Faster Risk Calculation: Next Generation dco/c++

Johannes Lotz (NAG) and Uwe Naumann (RWTH Aachen University and NAG)

Trading Desks can hedge auspiciously and gain a competitive edge in the market as our mathematical algorithm technology, dco/c++, gives them rich, cheap and accurate intra-day risk. This fast and accurate risk data at lower cost means more profits for traders and for the business.

AD Myths

Debunked

Automatic Differentiation (AD) and NAG's Portfolio

For an implementation of a multi-variate scalar function, i.e.,

$$y = f(\mathbf{x}), \quad \mathbf{x} \in I\!R^n, \text{ and } y \in I\!R,$$

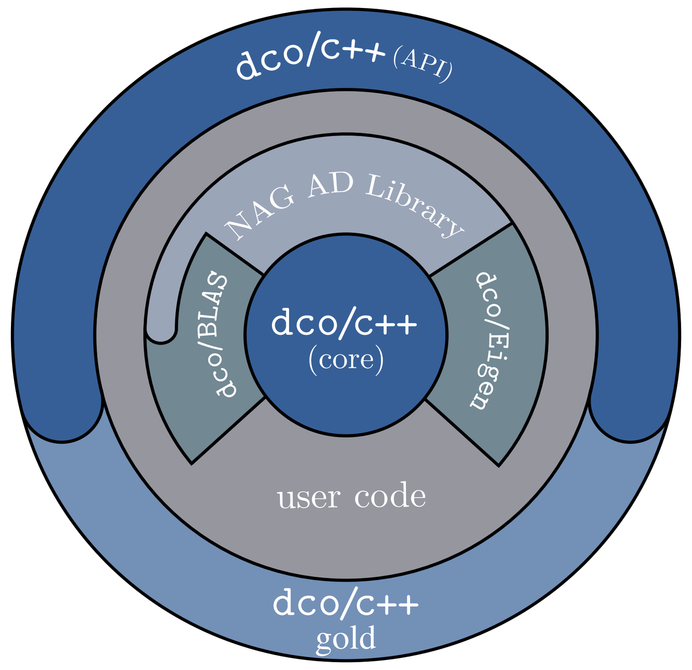
Automatic Differentiation (AD) computes exact and efficient first and higher derivatives.

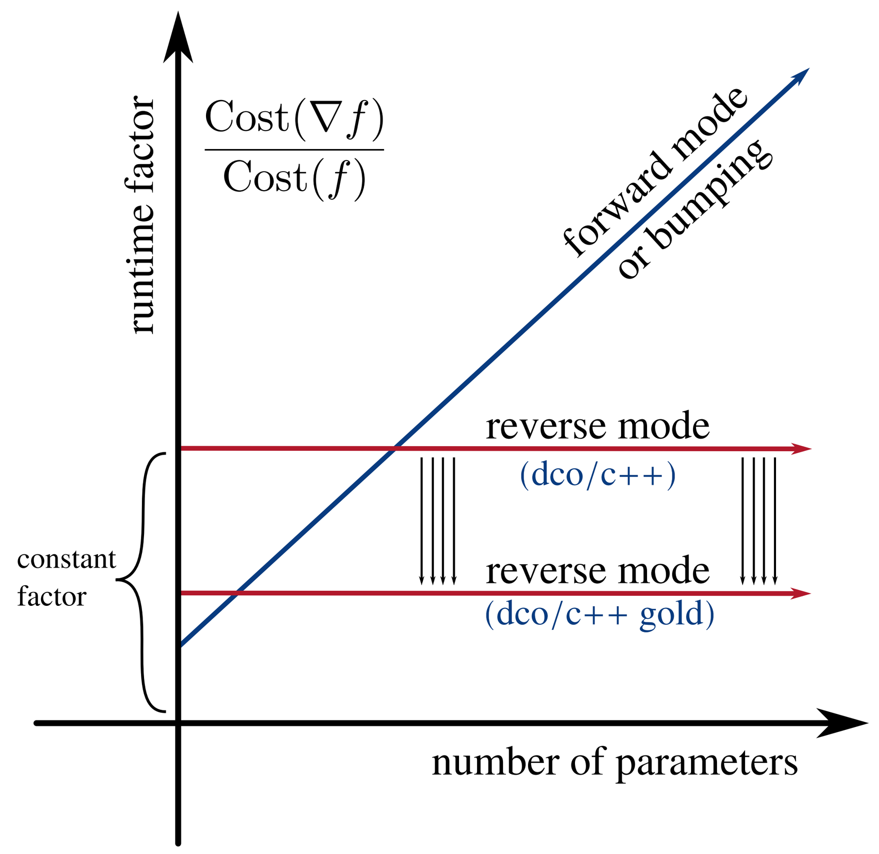
Forward Mode: $\text{Cost}(\nabla f) = \mathcal{O}(n) \cdot \text{Cost}(f)$

