglas**, with $F_0 = 0$ and $\theta = 1/2$.

Quantras

Craig-Sneyd

- If $F_0 = 0$ → Douglas ADI
- Introduce a correction with respect to the F_0 part as well
- O(2) can be attained independently of F_0—upon taking $\theta = \sigma = 1/2$.

$$
\begin{cases}
Y_0 = U_{n-1} + \Delta t\, F(t_{n-1}, U_{n-1}), \\
Y_j = Y_{j-1} + \theta \Delta t\, (F_j(t_n, Y_j) - F_j(t_{n-1}, U_{n-1})), \quad j = 1, 2, \ldots, k \\
\widetilde{Y}_0 = Y_0 + \sigma \Delta t\, (F_0(t_n, Y_k) - F_0(t_{n-1}, U_{n-1})), \\
\widetilde{Y}_j = \widetilde{Y}_{j-1} + \theta \Delta t\, (F_j(t_n, \widetilde{Y}_j) - F_j(t_{n-1}, U_{n-1})), \quad j = 1, 2, \ldots, k \\
U_n = \widetilde{Y}_k.
\end{cases}
$$

Hundsdorfer-Verwer

- best results for ***accuracy*** and ***order of convergence***
- uses the full right-hand side function F for the update Y_0, instead of just F_0 alone.

$$
\begin{cases}
Y_0 = U_{n-1} + \Delta t\, F(t_{n-1}, U_{n-1}), \\
Y_j = Y_{j-1} + \theta \Delta t\, (F_j(t_n, Y_j) - F_j(t_{n-1}, U_{n-1})), \quad j = 1, 2, \ldots, k \\
\widetilde{Y}_0 = Y_0 + \sigma \Delta t\, (F(t_n, Y_k) - F(t_{n-1}, U_{n-1})), \\
\widetilde{Y}_j = \widetilde{Y}_{j-1} + \theta \Delta t\, (F_j(t_n, \widetilde{Y}_j) - F_j(t_n, Y_k)), \quad j = 1, 2, \ldots, k \\
U_n = \widetilde{Y}_k.
\end{cases}
$$

Quantras

C. Methods comparison

• When combined with second-order central finite differences for the discretization of the space variables, these 2 schemes are ***unconditionally stable*** and ***efficient second-order*** methods in both space and time when applied to PDEs with ***mixed derivative terms***.

• A disadvantage of the Craig and Sneyd scheme is that it cannot maintain both unconditional stability and second-order accuracy when the number of spatial dimensions is greater than three.

• In addition, the Craig and Sneyd scheme may exhibit undesirable convergence behavior when the payoff functions are non-smooth, which is quite common for financial applications. Smoothing techniques may be required.

• The **Hundsdorfer-Verwer** method is unconditionally stable for arbitrary spatial dimensions and at the same time effectively damps the errors introduced by non-smooth payoff functions.

Quantras

3. CUDA implementation of ADI Schemes

"*The **ADI** methods offer sufficient parallelism for an effective GPU implementation of a single pricing problem*" Daniel Egloff (2011)

Extremely valuable work was published in the last 5 years :

- D.M. Dang, et al., *A parallel implementation on GPU of ADI finite difference methods for parabolic PDEs with application in finance,* Canadian Applied Mathematics Quarterly, 2011.

- Z. Wei, et al., *Parallelizing Alternating Direction Implicit Solver on GPUs*, Procedia Computer Science **18** (2013) 389 – 398.

- K. J. IN'T HOUT AND S. FOULON, *ADI finite difference schemes for option pricing in the Heston model with correlation*, International Journal Of Numerical Analysis And Modeling, **7** (2010), pp. 303–320.

Goal: Use of GPUs to solve time-dependent parabolic PDEs in three spatial dimensions with mixed spatial derivatives via a parallelization of the ADI scheme.

• The goal is not to parallelize across time, but rather to focus on the parallelism within one time-step, via a parallelization of the H-V scheme.

• The main components of a GPU-based parallelization of the ADI scheme are:
  • a parallel implementation of the explicit Euler predictor step,
  • a parallel solver for the independent tri-diagonal systems arising in the three implicit, but unidirectional, corrector steps.

• Applying a GPU-based parallel methods to price early exercise options written on three assets.

