Handling the time-wise dimension by FTs, BSDEs and GPUs

Kees Oosterlee

CWI - Center for Mathematics and Computer Science, Amsterdam, and
Delft University of Technology, Delft, the Netherlands

PinT workshop, Jülich, May 2014

Agenda

- The power of Fourier Techniques (FT),
  - Helmholtz equation vs wave equation
  - Feynman-Kac Theorem, COS method
  - Connection to Backward Stochastic Differential Equations, BCOS method
- Alternative: Monte Carlo methods
- The use of Graphics Processing Units (GPUs)

$\Rightarrow$ Try to avoid details of financial mathematics ...

Issues

Fourier techniques remind us of difficulties:

Issues

Fourier techniques remind us of difficulties:

- Constant coefficients

Issues

Fourier techniques remind us of difficulties:

- Constant coefficients
- Boundary conditions

Issues

Fourier techniques remind us of difficulties:

- Constant coefficients
- Boundary conditions
- Non-linear problems

Case 1: Seismics in Time or Frequency Domain

- Consider the wave equation with wave speed c.

$$\frac{\partial^2 v}{\partial t^2} = c^2 \boldsymbol{\Delta} v.$$

- Time harmonic standing waves of frequency ω; $v(\boldsymbol{x}, t) = e^{-j\omega t} u(\boldsymbol{x})$, $u(\boldsymbol{x})$ satisfies the Helmholtz equation (with wave number $k = \omega/c$),

$$(\boldsymbol{\Delta} + k^2) u = f(\boldsymbol{x}).$$

- $f(\boldsymbol{x})$ is a harmonic disturbance $f(\boldsymbol{x}) e^{-j\omega t}$ which is producing the waves.
- The solutions of the Helmholtz equation represent (the spatial part of) solutions of the wave equation.

Time or Frequency Domain

- In 2D, so-called two-way wave-equation migration can be carried out efficiently by working in the frequency domain. (The linear system is solved once with a direct method for each frequency.)
- The result can be used for the computation of all the wave fields for all shots and also for the back-propagated receiver wave fields.
- This method is an order of magnitude faster than its time domain counterpart, when many shots need to be processed.
- In 3D direct solution methods can no longer be efficiently applied. But also time-domain processing has its disadvantages.

$\Rightarrow$ Need for efficient iterative solution methods for the Helmholtz equation.

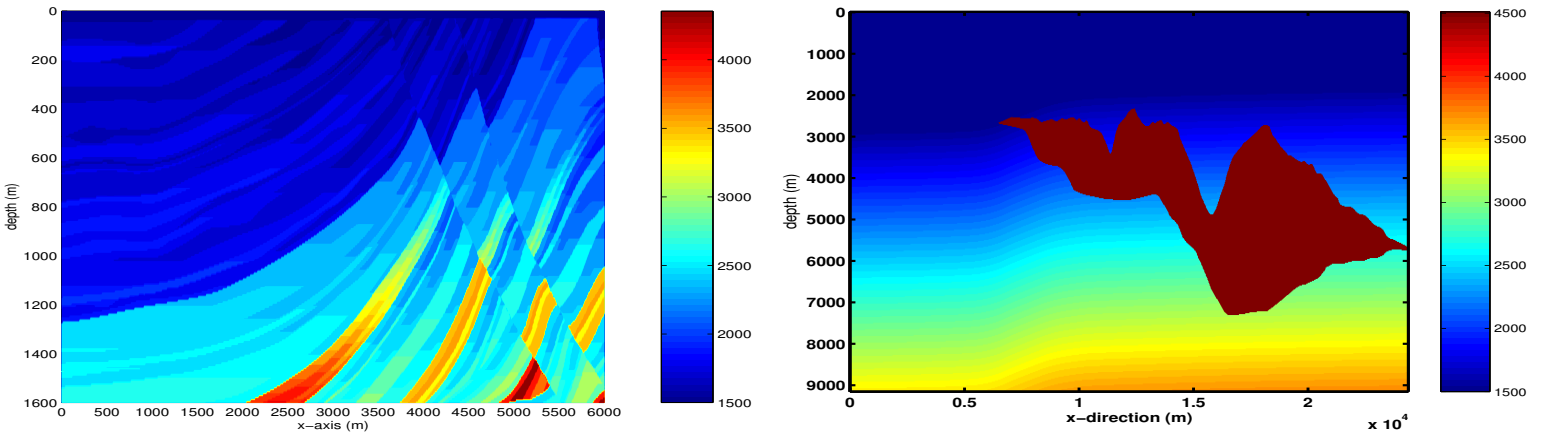
Mathematical Model

Consider the Helmholtz equation as follows:

$$-\Delta u - k^2(1-\alpha i)u = f \text{ in } \quad \Omega \subset \mathcal{R}^3, \tag{1}$$

and appropriate boundary condition where

- $\Delta = \partial_{xx} + \partial_{yy} + \partial_{zz}$, $k = \frac{\omega}{c}$ is the wavenumber, ω the angular frequency,
- The medium is barely attenuative if $0 \leq \alpha << 1$, ($i = \sqrt{-1}$, imaginary unit).
- c is the velocity which varies with position,
- u is the pressure field, f is the source term.



Solution Methods

Using 5-point finite difference stencil ($\mathcal{O}(h^2)$), we obtain a very large linear system:

$$\mathbf{Au} = \mathbf{f}, \tag{2}$$

where $\mathbf{A} \in \mathcal{C}^{N \times N}$, $\mathbf{u}, \mathbf{f} \in \mathcal{C}^N$.

- $\surd$ $\mathbf{A}$ is a sparse, symmetric.
- $\surd$ Indefinite for sufficiently large wavenumbers k.
- $\surd$ Real part of the eigenvalues of $\mathbf{A}$ are positive and negative.
  By preconditioning, we solve the equivalent linear system with a Krylov subspace method:

$$\mathbf{AM}^{-1}\hat{\mathbf{u}} = \mathbf{f}, \quad \hat{\mathbf{u}} = \mathbf{Mu}. \tag{3}$$

Shifted Laplacian preconditioner

- Erlangga, Vuik and O., 2006 proposed a preconditioner of the following form:

$$-\Delta\hat{u} - (1 + \frac{1}{2}i)k^2\hat{u} = u. \tag{4}$$

This is a complex-shifted Laplacian operator.
To compute approximately M^{-1}, we employ one multigrid iteration.

- Standard multigrid coarsening,
- damped-JAC smoother,
- Prolongation operator: operator-dependent interpolation
- Restriction operator: the full weighting operator
- F(1,1)-cycle

Truly shifted Laplacian preconditioner (with F. Gaspar, C. Rodrigo)

- Alternative: preconditioner using a complex shift in the Laplacian term:

$$-(1+\beta\, i)\Delta\hat{u} - k^2\hat{u} = u. \tag{5}$$

This is called a truly complex-shifted Laplacian operator.

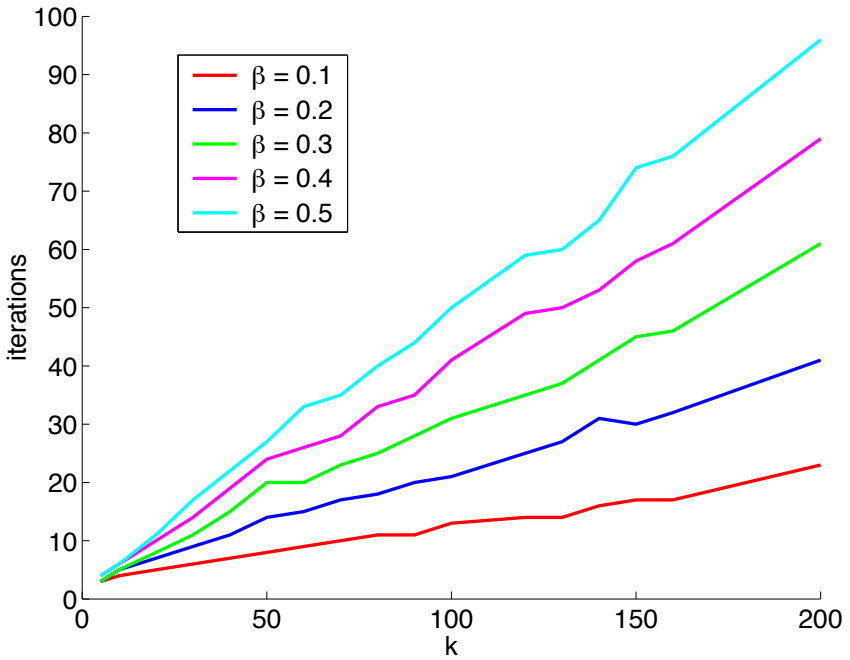


Figure: Number of Bi-CGSTAB iterations versus wave number values k for different values of β, using $k = 80$ and $h = 1/128$.

Truly shifted Laplacian preconditioner

To overcome such problems:

- Use of ω_j-Jacobi recombination as smoother when $k\,h = 1.25$ and $k\,h = 2.5$, (choice of suitable ω_j for ω_j-Jacobi and ω_j-Jacobi recombination by using LFA),
- Use of flexible GMRES.

