GPU Acceleration of Monte Carlo simulation for Capital Markets & Insurance

Serguei Issakov, Ph.D.

Global Head of Quantitative Research & Development, Senior Vice President

GPU Technology Conference, San Jose, May 9, 2017

numerix

Agenda

- Numerix introduction
- How Monte Carlo simulation is used for pricing in finance: pricing models, financial instruments
- Functionalities in production
- Use cases
- Pricing code reorganization to run on GPU
- GPU acceleration factors for financial instruments of different complexity
- Multi-GPU scaling on DGX-1
- Nested Monte Carlo for future capital and margin projections
- Roadmap / future work

numerix

Numerix is the award-winning industry leader in risk management and quantitative analytics for capital markets participants.

Our 200+ financial engineers, developers, and implementation experts based in sixteen countries help over seven hundred customers manage their most demanding trading, risk management, and regulatory compliance challenges

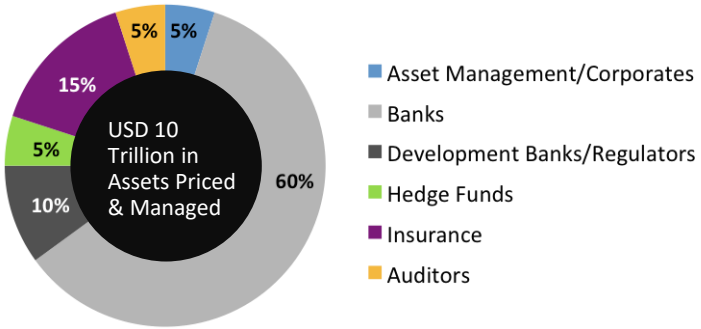
numerix

The World's Financial Institutions Rely on Numerix

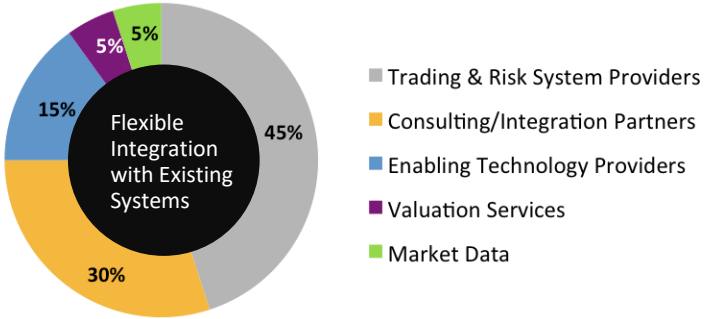
Client-focused

- Exceptional, award-winning client support
- *Celent* & *Chartis* Customer Service Awards
- Local time zone coverage/support
- Fast response time to client requests
- Agile development & flexible technology