Reverse Mode (AAD): $\text{Cost}(\nabla f) = \mathcal{O}(1) \cdot \text{Cost}(f)$

NAG's AD

Solutions

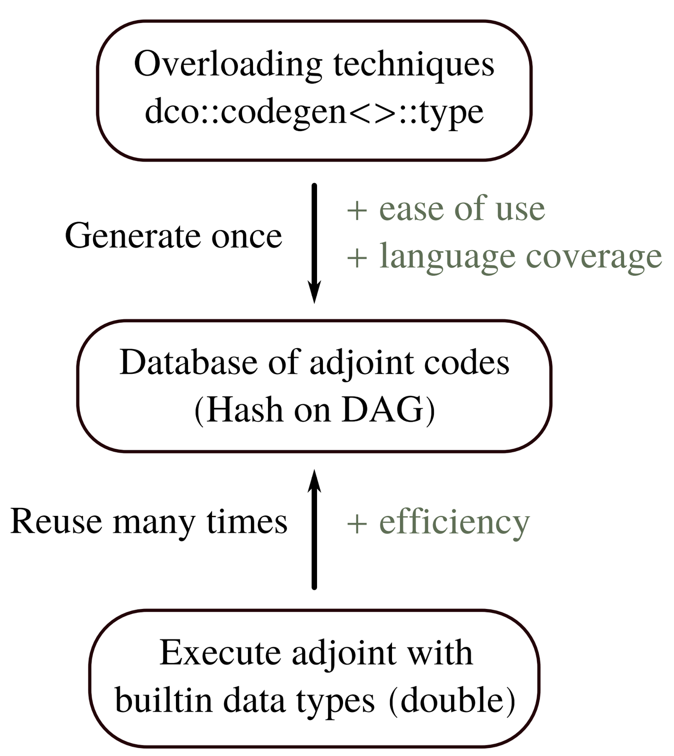




Increase Performance and Reduce Memory Use

dco/c++ provides low cost, accurate sensitivities computed 10x to 6000x faster than alternative methods, all whilst reducing memory usage. At NAG, research goes hand-in-hand with professional software development.

Code Generation



- Supports branches.
- Seamless integration with dco/c++.
- Supports just-in-time compilation.

```
auto cg =
  dco::generate(f,
          dco::in(x),
          dco::out(y)
          );

cg.adjoint(px, ax,
           py, ay);
cg.primal(px, py);
```

Parallel Taping

- Tape recording is the bottleneck (see on the right).
- We use OpenMP during tape recording.
- Each thread records a chunk of the tape.

Dedicated Adjoint Vector

- The tape holds memory for:
  a) statement-level gradients (sequential access),
  b) the vector of adjoints (random access).
- The gradients can be written to disk with reasonable runtime hit. The vector of adjoints can not!
- The *dedicated adjoint vector* reduces the required size of the vector of adjoints to fit it into memory.

Vectorization

- Better performing explicit use of **vector intrinsics** instead of relying on the compiler's auto-vectorizer.
- Biggest **benefit for dco/c++ vector modes** when computing Jacobians and Hessians.

Case Study: Monte Carlo Simulation

As case study we take a simple SDE-based European option pricer using Monte Carlo sampling.