Table: Number of flexible GMRES iterations for both multigrid preconditioners. We consider $k\,h = 0.625$

k (h)	20 (1/32)	40 (1/64)	80 (1/128)	160 (1/256)
SL (old)	14	26	56	117
SL (impr.)	12	23	41	81
TSL ($\beta = 0.3$)	10	19	37	75

Time or Frequency Domain (with H. Knibbe)

- Three-dimensional reverse-time migration, a well-known application in seismic imaging, can be efficiently performed on the basis of the Helmholtz equation on parallel hardware. Migration in the frequency domain can compete with a time-domain implementation when both are performed on a parallel architecture.
- Both methods are parallel over shots, but the frequency-domain method is also parallel over frequencies.
- The required time is dominated by the number of iterations for the highest frequency.
- Our GPU implementation of the 3D migration in frequency domain is superior to the time-domain GPU implementation.

Case 2: Feynman-Kac Theorem

- The linear partial differential equation:

$$v_t(t,x) + \mathcal{L}v(t,x) + g(t,x) = 0,\ v(T,x) = h(x),$$

with operator

$$\mathcal{L}v(t,x) = \mu^T(x)v_x(t,x) + \frac{1}{2}\mathrm{Tr}[D^2v(t,x)\sigma(x)\sigma^T(x)].$$

Feynman-Kac theorem:

$$v(t,x) = \mathbb{E}_t^x\left[\int_t^T g(s,X_s)ds + h(X_T)\right],$$

where X_s is the solution to the FSDE

$$dX_s = \mu(X_s)ds + \sigma(X_s)dW_s,\ X_t = x.$$

Financial industry; Work at banks

- Financial contract pricing approach:
  1. Start with some financial product
  2. Model asset prices involved (SDEs)
  3. Calibrate the model to market data (numerics, optimization)
  4. Model product price correspondingly (P(I)DE or integral)
  5. Price the product of interest (numerics, MC)
  6. Set up a hedge to remove the risk to the product (optimization)

Feynman-Kac Theorem (option pricing context)

Given the final condition problem

$$\begin{cases} \frac{\partial v}{\partial t} & + & \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 v}{\partial x^2} + rx\frac{\partial v}{\partial x} - rv = 0, \\ v(T,x) & = & h(T,x) = \text{ given} \end{cases}$$

Then the value, $v(t,x)$, is the unique solution of

$$v(t,x) = e^{-r(T-t)}\mathbb{E}^Q\{v(T,X_T)|\mathcal{F}_t\}$$

with the sum of the first derivatives of the option square integrable.
and $x = X_t$ satisfies the system of stochastic differential equations:

$$dX_t = rX_t dt + \sigma X_t dW_t^Q,$$

- Similar relations also hold for other (multi-D) SDEs and PDEs!

A pricing approach

$$v(t_0, x) = e^{-r(T-t_0)}\mathbb{E}^Q\{v(T, X_T)|\mathcal{F}_0\}$$

Quadrature:

$$v(t_0, x) = e^{-r(T-t_0)} \int_{\mathbb{R}} v(T, y) f(y, x) dy$$

- Trans. PDF, $f(y, x_0)$, typically not available, but the characteristic function, ϕ, often is.
- In probability theory a characteristic function of a continuous random variable X, equals the Fourier transform of the density of X.

$$\phi(u) = \int_{-\infty}^{\infty} e^{iuy} f(x, y) dy;$$

Class of AD processes

Suppose we have given a following system of SDEs:

$$d\mathbf{X}_t = \mu(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t,$$

with independent Brownian motions $\mathbf{W_t}$. For processes in the affine diffusion (AD) class it is assumed that drift, volatility, and interest rate components are of the affine form, like

$$\begin{aligned} \mu(\mathbf{X}_t) &= a_0 + a_1\mathbf{X}_t \text{ for } (a_0, a_1) \in \mathbb{R}^n \times \mathbb{R}^{n\times n}, \\ \sigma(\mathbf{X}_t)\sigma(\mathbf{X}_t)^T &= (c_0)_{ij} + (c_1)_{ij}^T\mathbf{X}_t, (c_0, c_1) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n\times n}, \end{aligned}$$

Characteristic function for AD

Duffie, Pan and Singleton (2000) have shown that for affine diffusion processes the characteristic function, as:

$$\phi(\mathbf{u}) = e^{A(\mathbf{u},t,T)+\mathbf{B}(\mathbf{u},t,T)^T \mathbf{X_t}},$$

The coefficients $A(\mathbf{u}, t, T)$ and $\mathbf{B}(\mathbf{u}, \mathbf{t}, \mathbf{T})^{\mathbf{T}}$ satisfy a system of Riccati-type ODEs.

Fourier-Cosine Expansions, COS Method (with F. Fang)

- The COS method:
  - Exponential convergence;
- All based on the availability of a characteristic function.
- The basic idea:
  - Replace the density by its Fourier-cosine series expansion;
  - Series coefficients have simple relation to characteristic function.

Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$h(x) = \sum_{k=0}^{\infty}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

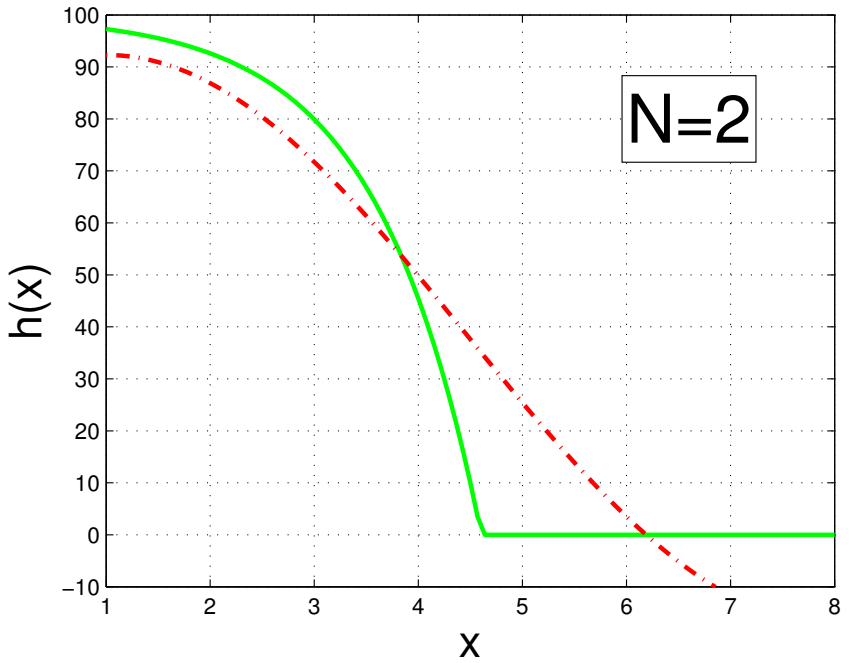
$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