Over 700 Global Financial Institutions



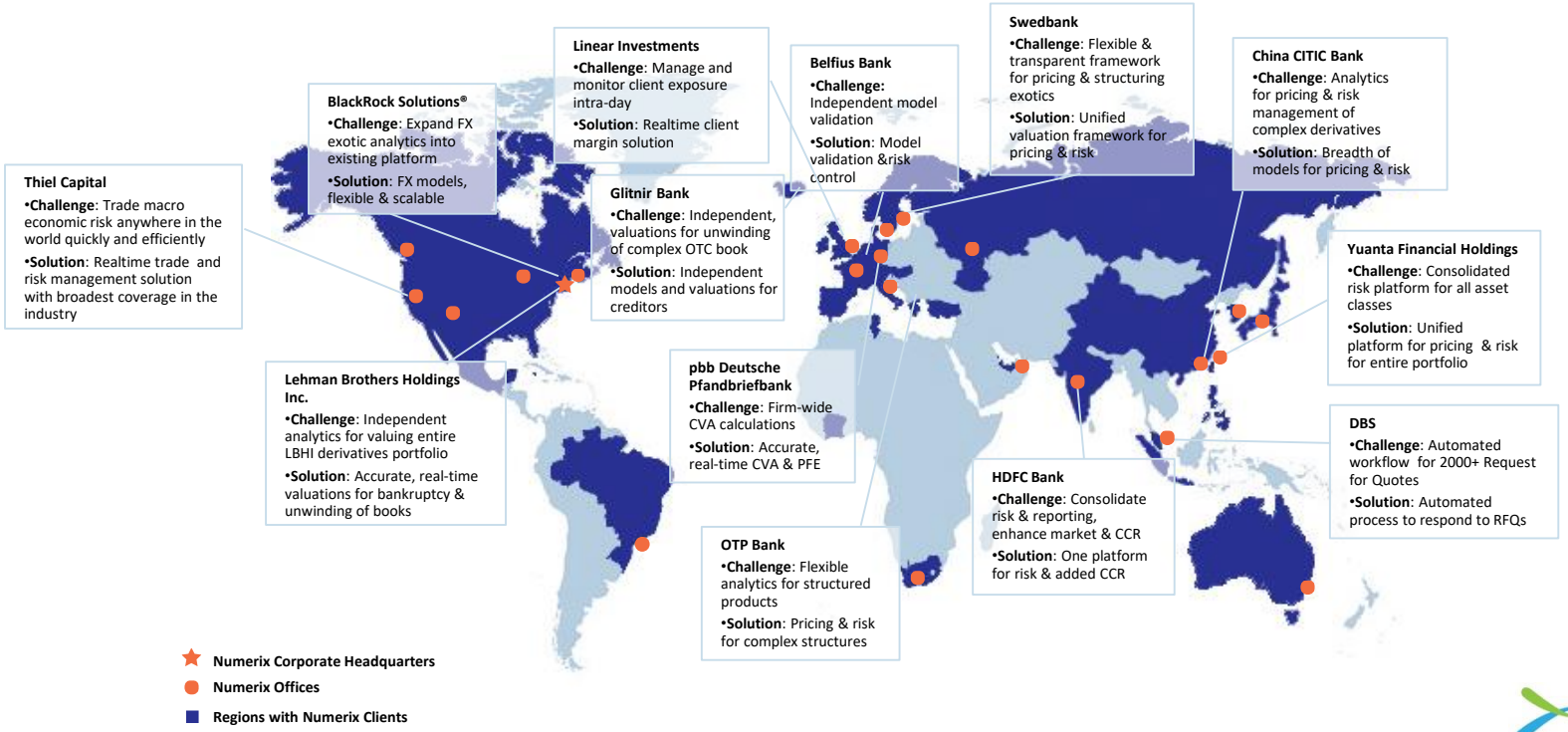
Over 90 Partners



numerix

Customer Use Cases & Global Presence

Financial institutions all over the world rely on Numerix, with 24 offices in 16 countries, to price their derivatives contracts, manage their risk exposures, and better serve their customers



numerix

Numerix Solutions

numerix

GPU support of Monte Carlo simulation at Numerix

Timeline

Aug 2016: First production release of Monte Carlo simulation on GPU for simpler trades (Capital Markets)

Nov 2016: Support of CUDA 8.0 environment required to run on the latest generation of GPUs, Pascal

Dec 2016: Added support for most complex trades (Insurance)

GPU advantages at a glance

Increased computation speed: acceleration of 20X on one GPU versus a single threaded computation on CPU

Support for running Monte Carlo simulation on multiple GPUs, with practically perfect parallelization (tested on NVIDIA DGX-1)