Quantras

Dang's implementation: ***19 point stencil***

PDE: $$\frac{\partial u}{\partial \tau} = \mathcal{L}u(x_1, x_2, x_3, \tau) \equiv \sum_{i,j=1}^{3} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{3} c_i \frac{\partial u}{\partial x_i} + c_0 u \qquad u(x_1, x_2, x_3, 0) = u_0(x_1, x_2, x_3)$$

Second order FD approximations to the 1st and 2nd partial derivatives of the ***space*** variables in are obtained by *central* schemes, while the cross-derivatives are approximated by a four-point FD stencil:

$$\frac{\partial u}{\partial x_1} \approx \frac{u^m_{i+1,j,k} - u^m_{i-1,j,k}}{2\Delta x_1} \qquad \frac{\partial^2 u}{\partial x_1^2} \approx \frac{u^m_{i+1,j,k} - 2u^m_{i,j,k} + u^m_{i-1,j,k}}{(\Delta x_1)^2}$$

$$\frac{\partial^2 u}{\partial x_1 \partial x_2} \approx \frac{u^m_{i+1,j+1,k} + u^m_{i-1,j-1,k} - u^m_{i-1,j+1,k} - u^m_{i+1,j-1,k}}{4\Delta x_1 \Delta x_2}$$

For the **boundary** points one uses one-sided FD approximations to derivatives:

$$\left.\frac{\partial u}{\partial x_1}\right|_{L_{x_1}} \approx \frac{u^m_{1,j,k} - u^m_{0,j,k}}{\Delta x_1} \qquad \left.\frac{\partial u}{\partial x_1}\right|_{U_{x_1}} \approx \frac{u^m_{n-1,j,k} - u^m_{n-2,j,k}}{\Delta x_1}$$

$$\left.\frac{\partial^2 u}{\partial x_1 \partial x_2}\right|_{L_{x_1}} \approx \frac{u^m_{1,j+1,k} + u^m_{0,j-1,k} - u^m_{0,j+1,k} - u^m_{1,j-1,k}}{2\Delta x_1 \Delta x_2},$$

$$\left.\frac{\partial^2 u}{\partial x_1 \partial x_2}\right|_{U_{x_1}} \approx \frac{u^m_{n-1,j+1,k} + u^m_{n-2,j-1,k} - u^m_{n-2,j+1,k} - u^m_{n-1,j-1,k}}{2\Delta x_1 \Delta x_2}$$

Quantras

Time discretization

Matrix A_m is decomposed into four sub-matrices: $\mathbf{A}^m = \mathbf{A}_0^m + \mathbf{A}_1^m + \mathbf{A}_2^m + \mathbf{A}_3^m$

The matrix $A^m{}_0$ is the part of A that comes from the FD discretization of the mixed derivative terms in the PDE, while the matrices $A^m{}_1$, $A^m{}_2$ and $A^m{}_3$ are the three parts of A^m that correspond to the spatial derivatives in the x_1-, x_2- and x_3-directions, respectively.

The matrices $A^m{}_1$, $A^m{}_2$ and $A^m{}_3$ are block-diagonal, with tri-diagonal blocks.

Hunsdorfer-Verwer scheme

Parameter θ is directly related to the stability
½ < θ < 1,
- θ = 1/2 is the most accurate,
- θ = 1 gives more damping of the error terms

$$\begin{cases} \mathbf{v}_0 = \mathbf{u}^{m-1} + \Delta\tau \mathbf{A}^{m-1}\mathbf{u}^{m-1}, \\ (\mathbf{I} - \theta\Delta\tau\mathbf{A}_i^m)\mathbf{v}_i = \mathbf{v}_{i-1} - \theta\Delta\tau\mathbf{A}_i^{m-1}\mathbf{u}^{m-1}, \quad i = 1,2,3, \\ \tilde{\mathbf{v}}_0 = \mathbf{v}_0 + \frac{1}{2}\Delta\tau(\mathbf{A}^m\mathbf{v}_3 - \mathbf{A}^{m-1}\mathbf{u}^{m-1}), \\ (\mathbf{I} - \theta\Delta\tau\mathbf{A}_i^m)\tilde{\mathbf{v}}_i = \tilde{\mathbf{v}}_{i-1} - \theta\Delta\tau\mathbf{A}_i^m\mathbf{v}_3, \quad i = 1,2,3, \\ \mathbf{u}^m = \tilde{\mathbf{v}}_3. \end{cases}$$

Apply the HV scheme with θ = 1 for the first 2 time-steps and then switch to θ = 1/2 for the remaining time-steps. We refer to this time-stepping technique as *HV smoothing*.

Quantras

• The H-V splitting scheme treats the mixed derivative part $\mathbf{A}^{m}_{0}$ in a fully explicit way while the $\mathbf{A}^{m}_{i}$ terms (i = 1, 2, 3), are treated implicitly. H-V scheme is composed of 2 phases:

• The first phase (steps 1-2) can be viewed as an **explicit Euler predictor** step followed by three **implicit**, but **unidirectional corrector** steps aiming to stabilize the predictor step.