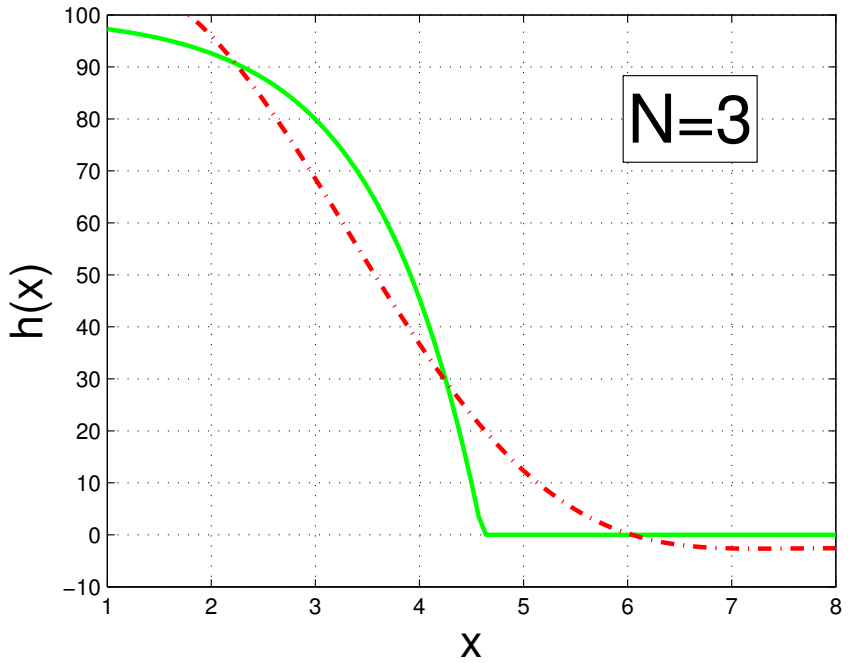


Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

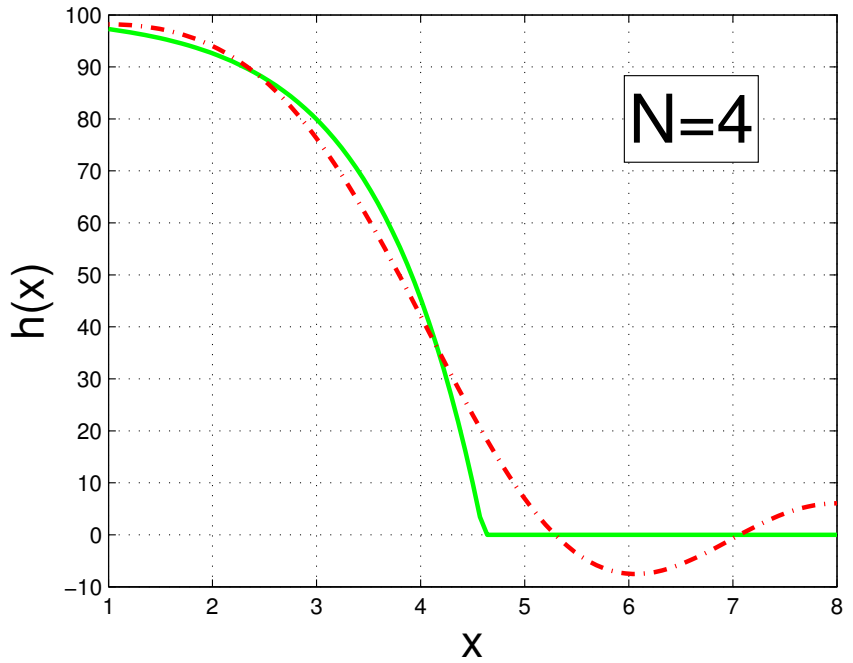


Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

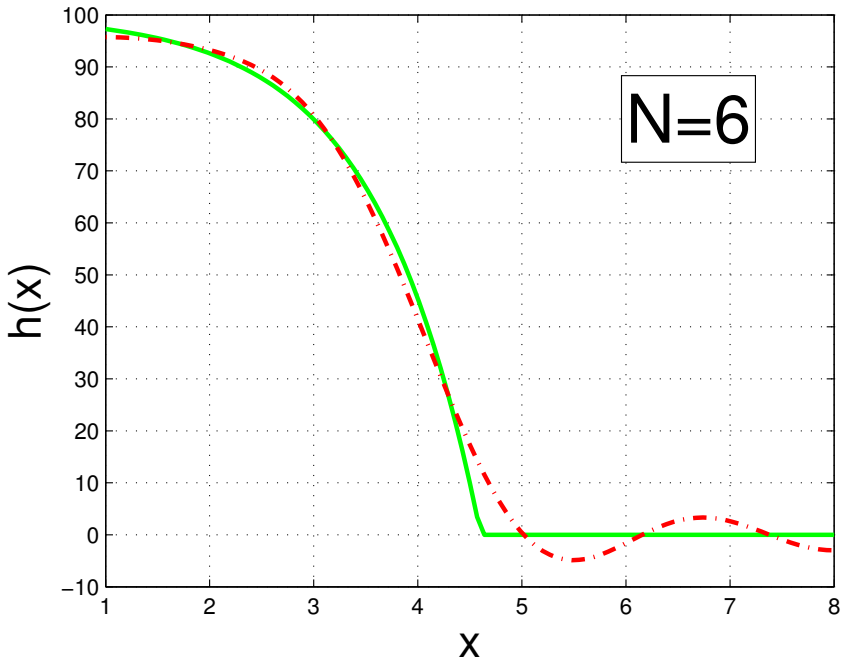


Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

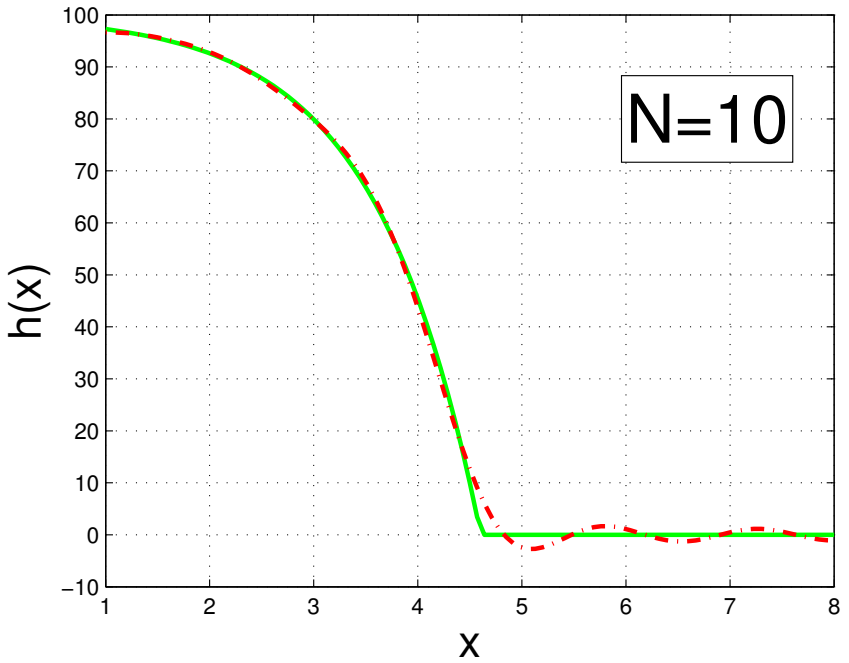


Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$

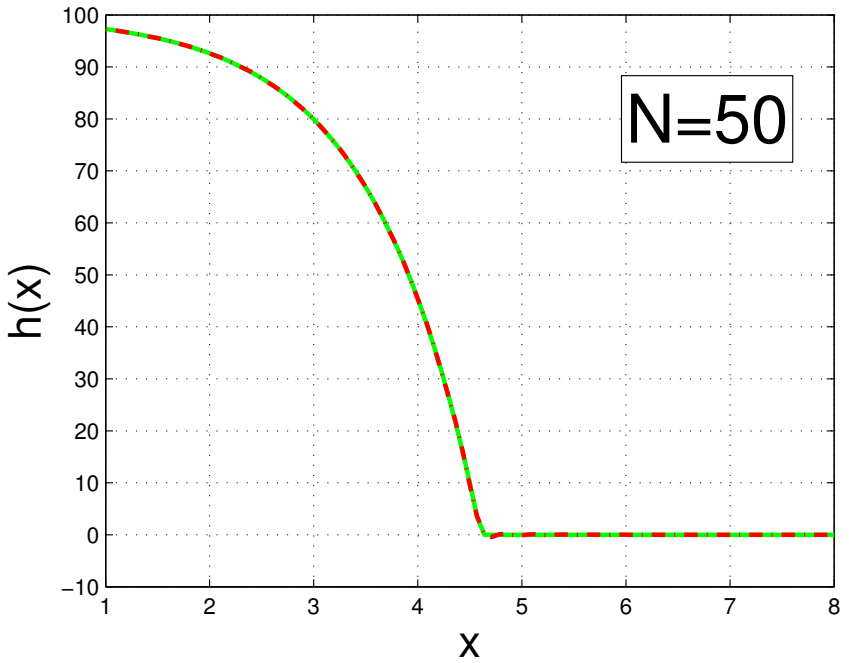


Fourier-cosine series expansion of function $h(x)$ on $[a, b]$:

$$\hat{h}(x) = \sum_{k=0}^{N-1}{}' H_k \cos\left(k\pi \frac{x-a}{b-a}\right), \qquad x \in [a, b], \tag{6}$$

with

$$H_k = \frac{2}{b-a} \int_a^b h(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{7}$$



Series Coefficients of the Density and the Ch.F.

- Fourier-Cosine expansion of density function on interval $[a, b]$:

$$f(x) = \sum_{n=0}^{\infty}{}' F_n \cos\left(n\pi \frac{x-a}{b-a}\right),$$

with $x \in [a, b] \subset \mathbb{R}$ and the coefficients defined as

$$F_n := \frac{2}{b-a} \int_a^b f(x) \cos\left(n\pi \frac{x-a}{b-a}\right) dx.$$

- F_n has direct relation to ch.f., $\phi(u) := \int_{\mathbb{R}} f(x) e^{iux} dx$ ($\int_{\mathbb{R}\setminus[a,b]} f(x) \approx 0$),

$$\begin{aligned} F_n \approx A_n &:= \frac{2}{b-a} \int_{\mathbb{R}} f(x) \cos\left(n\pi \frac{x-a}{b-a}\right) dx \\ &= \frac{2}{b-a} \mathrm{Re}\left\{\phi\left(\frac{n\pi}{b-a}\right) \exp\left(-i\frac{ka\pi}{b-a}\right)\right\}. \end{aligned}$$

Pricing European Options

- Start from the risk-neutral valuation formula:

$$v(t_0,x) = e^{-r\Delta t}\mathbb{E}^{\mathbb{Q}}\left[v(T,y)|\mathcal{F}_0\right] = e^{-r\Delta t}\int_{\mathbb{R}} v(T,y)f(y,x)dy.$$

- Truncate the integration range:

$$v(t_0,x) = e^{-r\Delta t}\int_{[a,b]} v(T,y)f(y,x)dy + \varepsilon.$$

- Replace the density by the COS approximation, and interchange summation and integration:

$$\hat{v}(t_0,x) = e^{-r\Delta t}\sum_{n=0}^{N-1}{}' \operatorname{Re}\left\{\phi\left(\frac{n\pi}{b-a};x\right)e^{-in\pi\frac{a}{b-a}}\right\}H_n,$$

where the series coefficients of the payoff, H_n, are analytic.