Allows to substantially increase the number of Monte Carlo paths for more accurate pricing and risk management

numerix

Monte Carlo simulation in finance

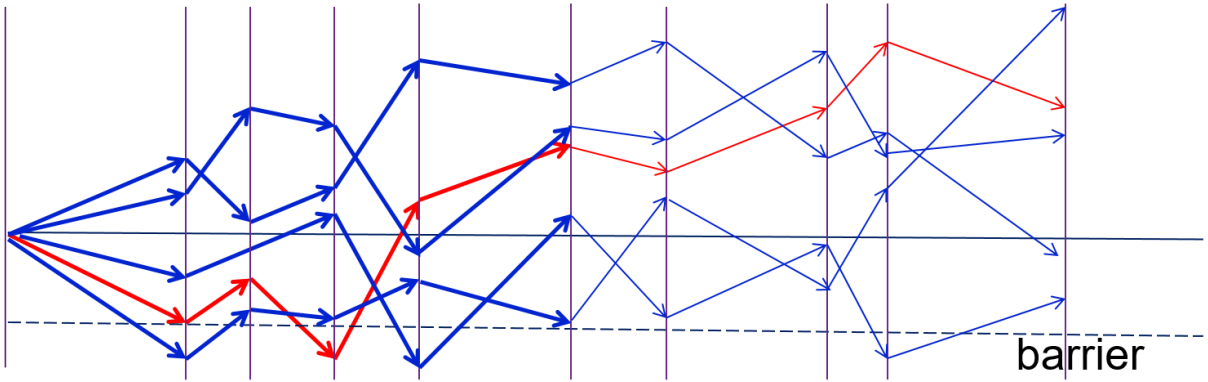
Evolution of markets follows stochastic processes

Pricing models: stochastic differential equations

Financial instruments / trades: Payoff “script” to define instrument

Monte Carlo pricing

- Generate random numbers
- Generate Monte Carlo “paths” according to model (discretization of stochastic evolution)
- Execute payoff script to compute distributions of future prices (most operations are path-wise)



numerix

Equity / Foreign Exchange Models

Black-Scholes model

$$dS_t = (r(t) - q(t))S_t dt + \sigma(t) S_t dW_t$$

$r(t)$ – short term rate, $q(t)$– continuous dividend, $\sigma(t)$– volatility (deterministic function of time)
dW_t– Brownian motion (in the risk-neutral measure)

Local Volatility (Dupire) model

$$dS_t = (r(t) - u(t))S_t dt + \sigma_{loc}(S_t, t) S_t dW_t$$

$u(t)$ – dividend curve, $\sigma_{loc}(S_t, t)$ – local vol (deterministic function of the asset level and time)

Stochastic Volatility (Heston) model

$$dS_t = (r(t) - u(t))S_t dt + S_t \sqrt{v_t} dW,$$
$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dV,$$
$$\langle dW dV \rangle = \rho dt.$$

v_t– variance, κ – mean reversion rate, θ – long-term variance, ξ – volatility of volatility.
ρ is correlation between the stochastic processes for asset level and its variance.

numerix

Payoff script example (simpler exotic trade)

Trade type: worst of down & in put for equity basket with 3 underlyings in the same currency, with continuous barrier monitoring

PAYOFFSCRIPT

```
PRODUCTS
    DISCOUNTING WO, WOKO123, WOKO123discrete, wodip, wodipSquared
    NONDISCOUNTING eq0[3], worstperf, isKo, OneAsset
    INTEGER i
END PRODUCTS

PAYOFFSCRIPT
    IF ISACTIVE(today) THEN
        isKO = 1
        Oneasset = 1
eq0[1] = 67.2
eq0[2] = 72.5
eq0[3] = 11.55
        AttachBarrier(Oneasset, EQ1, TODAY, Barrier * eq0[1], BarrierDown, 0, PayRebateAtMaturity, EXPIRY)
        AttachBarrier(Oneasset, EQ2, TODAY, Barrier * eq0[2], BarrierDown, 0, PayRebateAtMaturity, EXPIRY)
        AttachBarrier(Oneasset, EQ3, TODAY, Barrier * eq0[3], BarrierDown, 0, PayRebateAtMaturity, EXPIRY)
    END IF
    IF ISACTIVE(obsdates) THEN
        worstperf=10000
        FOR i=1 TO 3
            worstperf=min(eq[i] / eq0[i], worstperf)
        NEXT
        isKO *= STEP(worstperf-Barrier)
    END IF
    IF ISACTIVE(expiry) THEN
        WO = CASH(MAX(strike - worstperf, 0), expiry, THISPAY)
        WOKO123discrete = WO *isKO
        woko123 = WO * oneasset
        wodip = wo - woko123
        wodipSquared = wodip * wodip
    END IF
END PAYOFFSCRIPT
```