• Phase two (steps 3-4) **restores second order convergence** of the discretization method if the PDE contains mixed derivatives, while retaining the **unconditional stability** of the scheme.

• Since the matrices $\mathbf{A}^{m}_{i}$ are tri-diagonal, the number of floating-point operations per time-step is directly proportional to the number of points in the grid, which yields a significant reduction in computational cost.

•Moreover, the block diagonal structure of these matrices gives rise to a natural, efficient parallelization for the solutions of the linear systems in Steps 2-4.

Quantras

The first phase of the H-V scheme takes care of:

- data loading ,
- computation when stepping through time, and consists of the following steps:

(a) Computes the matrices $\mathbf{A}_i^m,\ i = 0, 1, 2, 3$ and

$\widehat{\mathbf{A}}_i^m,\ i = 1, 2, 3$ $\quad \widehat{\mathbf{A}}_i^m = \mathbf{I} - \theta \Delta\tau \mathbf{A}_i^m$

and the vectors $\mathbf{w}_i = \Delta\tau \mathbf{A}_i^{m-1}\mathbf{u}^{m-1}$ and $\mathbf{v_0}$

(b) $\widehat{\mathbf{v}}_1 = \mathbf{v}_0 - \theta \mathbf{w}_1$ and solve $\widehat{\mathbf{A}}_1^m \mathbf{v}_1 = \widehat{\mathbf{v}}_1$

(c) $\widehat{\mathbf{v}}_2 = \mathbf{v}_1 - \theta \mathbf{w}_2$ and solve $\widehat{\mathbf{A}}_2^m \mathbf{v}_2 = \widehat{\mathbf{v}}_2$

(d) $\widehat{\mathbf{v}}_3 = \mathbf{v}_2 - \theta \mathbf{w}_3$ and solve $\widehat{\mathbf{A}}_3^m \mathbf{v}_3 = \widehat{\mathbf{v}}_3$

Among these steps, Steps b, c, d are inherently parallelizable, due to the block diagonal structure of the matrices $\mathbf{A^m_i}$ while it is not as obvious how Step a, in particular the computation of the vectors $\mathbf{w_i}$, i = 0, 1, 2, 3, can be efficiently parallelized. This is the most sensitive part of the algorithm.

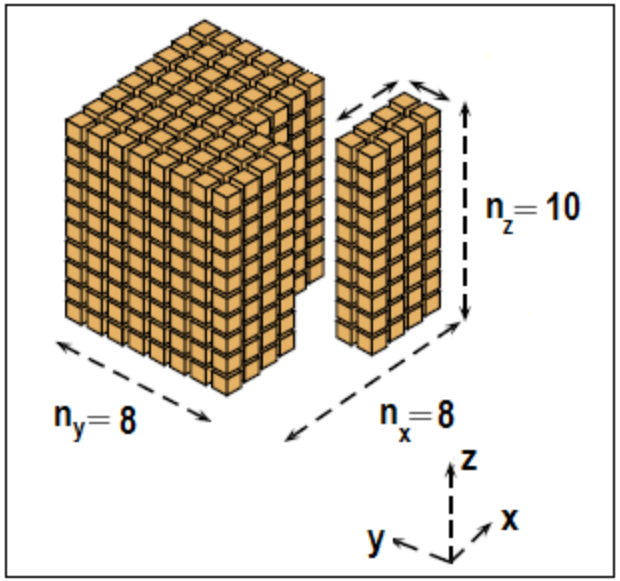
Quantras

• During each iteration, each tread-block loads data from the global memory to its shared memory. The computational grid is partitioned into a grid of blocks, then grid-points are assigned to threads of thread-blocks.

• Discretization grid of $n_x \times n_y \times n_z$ points. We can view a set of n_z consecutive grid-points in the z-direction as a "stack" of n_y grid-points. Clearly, there are $n_x \times n_y$ such stacks in the discretization grid.

• Distributing the data and computation of step a is to assign the work associated with each stack of n_y grid-points to a different thread. Thus, there is an one-to-one correspondence between the set of stacks and the set of threads.

• We partition the computational grid of size $n_x \times n_y \times n_z$ into 3-D blocks, each of which can be viewed as consisting of n_z 2-D blocks,



• For each time step, each thread of a thread block carries out all the computations/work associated with one grid-point, and each thread block processes one tile.

Quantras

Construction of the matrices:

• Note that each of the matrices $\mathbf{A}^{\mathbf{m}}_{\mathbf{i}}$ has a total of $\mathbf{n_x \times n_y \times n_z}$ rows, with each row corresponding to a grid-point of the computational domain. The approach is to assign each of the threads to assemble n_z rows of each of the matrices (a total of three entries per row of each matrix, since all matrices are tri-diagonal).