Pricing European Options

- Log-asset prices: $x := \ln(S_0/K) \quad \text{and} \quad y := \ln(S_T/K),$
- The payoff for European options reads

$$v(y,T) \equiv [\alpha \cdot K(e^y - 1)]^+.$$

- For a call option, we obtain

$$\begin{aligned} H_k^{call} &= \frac{2}{b-a}\int_0^b K(e^y-1)\cos\left(k\pi\frac{y-a}{b-a}\right)dy \\ &= \frac{2}{b-a}K\left(\chi_k(0,b)-\psi_k(0,b)\right), \end{aligned}$$

- For a vanilla put, we find

$$H_k^{put} = \frac{2}{b-a}K\left(-\chi_k(a,0)+\psi_k(a,0)\right).$$

GPU Results (Single Precision)

$$v_t = \frac{1}{2}x^2 y v_{xx} + \rho\gamma x y v_{xy} + \frac{1}{2}\gamma^2 y v_{yy} + rxv_x + \kappa(\overline{\sigma} - y)v_y - rv.$$

- It is difficult to speed-up efficient numerical techniques on modern hardware.
- GPU computing: Benefit from multiple strikes for parallelism, 21 IC's.

Heston model				
	N	64	128	256
MATLAB	msec	3.850890	7.703350	15.556240
	max.abs.err	6.0991e-04	2.7601e-08	$< 10^{-14}$
GPU	msec	0.177860	0.209093	0.333786
	max.abs.err	0.000534	0.000144	0.000144

Table: Convergence and maximum absolute error when pricing a vector of 21 strikes.

- Exponential convergence, Error analysis in our papers.
- Also work with wavelets instead of cosines.

Coefficients in 2D

$$F_{k_1,k_2}(\mathbf{x}) \approx \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1}\right)\cos\left(k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\,dy_2.$$

Coefficients in 2D

$$F_{k_1,k_2}(\mathbf{x}) \approx \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1}\right)\cos\left(k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\, dy_2.$$

We use the following goniometric relation:

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta)+\cos(\alpha-\beta).$$

Coefficients in 2D

$$F_{k_1,k_2}(\mathbf{x}) \approx \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1}\right)\cos\left(k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\,dy_2.$$

We use the following goniometric relation:

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta)+\cos(\alpha-\beta).$$

Then

$$2F_{k_1,k_2}(\mathbf{x}) = F^{+}_{k_1,k_2}(\mathbf{x}) + F^{-}_{k_1,k_2}(\mathbf{x}),$$

where

$$\begin{aligned} F^{\pm}_{k_1,k_2}(\mathbf{x}) &:= \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1} \pm k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\,dy_2 \\ &= \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\mathrm{Re}\left\{\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \exp\left(ik_1\pi\frac{y_1-a_1}{b_1-a_1} \pm ik_2\pi\frac{y_2-\pm a_2}{b_2-a_2}\right) dy_1\,dy_2\right\} \end{aligned} \tag{8}$$

Coefficients in 2D

$$F_{k_1,k_2}(\mathbf{x}) \approx \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1}\right)\cos\left(k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\,dy_2.$$

We use the following goniometric relation:

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta)+\cos(\alpha-\beta).$$

Then

$$2F_{k_1,k_2}(\mathbf{x}) = F^{+}_{k_1,k_2}(\mathbf{x}) + F^{-}_{k_1,k_2}(\mathbf{x}),$$

where

$$\begin{aligned} F^{\pm}_{k_1,k_2}(\mathbf{x}) &:= \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \cos\left(k_1\pi\frac{y_1-a_1}{b_1-a_1} \pm k_2\pi\frac{y_2-a_2}{b_2-a_2}\right) dy_1\,dy_2 \\ &= \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\mathrm{Re}\left\{\iint_{\mathbb{R}^2} f(\mathbf{y}|\mathbf{x}) \exp\left(ik_1\pi\frac{y_1-a_1}{b_1-a_1} \pm ik_2\pi\frac{y_2-\pm a_2}{b_2-a_2}\right) dy_1\,dy_2\right\} \\ &= \frac{2}{b_1-a_1}\frac{2}{b_2-a_2}\mathrm{Re}\left\{\varphi\left(\frac{k_1\pi}{b_1-a_1}, \pm\frac{k_2\pi}{b_2-a_2}\middle|\mathbf{x}\right\}\exp\left(-ik_1\pi\frac{a_1}{b_1-a_1} \mp ik_2\pi\frac{a_2}{b_2-a_2}\right)\right). \end{aligned} \quad (8)$$

Case 3: Semilinear PDE and BSDEs (with Marjon Ruijter)

- The semilinear partial differential equation:

$$v_t(t,x) + \mathcal{L}v(t,x) + g(t,x,v,\sigma^T(x)Dv(t,x)) = 0,\ v(T,x) = h(x),$$

with operator

$$\mathcal{L}v(t,x) = \mu^T(x)Dv(t,x) + \frac{1}{2}\mathrm{Tr}[D^2v(t,x)\sigma(x)\sigma^T(x)].$$

We can solve this PDE by means of the FSDE:

$$dX_s = \mu(X_s)ds + \sigma(X_s)dW_s,\ X_t = x.$$

and the BSDE:

$$dY_s = -g(s,X_s,Y_s,Z_s)ds + Z_s dW_s,\ Y_T = h(X_T).$$

- Theorem $(t \leq s \leq T)$:

$$Y_s^{t,x} = v(s,X_s^{t,x}),\ Z_s^{t,x} = \sigma^T(s,X_s^{t,x})Dv(s,X_s^{t,x}).$$

is the solution to the BSDE.

Forward SDE, first

- Well-known is an FSDE, and its discretization, like

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t,\ X_0 = x_0.$$

Forward SDE, first

- Well-known is an FSDE, and its discretization, like

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t,\ X_0 = x_0.$$

Partition $0 = t_0 < t_1 < \ldots < t_m < \ldots < t_M = T$, with

$$\Delta t = t_{m+1} - t_m \text{ and } \Delta W_m = W_{m+1} - W_m \sim \mathcal{N}(0, \Delta t).$$

Euler discretization, $m = 0 : M - 1$:

$$X_0 = x_0,\ X_{m+1} \approx X_m + \mu(X_m)\Delta t + \sigma(X_m)\Delta W_m.$$

Backward SDE

$$dY_t = -g(t, Y_t, Z_t)dt + Z_t dW_t, \quad Y_T = h(X_T).$$

A solution is a pair of adapted processes (Y, Z), satisfying:

$$Y_t = h(X_T) + \int_t^T g(s, Y_s, Z_s)ds - \int_t^T Z_s dW_s.$$

- Y is adapted if and only if, for every realization and every t, Y_t is known at time t. Adapted processes cannot "see into the future".

Backward SDE

$$dY_t = -g(t, Y_t, Z_t)dt + Z_t dW_t, \quad Y_T = h(X_T).$$

A solution is a pair of adapted processes (Y, Z), satisfying:

$$Y_t = h(X_T) + \int_t^T g(s, Y_s, Z_s)ds - \int_t^T Z_s dW_s.$$

- Y is adapted if and only if, for every realization and every t, Y_t is known at time t. Adapted processes cannot "see into the future".
- A BSDE is not a time-reversed FSDE, because at time t the pair (Y_t, Z_t) is $\mathcal{F}_t$-measurable and the process does not "know" the terminal condition yet.

Discretization BSDE

$$-dY_t = g(t, Y_t, Z_t)dt - Z_t dW_t, \;\; Y_T = h(X_T).$$

g is the driver function. $h(X_T)$ is $\mathcal{F}_T$-measurable random variable.

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s)ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Discretization BSDE

$$-dY_t = g(t, Y_t, Z_t)dt - Z_t dW_t, \;\; Y_T = h(X_T).$$

g is the driver function. $h(X_T)$ is $\mathcal{F}_T$-measurable random variable.

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s)ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Euler discretization, backward in time ($\Delta W_m = W_{m+1} - W_m$):

$$Y_m \approx Y_{m+1} + g(t_m, Y_m, Z_m)\Delta t - Z_m \Delta W_m, \;\; Y_M = h(X_T).$$

Problem: To compute Y_m we need unknown Y_{m+1}. The above backward equation does not take into account the adaptability constraint on Y and Z.