Monte Carlo pricing on GPU in production

Forward Monte Carlo simulation on GPU

Supported pricing models & model configurations

- ✓ **Equity/FX** models. H2 2016: Black-Scholes, Local Vol (Dupire)
  Q1 2017: Stochastic Vol (Heston), 'Hot start' Heston [*]
  Q2 2017: Local Stochastic Vol (LSV), Stochastic Vol with Jumps (Bates)
- ✓ **Equity/FX basket** models with above models for individual equities
- ✓ Single currency **Hybrid model** with the above models for individual equities & deterministic IR model

Random numbers & Floating point precision

- ✓ Quasi-random numbers (e.g. Sobol sequences) & pseudorandom numbers (lower memory footprint)
- ✓ Double precision/FP64 & single precision/FP32 in Monte Carlo simulation

[*] S. Mechkov, 'Hot-start' initialisation of the Heston model (2016), **RISK**, November
http://www.risk.net/risk-management/2475720/hot-start-initialisation-heston-model
Serguei Mechkov initialises Heston model's parameters using probability distributions

numerix

Trade types & GPU controls

Trade types: All trades that can be priced by Forward Monte Carlo simulation are supported on GPU

Trade complexity	# lines in payoff script	Example
Simpler exotics	30	Options on small equity baskets with barrier conditions
Structured deals of average complexity	300	FX TARF (Target Redemption Forward) allows to buy or sell foreign currency at an agreed “enhanced rate” for a number of expiry dates
Most complex structured deals	3,000	Variable Annuities

GPU controls

- User’s access to GPU card parameters, the numbers of blocks and threads, to choose optimal GPU hardware configuration
- Ability to direct simulation to a particular GPU in the multi-GPU setup

numerix

Use Cases: Monte Carlo pricing on GPU

EMEA: Swiss Private Bank

Requires high accuracy (a very large number of Monte Carlo paths) and high speed. Already running Monte Carlo simulation on GPU in production, with a simple model (Black-Scholes). Needs a more advanced Local Vol model.

Timing requirements: 1 second on a modern GPU, for pricing and greeks for an equity basket option trade, with 300,000 Monte Carlo paths and 100 timesteps.

APAC: Major Commercial Bank in East Asia

Simulation time on one CPU core: 120 min. Required time to simulate a portfolio (price and greeks): 20 seconds

Objective: optimal solution with a CPU/GPU configuration

Trade type: structured product FX TARF (Target Redemption Forward)

Models: FX Local Volatility model, FX Local Stochastic Volatility model

Americas: US Insurer

Representative portfolio/block of 60,000 policies (Variable Annuities): runtime 1.5 hours

Hybrid model with Black-Scholes equity basket models and deterministic rates

numerix

Pricing code re-organization for running on GPU

Pricing code (on CPU) is re-written / re-reorganized as a long unrolled sequence ("batch"), of tens of thousands to hundreds of thousands of short function calls.

The length is proportional to the simulation length (number of dates) and also depends on the instrument complexity.

List of functions

```
__device__ const nsSchedule::Vvvcc functions_[]= {
    Vneg, Vabs, Vexp, Vlog, Vstep,                          //self transformation
    VassignC, VplusC, VmultC, VmaxC, VpowC,                 //number r.h.s
    VassignV, VplusV, VminusV, VmultV, VdivV, VmaxV,        //vector r.h.s
    VshiftVC, VbarrierDnVCC, VbarrierUpVCC,                 //combinations
    VassignR,                                               //pseudo-random
    VmultB, VsumB,                                          //MC normalization
    ...
};
```