• There is a total of $\mathbf{n_x \times n_y}$ consecutive rows are constructed in parallel by the thread blocks during each iteration.

Compuation of the vectors:

• Note that the data of the previous time step, i.e. the vector $\mathbf{u^{m-1}}$, and model constant parameters, if any, are available in the global memory and constant cache, respectively.

• We emphasize that the data copying from the host memory to the device memory **occurs on the first time step only**, for the initial condition data, $\mathbf{u_0}$, and the model constants. Data for the subsequent time steps and the ADI time stepping scheme are stored on the device memory.

Memory coalescing:

• To optimize performance, one must ensure coalesced data loads from the global memory. Threads in a warp should execute the same instruction at any given time, in order to avoid potential serial execution of threads.

Quantras

Steps b-c-d:

• After step a, all the matrices and the vectors are stored in the device memory. The solution of the tri-diagonal systems is based on the parallelism arising from independent tri-diagonal solutions, rather the parallelism within each one. To this end, each independent tri-diagonal system is assigned to a different thread. Moreover, when we solve in one direction, the data are partitioned with respect to the other two.

• An approach that can be employed for the solution of the tri-diagonal systems arising in Steps b-c-d is to use parallel cyclic reduction methods.

• Pricing the options is achieved by solving the PDE backward in time from the maturity T of the option to the current time. The payoff function provides the terminal condition at time T.

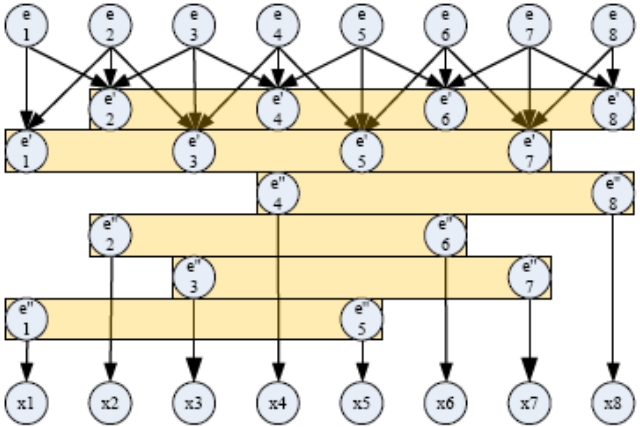
• Regarding the parallel implementation, the host (CPU) first sets up the payoff vectors (terminal conditions), then offloads the computations of the numerical solution for the PDE to a GPU.

• The ADI technique is employed for each time step and is executed in a highly parallel fashion.

Quantras

PCR method

Parallel Cyclic Reduction (PCR)
log2(8) = 3 steps.
The vector unit is fully utilized across all steps.
More work per step, fewer steps.



After log2(n) forward reduction steps, the original matrix system is reduced to n linear independent equations each that can be directly solved.

The parallelism of PCR is maintained at n throughout the algorithm.

Use of hybrid of CR + PCR proposed by Y. Zhang and Cohen.

Quantras

4. Practical applications of ADI Schemes

FX swaps

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

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

Quantras

5. Conclusions & future work

- The ADI schemes improve the quality of solving PDEs with mixed derivatives, because they are:
  - ➢Unconditionally stable
  - ➢O(2) accurate in both space & time
  - ➢Computationally faster than the common pure implicit FDMs.

- ADI methods offer more potential for the use of GPU because they allow for a lot more parallelism.

- Exploring new **hybrid methods** for Tri-diagonal solvers to be used for the implicit pass.

Quantras

Bibliography

J. CRAIG AND A. SNEYD, *An alternating-direction implicit scheme for parabolic equations with mixed derivatives*, Comp. Math. Appl., **16** (1988), pp. 341–350.

W. HUNDSDORFER AND J. VERWER, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations, Springer Series in Computational Mathematics*, vol. **33**, Springer-Verlag, Berlin,2003.

J. DOUGLAS and H. RACHFORD, On the numerical solution of heat conduction problems in two and three space varaibles, Trans. Amer. Math. Soc., **82** (1956), pp. 421-439.

K. IN'T HOUT AND B. WELFERT, *Unconditional stability of second-order ADI schemes applied to multi-dim. diffusion equations with mixed derivative terms*, Appl. Numer. Math, **59** (2009), pp. 677–692.

Duy Minh Dang, C. Christara, K.R. Jackson, *A parallel implementation on GPU of ADI finite difference methods for parabolic PDEs with application in finance,* Canadian Applied Mathematics Quarterly, 2011.

Z. Wei, B. Jangb, Y. Zhang, Y. Jia, *Parallelizing Alternating Direction Implicit Solver on GPUs*, Procedia Computer Science **18** (2013) 389 – 398

Quantras

Q & A