Y-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Taking conditional expectation $\mathbb{E}_m[\cdot] = \mathbb{E}[\cdot|\mathcal{F}_{t_m}]$:

$$Y_m = \mathbb{E}[Y_{m+1}] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s)] ds - \mathbb{E}_m \left[\int_{t_m}^{t_{m+1}} Z_s dW_s \right].$$

Y-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Taking conditional expectation $\mathbb{E}_m[\cdot] = \mathbb{E}[\cdot|\mathcal{F}_{t_m}]$:

$$Y_m = \mathbb{E}[Y_{m+1}] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s)] ds - 0$$

Y-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Taking conditional expectation $\mathbb{E}_m[\cdot] = \mathbb{E}[\cdot|\mathcal{F}_{t_m}]$:

$$\begin{aligned} Y_m &= \mathbb{E}[Y_{m+1}] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s)] ds - 0 \\ &\approx \mathbb{E}_m[Y_{m+1}] + g(t_m, Y_m, Z_m)\Delta t \end{aligned}$$

Z-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Multiplying by ΔW_m, taking the conditional expectation gives:

$$\begin{aligned} \mathbb{E}_m[Y_m \Delta W_m] &= \mathbb{E}[Y_{m+1} \Delta W_m] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s) \Delta W_m] ds \\ &- \mathbb{E}_m \left[\int_{t_m}^{t_{m+1}} Z_s dW_s \Delta W_m \right]. \end{aligned}$$

Z-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s)ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Multiplying by ΔW_m, taking the conditional expectation gives:

$$\begin{aligned} 0 &= \mathbb{E}[Y_{m+1}\Delta W_m] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s)\Delta W_m]ds \\ &- \mathbb{E}_m\left[\int_{t_m}^{t_{m+1}} Z_s dW_s \Delta W_m\right]. \end{aligned}$$

Z-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Multiplying by ΔW_m, taking the conditional expectation gives:

$$\begin{aligned} 0 &= \mathbb{E}[Y_{m+1} \Delta W_m] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s) \Delta W_m] ds \\ &- \mathbb{E}_m \left[\int_{t_m}^{t_{m+1}} Z_s \right] ds. \end{aligned}$$

Z-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Multiplying by ΔW_m, taking the conditional expectation gives:

$$\begin{aligned} 0 &= \mathbb{E}[Y_{m+1} \Delta W_m] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s) \Delta W_m] ds \\ &- \mathbb{E}_m \left[\int_{t_m}^{t_{m+1}} Z_s \right] ds. \\ &\approx \mathbb{E}_m[Y_{m+1} \Delta W_m] - \Delta t Z_m. \end{aligned}$$

Z-component

$$Y_m = Y_{m+1} + \int_{t_m}^{t_{m+1}} g(s, Y_s, Z_s) ds - \int_{t_m}^{t_{m+1}} Z_s dW_s.$$

Multiplying by ΔW_m, taking the conditional expectation gives:

$$\begin{aligned} 0 &= \mathbb{E}[Y_{m+1} \Delta W_m] + \int_{t_m}^{t_{m+1}} \mathbb{E}_m[g(s, Y_s, Z_s) \Delta W_m] ds \\ &- \mathbb{E}_m \left[\int_{t_m}^{t_{m+1}} Z_s \right] ds. \\ &\approx \mathbb{E}_m[Y_{m+1} \Delta W_m] - \Delta t Z_m. \end{aligned}$$

$$Z_m \approx \frac{1}{\Delta t} \mathbb{E}_m[Y_{m+1} \Delta W_m].$$

- Discretization scheme:

$$Y_M = h(X_T)$$
$$\text{for } m = M-1, \ldots 0:$$

$$\begin{aligned} Z_m &= \frac{1}{\Delta t}\mathbb{E}_m[Y_{m+1}\Delta W_m], \\ Y_m &= \mathbb{E}_m[Y_{m+1}] + g(t_m, Y_m, Z_m)\Delta t. \end{aligned}$$

Euler method:

$$\int_{t_m}^{t_{m+1}} H(s)ds \approx H(t_m)\Delta t \quad \Rightarrow \mathcal{O}(\Delta t)$$

- Discretization scheme:

$Y_M = h(X_T)$
for $m = M-1, \ldots 0$:

$$\begin{aligned} Z_m &= \frac{1}{\Delta t}\mathbb{E}_m[Y_{m+1}\Delta W_m], \\ Y_m &= \mathbb{E}_m[Y_{m+1}] + g(t_m, Y_m, Z_m)\Delta t. \end{aligned}$$

Euler method:

$$\int_{t_m}^{t_{m+1}} H(s)ds \approx H(t_m)\Delta t \quad \Rightarrow \mathcal{O}(\Delta t)$$

θ-method:

$$\begin{aligned} \int_{t_m}^{t_{m+1}} H(s)ds &\approx \theta H(t_m)\Delta t + (1-\theta)H(t_{m+1})\Delta t, \\ \theta \in [0,1],\ \theta &= \frac{1}{2} \quad \Rightarrow \mathcal{O}((\Delta t)^2) \end{aligned}$$

Forward-backward SDE

$X_0 = x_0$
for $m = 0, \ldots, M-1$:
$X_{m+1} \approx X_m + \mu(X_m)\Delta t + \sigma(X_m)\Delta W_m$

$Y_M = g(X_M)$

for $m = M-1, \ldots 0$:
$Z_m \approx \frac{1}{\Delta t}\mathbb{E}_m[Y_{m+1}\Delta W_m]$,
$Y_m \approx \mathbb{E}_m[Y_{m+1}] + g(t_m, Y_m, Z_m)\Delta t$.

Forward-backward SDE

$X_0 = x_0$
for $m = 0, \ldots, M-1$:
$X_{m+1} \approx X_m + \mu(X_m)\Delta t + \sigma(X_m)\Delta W_m$

$Y_M = g(X_M)$

for $m = M-1, \ldots 0$:
$Z_m \approx \frac{1}{\Delta t}\mathbb{E}_m[Y_{m+1}\Delta W_m]$,
$Y_m \approx \mathbb{E}_m[Y_{m+1}] + g(t_m, Y_m, Z_m)\Delta t$.

"All processes go forward in time but the BSDE is approximated backwards in time."

$\Rightarrow$ We use the BCOS method to approximate the conditional expectations.

- Second-order accuracy with a $\theta = \frac{1}{2}$ scheme, GBM,BM, JD.

Time/spatially dependent coefficients (with M. Ruijter)

- We can write the Euler, Milstein, and 2.0-weak-Taylor SDE discretization schemes in the following general form

$$X_{m+1} = x + m(x)\Delta t + s(x)\Delta W_{m+1} + \kappa(x)(\Delta W_{m+1})^2, \quad X_m = x. \tag{9}$$

- For the Euler scheme:

$$m(x) = \mu(x), \quad s(x) = \sigma(x), \quad \kappa(x) = 0,$$

- for the Milstein scheme

$$m(x) = \mu(x) - \tfrac{1}{2}\sigma(x)\sigma_x(x), \quad s(x) = \sigma(x), \quad \kappa(x) = \tfrac{1}{2}\sigma(x)\sigma_x(x),$$

- and for the 2.0-weak-Taylor scheme

$$\begin{aligned} m(x) &= \mu(x) - \tfrac{1}{2}\sigma(x)\sigma_x(x) + \tfrac{1}{2}\left(\mu(x)\mu_x(x) + \tfrac{1}{2}\mu_{xx}(x)\sigma^2(x)\right)\Delta t, \\ s(x) &= \sigma(x) + \tfrac{1}{2}\left(\mu_x(x)\sigma(x) + \mu(x)\sigma_x(x) + \tfrac{1}{2}\sigma_{xx}(x)\sigma^2(x)\right)\Delta t, \\ \kappa(x) &= \tfrac{1}{2}\sigma(x)\sigma_x(x). \end{aligned}$$

Lemma

- The characteristic function of X_{m+1}, given $X_m = x$, in Eq. (9) is given by

$$\phi_{X_{m+1}}(u|X_m = x) = \mathbb{E}\left[\exp\left(iuX_{m+1}\right)\Big|X_m = x\right]$$
$$= \exp\left(iux + ium(x)\Delta t - \frac{\frac{1}{2}u^2 s^2(x)\Delta t}{1 - 2iu\kappa(x)\Delta t}\right)(1 - 2iu\kappa(x)\Delta t)^{-}$$

- For $\kappa(x) = 0$ it follows that

$$\phi_{X_{m+1}}(u|Xa_m = x) = e^{iux + ium(x)\Delta t - \frac{1}{2}u^2 s^2(x)\Delta t}.$$

Example nonconstant coefficients

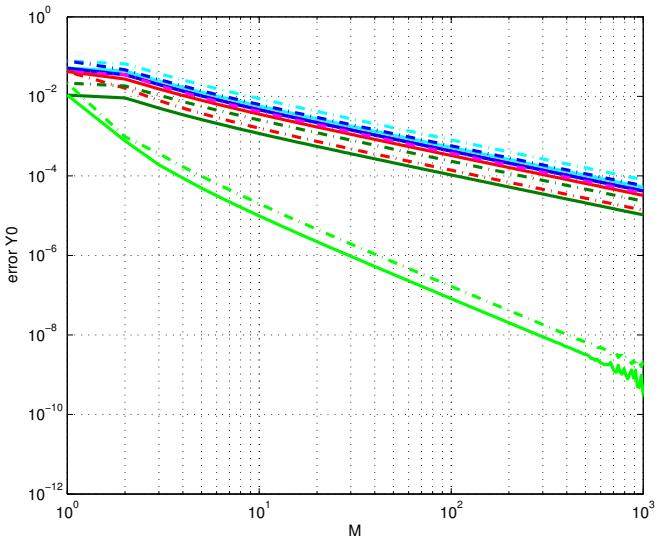
- European call option - CEV $dX_s = \bar{\mu} X_s ds + \bar{\sigma} X_s^{\gamma} dW_s$.