Registration on CPU

```
Void registerEvent(Vvvcc fun, nsFloat* _, const nsFloat* v0, const nsFloat* v1,
                   nsFloat c0, nsFloat c1, void* data);
```

results in a batch of "events"

```
Event {Vvvcc f; nsFloat *_; const nsFloat *v0, *v1; nsFloat c0, c1; void *d;};
```

prepared and executed on GPU

numerix

Examples on functions on GPU

```
// Assign Vector
__device__ void VassignV(nsFloat* _,const nsFloat* x,const nsFloat* y,
                         nsFloat a,nsFloat b,unsigned n,void* data)
{
    int step= gridDim.x*blockDim.x;
    for(int tid=threadIdx.x+blockIdx.x*blockDim.x; tid<n; tid+=step)
        _[tid]= x[tid];
}

// Initialization of pseudo-random numbers
__device__ void VassignR(nsFloat* _,const nsFloat* x,const nsFloat* y,
                         nsFloat a,nsFloat b,unsigned n,void* data)
{
    nsRandTaus* r= (nsRandTaus*)data;
    int step= gridDim.x*blockDim.x;
    for(int tid=threadIdx.x+blockIdx.x*blockDim.x; tid<n; tid+=step)
        _[tid]= normal(r[tid]);
}
```

There are functions that do averaging over paths. Done in two steps: first averaging over threads in a block, in shared memory, and then averaging over blocks.

numerix

Custom functions on GPU

```
__device__ void volsFromStates(nsFloat* vols,const nsFloat* states,const nsFloat*,
                                        nsFloat from,nsFloat to,unsigned np,void* data)
{
    const nsSimLV::LVstep& lv_step= *(nsSimLV::LVstep*)(data);
    unsigned n= lv_step.n;
    const nsFloat* ddates= lv_step.dates;
    int step= gridDim.x*blockDim.x;
    for(int tid=threadIdx.x+blockIdx.x*blockDim.x; tid<np; tid+=step)
    {
        nsFloat x= states[tid]+lv_step.dstate;
        if(n<2)
            vols[tid]= volatilityFromState(x,lv_step.maps[0].v,lv_step.maps[0].n);
        else
        {
            nsFloat vv= 0.;
            for(size_t t=0; t<n; ++t)
            {
                nsFloat v= volatilityFromState(x,lv_step.maps[t].v,lv_step.maps[t].n);
                vv+= v*v*(ddates[t+1]-ddates[t]);
            }
            vols[tid]= sqrt(vv/(ddates[n]-ddates[0]));
        }
    }
}
```

GPU Benchmarks: Equity basket options

Equity basket options with barriers. Equity basket model with Black-Scholes for individual equities

Workstation: CPU 10 cores, RAM 64GB
GPU: GeForce GTX Titan (Kepler), 2688 CUDA cores

Workstation with GeForce GTX Titan						
# Paths	CPU BATCH FP64	CPU BATCH FP32	GPU FP64	GPU FP32	GPU FP64 speedup	GPU FP32 speedup
Pseudorandom numbers						
100K	2.57	2.67	0.23	0.11	11	24
200K	5.26	3.64	0.37	0.22	14	16
300K	8.90	4.53	0.53	0.33	17	14
500K	14.22	10.50	0.78	0.54	18	19
Quasi-random numbers						
100K	2.38	2.01	0.25	0.19	9.5	11
200K	4.87	4.03	0.43	0.34	11.3	12
300K	6.75	5.85	0.65	0.48	10.4	12
500K	12.10	9.41	0.98	0.73	12.3	13

Laptop: CPU 6th gen i7 6820-HQ 2.7GHz 4 cores, RAM 16GB
GPU: Quadro M1000M, 512 CUDA cores

Laptop with Quadro M1000M						
# Paths	CPU BATCH FP64	CPU BATCH FP32	GPU FP64	GPU FP32	GPU FP64 speedup	GPU FP32 speedup
Pseudorandom numbers						
50K	1.38	1.16	0.19	0.08	7.3	21
100K	2.92	2.63	0.31	0.15	9.4	17.5
200K	6.25	5.15	0.60	0.27	12.6	19
300K	10.01	7.49	0.88	0.42	11.4	18
Quasi-random numbers						
50K	1.11	0.84	0.26	0.1	4.3	8.4
100K	2.43	1.72	0.48	0.17	5.1	10.0
200K	5.19	3.60	0.87	0.34	6.0	10.6
300K	8.10	5.69	1.28	0.55	6.3	10.3

