CVA at Scale
Combining the NAG Library with dco/c++ and Origami

Event name
14th Quant Finance Conference
26th - 28th September 2018

Justin Ware
XVA calculations – computational challenge

- Generating XVA in reasonable time for the business.
  - Number of trades encompasses the entire institution, not just a desk’s book.
  - Intraday as positions and markets update.

- Approaches practitioners use.
  - Compute resources: grid / cloud / GPU.
  - Sophisticated algorithms.

- XVA calculations have stages that are highly data parallel and computationally intensive.
  - Ideal for splitting into tasks on a grid / cloud.
NAG have developed CVA demonstrator
- To demonstrate how to use tools, not how to model CVA.
- G2++ for interest rate, CIR++ for default intensity.
- Attempt to have a realistic example.

Calibrates to (and generates derivatives w.r.t) market instruments.
- OIS, (L)IBOR cash / FRAs / swaps, CDS, Swaption vols.

Variety of numerical methods.
- Root finders, quadrature and optimisation for calibration.
- Longstaff-Schwartz American Monte-Carlo for conditional expectation of trade PVs.
Which tools?

Numerical Software
- Quality (through rigorous testing) and performance are key.
- NAG numerical libraries

Algorithmic Differentiation (AD)
- XVA on its own isn’t enough – need sensitivities (derivatives).
- AD faster and more accurate than finite differencing.
- dco/c++

Grid / Cloud
- Exploit parallelism in XVA calculations.
- Origami – in partnership with Xi-FINTIQ
Adjoint Algorithmic Differentiation – very briefly!

inputs

C++

result(s)

interpt_adjoint

derivatives w.r.t. inputs

RECORDING

PLAYBACK
Challenges in using AD

- How much memory will it need?
  - Derivatives of CVA, 50 swaps, 2000 paths: 2.4 Gb (60 Mb without)

- What if it isn’t single executable / core?
  - Multi-threaded?
  - Multi-process (grid / cloud)?

- What if the application has parts you can’t use AD on?
  - e.g. closed-source library (NAG libraries).
Using too much memory? Checkpoint!

- Bigger the calculation the more memory AD will need.
  - Not influenced by complexity / sophistication, just size.

- As with performance, memory usage *hot-spots*.
  - AD software should help you look for this (e.g. tape size)

- Look for rapid growth in memory versus fewer inputs /outputs.
  - *Checkpointing* more likely to be effective here.
Making and filling a gap
Checkpointing – does it work?

- Yes and it’s essential for large scale calculations.
  - Can now use *workable* amount of memory.
  - Simple change is to checkpoint linear regression for AMC: 2.4Gb to 1.3Gb

- It isn’t free!
  - Repeated calculations, serialization overheads.
  - Above example slows from 35.8s to 37.4s.

- Software design implications.
  - Temporary objects destructed before playback?
  - Are objects in the same state during playback?

- dco/c++ is excellent at supporting checkpointing.
AD on the grid / cloud

» Before AD, split CVA calculation into (parallel) tasks.
  • Results pass between tasks; final result is CVA.

» Adjoint AD is the back-propagation of derivatives.

» Split adjoint calculation into (parallel) tasks.
  • Use AAD (dco/c++) in each task to back-propagate derivatives.
  • Derivatives pass between tasks; final results are derivatives w.r.t. market instruments.
  • Tasks have reverse order of dependency.

» Don’t break the chain rule!
  • Ensure all derivative contributions are back-propagated.
AD, grid / cloud, CVA

crc

CDS

batch1

OIS, Libor, Swopt Vols

batch2

irc

batchN

cva

derivatives w.r.t. CDS

crc

derivatives w.r.t. OIS, Libor, Swopt Vols

batch1

batch2

batchN

irc
Directed Acyclic Graphs

- Tasks and their dependencies can become complex.
- Directed acyclic graphs (DAG) have proven extremely useful in HPC.
- DAG *implicitly* describes the parallelism which grid software exploits.
- DAGs are platform agnostic.
  - Local / remote
  - CPU / GPU
Origami – a DAG execution engine

- Supports DAGs including: data transfers, metrics, recovery from task failure.
- Can run on dev machines, in-house grid, production cloud, CPU, CPU / GPU – mixture.
AD and closed-source

- Don’t always need adjoint.
  - Implicit function theorem.

- Numerical software provides AD versions
  - NAG libraries have a subset of routines with tangent and adjoint versions.
  - Seamless integration with dco/c++.
Calculate CVA and derivatives w.r.t. EONIA, EURIBOR 6M, EURIBOR 6M Swaptions, CDS, recovery rate, model params and correlations: 94 in total.

One netting set of 30k swaps and 20k Bermudan swaptions; 2000 paths.

<table>
<thead>
<tr>
<th>Description</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single core native CVA only</td>
<td>8m 52s</td>
</tr>
<tr>
<td>Single core and finite difference</td>
<td>13h 53m 28s (projected)</td>
</tr>
<tr>
<td>Origami (4 vms x 4 cores) and AD</td>
<td>29m 25s</td>
</tr>
</tbody>
</table>

Room for improvement e.g. implicit function theorem for adjoint of calibration
Efficient risk management and capital allocation requires fast XVA and sensitivities.

Massive computational challenge.

Cloud/grid + DAG + (A)AD (+ GPU) can deliver scalable performance.

Requires correct tools that can be integrated.

NAG has the tools and expertise to do this.
References

- (lots of others)

Acknowledgements

- Ian Hotchkiss, Xi-FINTIQ Ltd
- Jacques Du Toit and Viktor Mosenkis, NAG
Experts in High Performance Computing, Algorithms and Numerical Software Engineering

www.nag.com | blog.nag.com | @NAGtalk