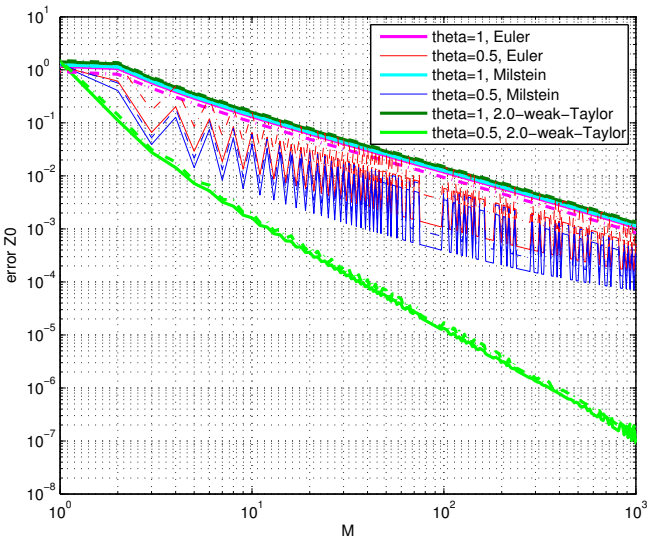
$$dY_s = -g(s, X_s, Y_s, Z_s)ds + Z_s dW_s, \quad Y_T = \max(X_T - K, 0),$$

$$g(t, x, y, z) = -ry - \frac{\mu(x) - rx}{\sigma(x)} z = -ry - \frac{\bar{\mu} - r}{\bar{\sigma}} x^{1-\gamma} z.$$

$X_0 = 100,\ K = 100,\ r = 0.1,\ \bar{\mu} = 0.2,\ T = 0.1.$

- We take $\gamma = 0.2$ and $\gamma = 0.8$.





Quasilinear PDE and coupled BSDEs (with Th. Huijskens, M. Ruijter)

- The quasilinear partial differential equation:

$$v_t + \mu^T(t,x,y,z)Dv + \frac{1}{2}\mathrm{Tr}[D^2 v\sigma\sigma^T(t,x,y)] + \tilde{g}(t,x,v,Dv,D^2 v)\} = 0$$
$$v(T,x) = h(x)$$

We can solve this PDE by means of the coupled FBSDE:

$$dX_s = \mu(s,X_s,Y_s,Z_s)ds + \sigma(s,X_s,Y_s)dW_s,\ X_t = x.$$

$$dY_s = -\tilde{g}(s,X_s,Y_s,Z_s)ds + Z_s^T\sigma(s,X_s,Y_s)dW_s,\ Y_T = h(X_T).$$

- Theorem: $Y_s^{t,x} = v(s,X_s^{t,x}),\ Z_s^{t,x} = Dv(s,X_s^{t,x}).$ is the solution to the coupled FBSDE.

$\Rightarrow$ Also a formulation exists for fully nonlinear PDEs!

Case 4: HJB equation, dynamic programming

- Suppose we consider the Hamilton-Jacobi-Bellman (HJB) equation:

$$v_t(t,x) + \sup_{a \in A}\{\mu^T(x,a)Dv(t,x) + \frac{1}{2}\mathrm{Tr}[D^2 v(t,x)\sigma\sigma^T(x,a)] + g(t,x,a)\} = 0$$

$$v(T,x) = h(x)$$

This dynamic programming equation is associated to a stochastic control problem with value function

$$v(t,x) = \sup_{\alpha} \mathbb{E}_t^x \left[\int_t^T g(s, X_s^\alpha, \alpha_s) ds + h(X_T^\alpha) \right],$$

where X_s is the solution to the controlled FSDE

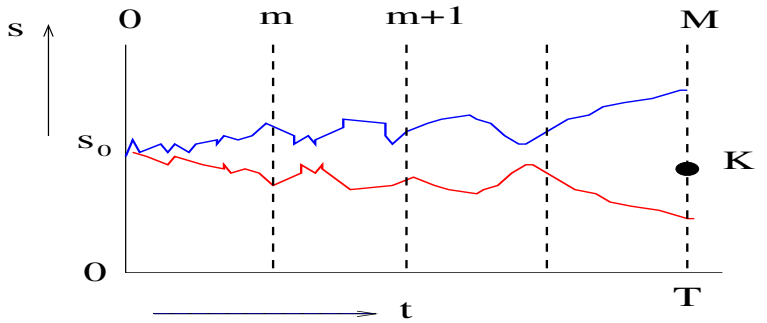
$$dX_s^\alpha = \mu(X_s^\alpha, \alpha_s)ds + \sigma(X_s^\alpha, \alpha_s)dW_s, \; X_t^\alpha = x.$$

Dynamic Programming Formulation

- Valuation with intermediate decisions to stop

Dynamic Programming Formulation

- Valuation with intermediate decisions to stop



- Bellman principle of optimality
- At the terminal time the option value is given by,

$$v(T, \mathbf{x}_T) = h(x) := \max(\tilde{h}(\mathbf{x}_T), 0). \tag{10}$$

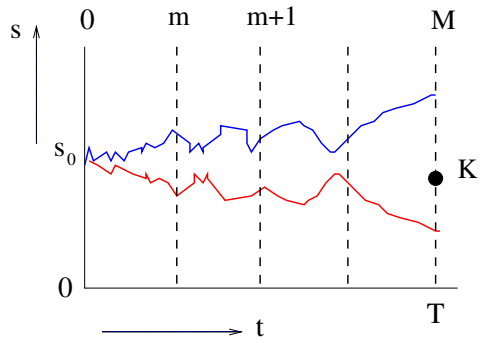
- The Bermudan option value at time t_m and state $\mathbf{x}_{t_m}$ is given by

$$v(t_m, \mathbf{x}_{t_m}) = \max(\tilde{h}(\mathbf{x}_{t_m}), c(t_m, \mathbf{x}_{t_m})). \tag{11}$$

- The continuation value c at t_m, is :

$$c(t_m, \mathbf{x}_{t_m}) = D_{t_m} \mathbb{E}\left[v(t_{m+1}, \mathbf{x}_{t_{m+1}}) | \mathcal{F}_{t_m}\right].$$

Pricing Bermudan Options



- The pricing formulas, for $m = M-1, \ldots, 1$:

$$\begin{cases} c(t_m, x) &= e^{-r\Delta t} \int_{\mathbb{R}} v(t_{m+1}, y) f(y, x) dy, \\ v(t_m, x) &= \max\left(h(t_m, x), c(t_m, x)\right), \end{cases}$$

  followed by

$$v(t_0, x) = e^{-r\Delta t} \int_{\mathbb{R}} v(t_1, y) f(y, x) dy.$$

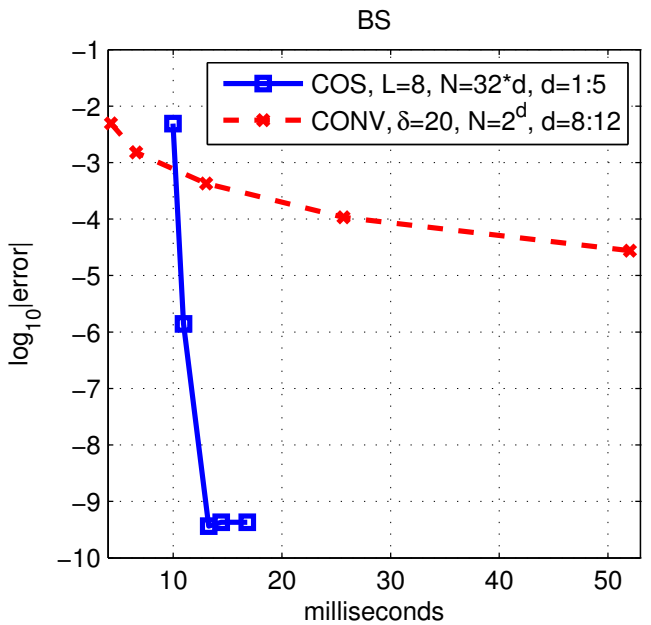
  $\Rightarrow$ A backward recursion procedure is performed on the Fourier–cosine coefficient $\mathcal{V}_k(t_{m+1})$:

$$\mathcal{V}_k(t_{m+1}) = \frac{2}{b-a} \int_a^b \max(c(t_{m+1}, x), h(t_{m+1}, x)) \cos\left(k\pi \frac{y-a}{b-a}\right) dy,$$

Bermudan puts with 10 early-exercise dates

Table: Test parameters for pricing Bermudan options

Test No.	Model	S_0	K	T	r	σ	Other Parameters
2	BS	100	110	1	0.1	0.2	–



Boundary conditions (like discrete barrier options)

- The value of an M-times monitored up-and-out option satisfies

$$\begin{cases} c(x, t_{m-1}) &= e^{-r(t_m - t_{m-1})} \int_{\mathbb{R}} v(x, t_m) f(y|x) dy \\ v(x, t_{m-1}) &= \begin{cases} e^{-r(T-t_{m-1})} R, & x \geq b \\ c(x, t_{m-1}), & x < b \end{cases} \end{cases}$$

  where $b = \ln(B/K)$, and $v(x, t_0) = e^{-r(t_m - t_{m-1})} \int_{\mathbb{R}} v(x, t_1) f(y, x) dy$.
- The technique:
  - Split the integration range at the barrier level (no Newton required)
  - Recover $H_n(t_1)$ recursively, from $H_n(t_M)$, $H_n(t_{M-1}), \cdots, H_n(t_2)$ in $O((M-1)N \log_2(N))$ operations.
  - Insert $H_n(t_1)$ in the COS formula to get $v(x, t_0)$, in $O(N)$ operations.