Time in seconds

numerix

GPU Benchmarks, with 2017 optimizations

Trade type: worst of down & in put for equity basket with 3 underlyings in the same currency, with continuous barrier observation

Model: Equity basket model with 3 Black Scholes models for underlying equities

Simulation parameters: 300,000 Monte Carlo paths, 100 timesteps

Quantities computed: price plus delta, gamma, and vega for each of 3 underlying equities. Greeks are computed as central finite differences, thus requiring 13 PV computations total for price and all Greeks.

Accelerated computation of Greeks on GPU: by reusing the same random numbers for price and Greeks

Time in seconds, on one GPU				
GPU Architecture	GPU Grade	GPU Model	Time, double precision FP64	Time, single precision FP32
Pascal	Consumer	GTX GeForce 1080	2.26	0.90
Kepler	Professional	Tesla K80	2.31	1.24

numerix

Benchmarking on NVIDIA DGX-1

Multi-GPU time scaling

The execution time of 1 task on 1 GPU device is measured (in seconds) as

total_time*#GPUdevices/#CPUthreads

For perfect scaling this number should be invariant

2X 20-core Intel® Xeon® E5-2698 v4 8X NVIDIA Tesla P100			
CPU Threads	GPU Devices	Time, single precision FP32	Time, double precision FP64
1	1	0.0573	0.0820
40	1	0.0572	0.0818
1	4	0.0592	0.0852
40	4	0.0621	0.0914
80	8	0.0827	0.1375

- We are working with NVIDIA to make available the option for

Containerized Numerix applications on NVIDIA's DGX-1

numerix

GPU Benchmarks: FX TARF

Structured deal of average complexity: ~300 lines of price script

FX TARF, 20K Monte Carlo paths						
Laptop with Quadro M1000M, 512 CUDA cores, time in seconds						
Model	CPU BATCH FP64	CPU BATCH FP32	GPU FP64	GPU FP32	GPU FP64 speedup	GPU FP32 speedup
Pseudorandom numbers						
Black-Scholes	1.57	1.37	0.16	0.06	9.8	23
Local Vol	1.34	1.26	0.17	0.10	7.9	13
Quasi-random numbers						
Black-Scholes	1.45	1.00	0.26	0.11	5.6	9.1
Local Vol	1.54	1.15	0.27	0.13	5.7	12

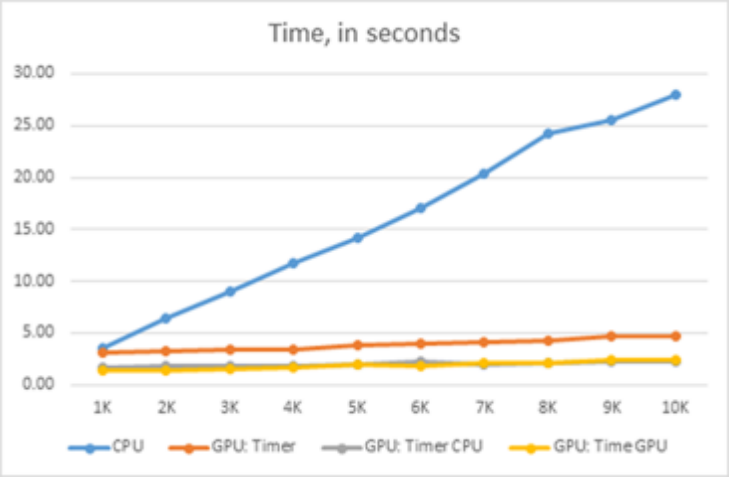
numerix

GPU benchmarks for Insurance: Variable Annuities

Most complex structured deal: ~3,000 lines of pricing (payoff) "script"