Stochastic Grid Bundling Method (SGBM) with S. Jain

Step 1: The grid points in SGBM are generated by simulating independent copies of sample paths, $\{\mathbf{X}_{t_0}(n), \ldots, \mathbf{X}_{t_M}(n)\},\ n = 1, \ldots, N,$

Step 2: Compute the option value for grid points at terminal time.

Step 3: Bundle the grid points at t_{m-1} into $\mathcal{B}_{t_{m-1}}(1), \ldots, \mathcal{B}_{t_{m-1}}(\nu)$ non-overlapping bundles.

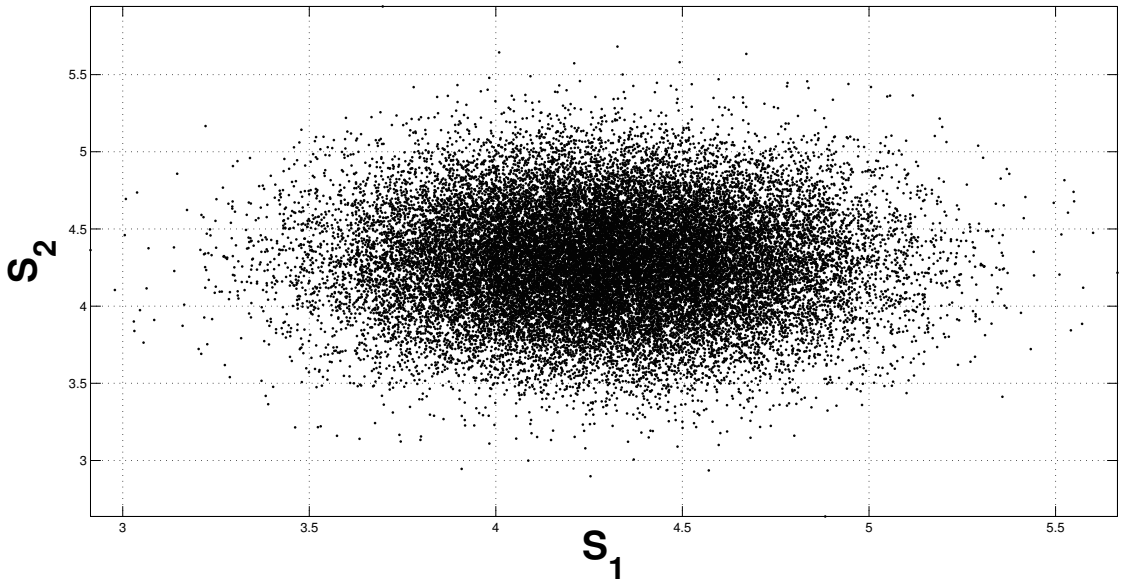
Step 4: For each $\mathcal{B}_{t_{m-1}}(\beta),\ \beta = 1, \ldots, \nu$, compute $\hat{v}(\mathbf{X}_{t_m}, \alpha_{t_m}^{\beta})$.
$\hat{v} : \mathbb{R}^d \times \mathbb{R}^K \mapsto \mathbb{R}$, is a parametrized function which assigns values to states $\mathbf{X}_{t_m}$.

Step 5: The continuation values for grid points in bundle $\mathcal{B}_{t_{m-1}}(\beta),\ \beta = 1, \ldots, \nu$, are approximated by

$$\widehat{c}(t_{m-1}, \mathbf{X}_{t_{m-1}}(n)) = \mathbb{E}[\hat{v}(\mathbf{X}_{t_m}, \alpha_{t_m}^{\beta}) | \mathbf{X}_{t_{m-1}}(n)]$$

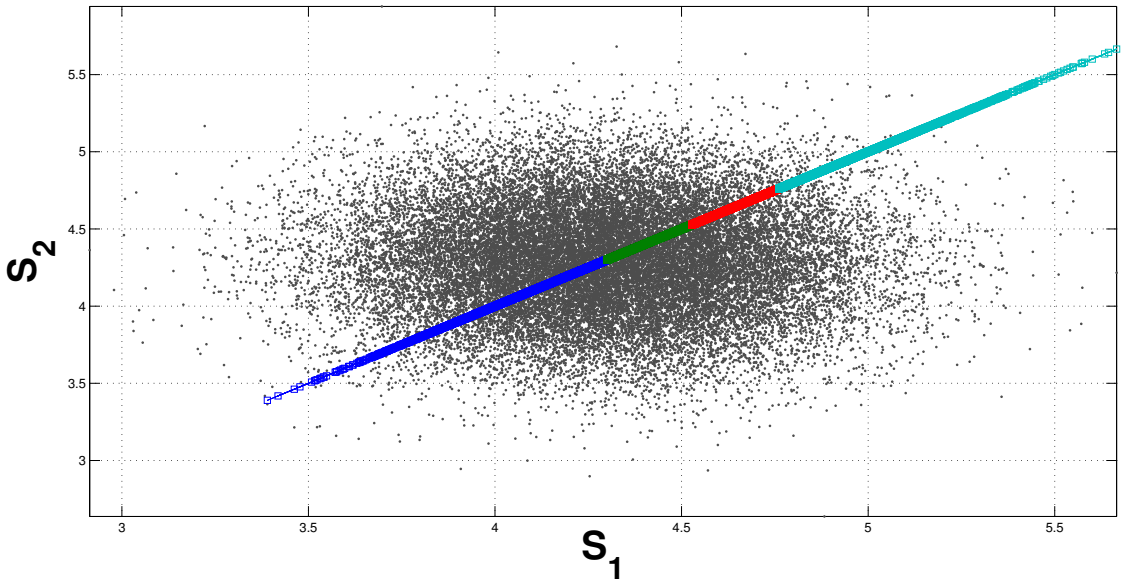
Reduced state space bundling

- Map the high-dimensional space — using payoff as the mapping function — to a reduced space.
- Perform recursive bifurcation of grid points mapped to this reduced state space.



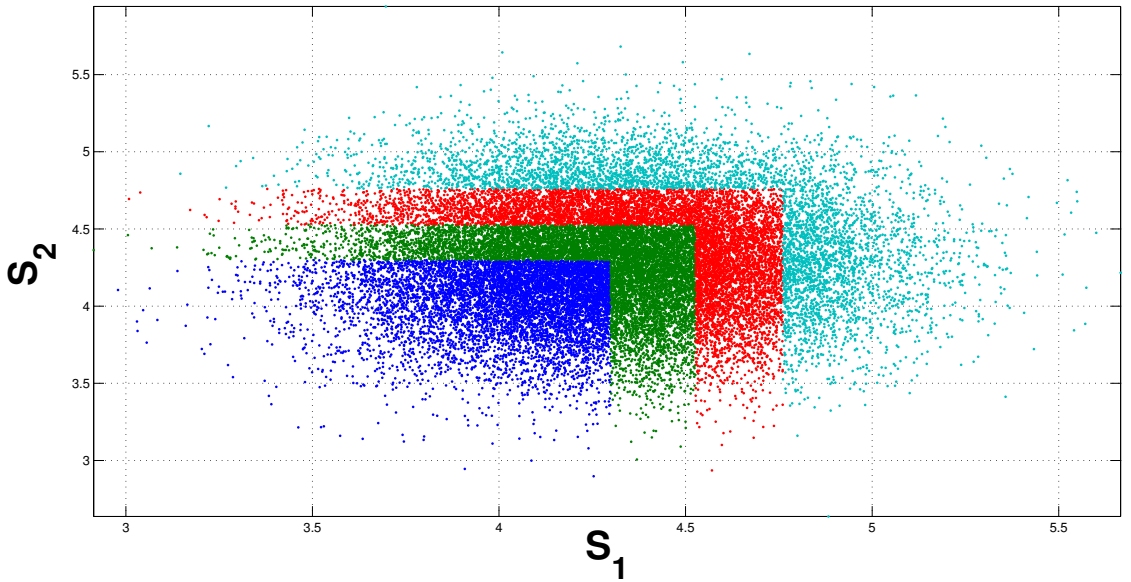
Reduced state space bundling

- Map the high-dimensional space — using payoff as the mapping function — to a reduced space.
- Perform recursive bifurcation of grid points mapped to this reduced state space.



Reduced state space bundling

- Map the high-dimensional space — using payoff as the mapping function — to a reduced space.
- Perform recursive bifurcation of grid points mapped to this reduced state space.



Parametrizing the option values

- The objective is to choose, corresponding to each bundle β at t_{m-1}, a parameter vector $\alpha_{t_m}^{\beta}$ so that,

$$v_{t_m}(\mathbf{S}_{t_m}) \approx \hat{v}(\mathbf{S}_{t_m}, \alpha_{t_m}^{\beta}),$$

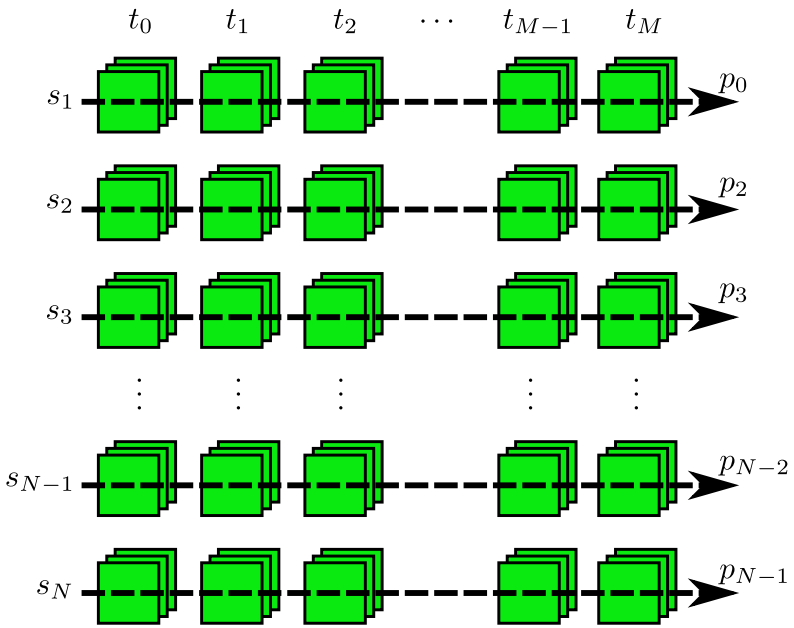
- We use OLS, to define

$$\hat{v}(\mathbf{S}_{t_m}, \widehat{\alpha}_{t_m}^{\beta}) = \sum_{k=1}^{K} \widehat{\alpha}_{t_m}^{\beta}(k)\phi_k(\mathbf{S}_{t_m}), \tag{12}$$

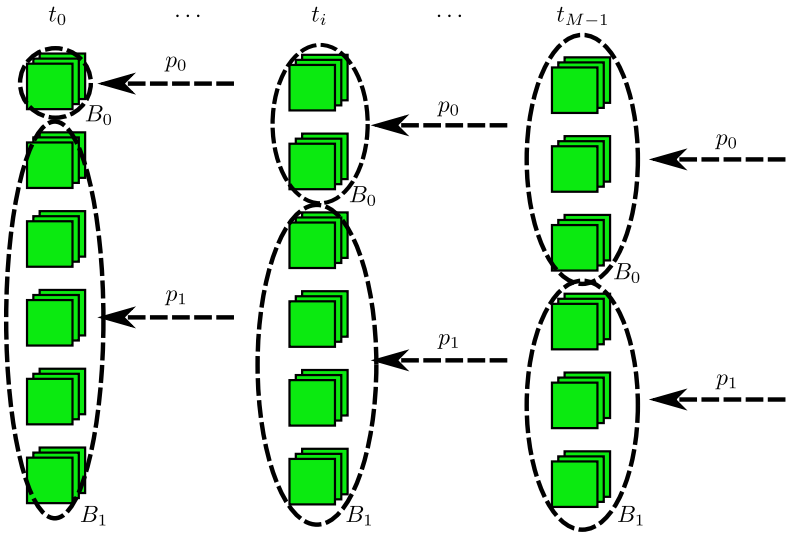
satisfying,

$$\underset{\widehat{\alpha}_{t_m}^{\beta}}{\operatorname{argmin}} \sum_{n=1}^{|\mathcal{B}_{t_{m-1}}(\beta)|} \left(v_{t_m}(\mathbf{S}_{t_m}(n)) - \sum_{k=1}^{K} \widehat{\alpha}_{t_m}^{\beta}(k)\phi_k(\mathbf{S}_{t_m}(n)) \right)^2. \tag{13}$$

Implementation scheme



(a) SGBM Monte Carlo stage



(b) SGBM bundling stage

Figure: SGBM Monte Carlo and bundling stages

Results

- Accelerator Island system of Cartesius Supercomputer.
  - Intel Xeon E5-2420 (Sandy Bridge).
  - NVIDIA Tesla K20m.
  - C-compiler: GCC 4.4.6.
  - CUDA version: 5.5.
- Geometric basket Bermudan put option: $\mathbf{S}_{t_0} = (40, \ldots, 40) \in \mathbb{R}^d$, $K = 40$, $r_t = 0.06$, $\sigma = (0.2, \ldots, 0.2) \in \mathbb{R}^d$, $\rho_{ij} = 0.25$, $T = 1$ and $M = 10$.

Table: Time (s) for a high-dimensional problem with equal-partitioning. Test configuration: $N = 2^{22}$ and $\delta t = T/M$.

	$\nu = 4096$		$\nu = 8192$		$\nu = 16384$	
	$30d$	$50d$	$30d$	$50d$	$30d$	$50d$
C	570.83	989.15	573.88	986.16	571.97	984.51
CUDA	18.06	25.09	18.43	25.42	19.25	26.06
Speedup	31.61	39.42	31.14	38.79	29.71	37.78

Conclusions

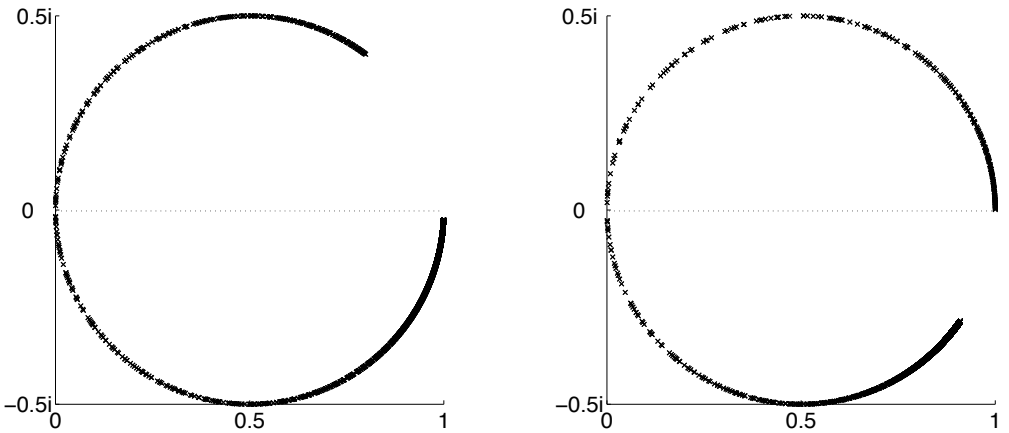
- We still extend the range of applicability of Fourier techniques
- The COS method is an efficient method, and an interesting object of study.
- Range of applicability increases with each PhD student interested (nowadays BSDEs, CVA, . . .)
- Research line in computational finance can be generalized.
- We are not (yet) PinT!

Truly shifted Laplacian preconditioner

We consider $k\,h = 0.625$ and compare both exact preconditioners:

- Shifted Laplacian (SL)
- Truly shifted Laplacian (TSL (β))

Figure: Spectral pictures of $A_h M_h^{-1}$ for both preconditioners, using $k = 80$ and $h = \frac{1}{128}$



SL TSL ($\beta = 0.3$)

Table: Number of Bi-CGSTAB iterations for different values of wave numbers

k (h)	20 (1/32)	40 (1/64)	80 (1/128)	160 (1/256)
SL	12	22	40	76
TSL ($\beta = 0.3$)	8	15	25	46

SGBM on GPU (with A. Leitao)

- NVIDIA CUDA platform.
- Two parallelization stages:
  - Forward: Monte Carlo simulation.
  - Backward: Bundles in each time step.
- Memory transfers to/from GPU: Unified Virtual Addressing (UVA) eases asynchronous transfers and page-locked memory accesses.
- Random numbers "on the fly": cuRAND library.
- Each bundle calculations (option value and regression) are made in parallel (one GPU thread per bundle).
- All threads collaborate in order to compute the continuation value.
- Final reduction (payoff): Thrust library.