Hybrid model with Equity Black-Scholes and deterministic rates, FP64

Laptop 4 CPUs, M1000M GPU 512 cores				
# Monte Carlo paths	CPU only **Total time**	CPU + GPU: **Total time**	CPU + GPU: **CPU time**	CPU + GPU: **GPU time**
1K	3.60	3.14	1.72	1.42
2K	6.48	3.19	1.81	1.38
3K	8.94	3.35	1.80	1.55
4K	11.80	3.47	1.78	1.69
5K	14.19	3.85	1.94	1.91
6K	17.01	4.02	2.19	1.83
7K	20.37	4.13	2.02	2.11
8K	24.18	4.29	2.13	2.16
9K	25.52	4.66	2.25	2.41
10K	28.03	4.73	2.31	2.42



Double precision FP64 GPU acceleration factor: 6 for 10K paths. NVLink between CPU and GPU should help accelerate more.

- Strategy for a smaller # Monte Carlo paths:

Dynamic compilation (using CUDA PTX) of a payoff script to reuse it: (a) for Greeks, (b) for computing a block of insurance policies with the same definition and differed by parameters only

numerix

GPU to enhance business processes in Insurance

Risk-neutral (RN) pricing and greek computations for a portfolio of hedge assets and liabilities

- Stochastic RN scenarios (e.g. 50-year monthly timestep projection with 10,000 paths = ((50 * 12) +1) * 10,000) = 6,010,000 values to compute for each index in the simulation)
- Hedging for a block of Variable Annuity or Fixed Index Annuity business
- Intra-day pricing where the speed of these computations on a portfolio level is critical

Insurance Reserves & Capital (nested stochastic)

- Stochastic Real World scenarios for an outer loop (e.g. 50-year monthly timestep projection with 10,000 paths)
- Along each Real World path and timestep value hedge assets using risk-neutral pricing framework
  - Total paths required = # RW Paths * # RN Paths = 10,000 * 10,000 = 100,000,000 paths
  - Total values to compute = Total paths required * ((50 * 12) +1) = 60,100,000,000 values for each index in the simulation

Financial Planning

- Examine company financials under various planning scenarios (requires looking at reserves and capital in these various macro scenarios)
- Essentially a ‘third’ loop to run the above frameworks (triple stochastic)

numerix

Nested Monte Carlo simulation for XVA

XVA components (Valuation Adjustments)

Adjustment	Description
CVA (2002+)	Impact of counterparty credit risk
DVA (2002+)	Benefit a bank derives in the event of its own default (the 'other side' of CVA)
COLVA (2010+)	Cost of funding a collateralised derivative position, at new 'risk free' rate
FVA (2011+)	Captures the funding cost of uncollateralised derivatives above the 'risk free rate'
KVA (2015+)	Cost of holding regulatory capital as a result of the derivative position
MVA (2015+)	Cost of posting 'initial margin' against a derivative position

- MVA: cost of future initial margin

Work in progress in the industry, after new initial margin rules (in effect Sep 2016 for larger banks in US and Japan, 2017 in Europe). Expected to become a major contribution into XVA.

- KVA – cost of future capital

New FRTB (Fundamental Review of Trading Book) regulatory capital requirements (2016)

Nested Monte Carlo

Outer loop: generate Monte Carlo scenarios

Inner loop: simulate margin / capital

numerix

Numerix GPU Roadmap

Pricing Models

H2 2017

Extending support of Forward Monte Carlo simulation on GPU to

- Stochastic Interest Rate models
- Hybrid models with stochastic interest rates

2018

- Support of American Monte Carlo / Least Squares Monte Carlo on GPU
  (to price callable structured/exotic trades)

Acceleration of Risk Management & XVA for Front & Middle Office

H2 2017

- Middle office Counterparty Risk (Expected Exposures, PFE, etc.) for risk neutral & real world scenarios for simple trades
- XVA for simple trades (vanilla swaps, FX forwards), typically a majority of a portfolio

2018

- XVA for structured/callable trades

numerix

isakov@numerix.com

THANK YOU

numerix