- Top-level adjoint code:

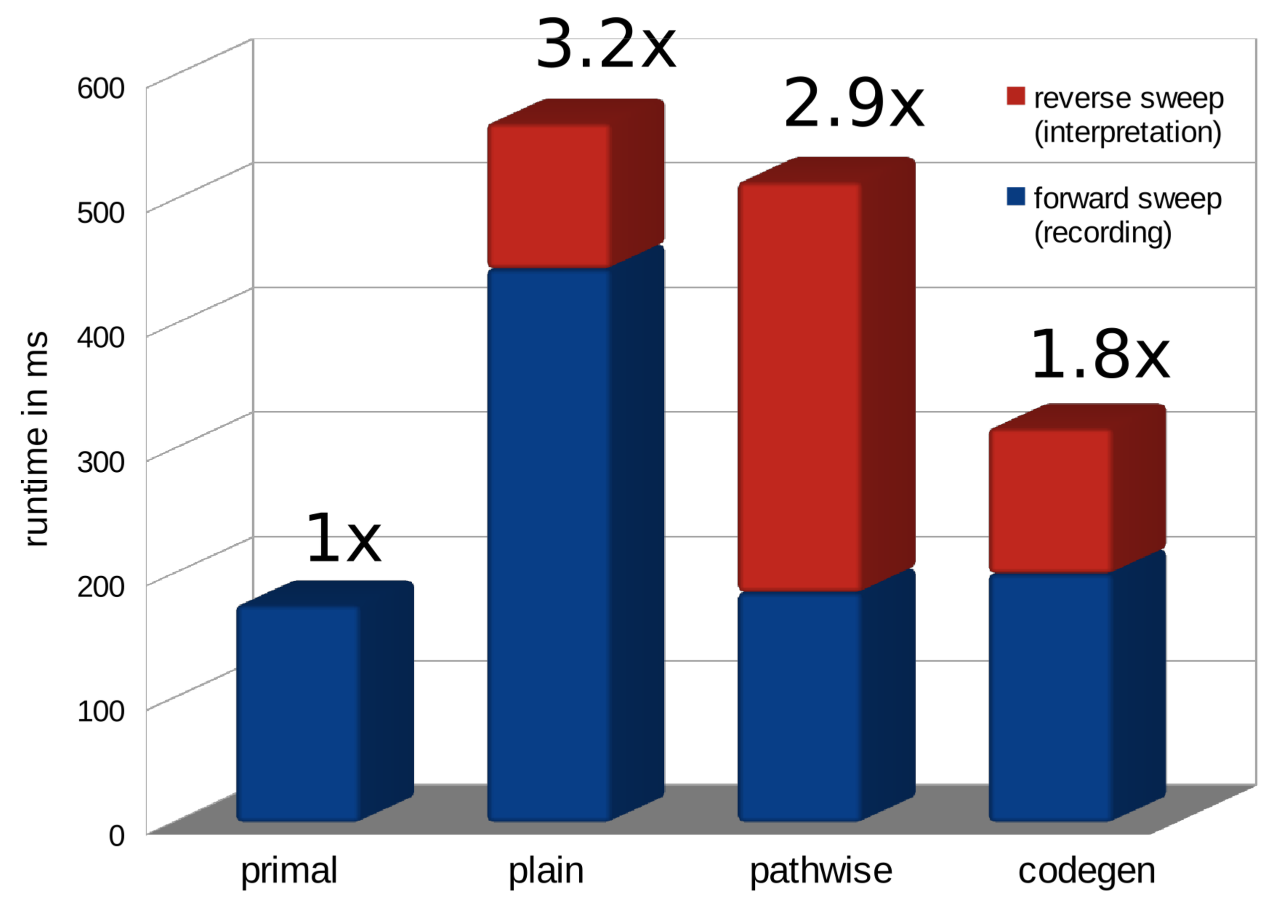
```
using mode_t = dco::ga1s<double>;
using type   = mode_t::type;

mc::active_inputs<type> X(S0, r, K, T, vols);
mc::passive_inputs XP(N, M);
mc::active_outputs<type> Y;
mc::passive_outputs YP;

auto pricer = [&](auto const& X, auto & Y)
   { mc::price(X, XP, Y, YP); };

auto jac = dco::jacobian(pricer,
                         dco::in(X),
                         dco::out(Y));

std::cout << "Y     = " << Y.V << "\n";
std::cout << "dY/dX = " << jac << "\n";

Y     = 32.6944
dY/dX = 0.982097 [...] -1.04929
```

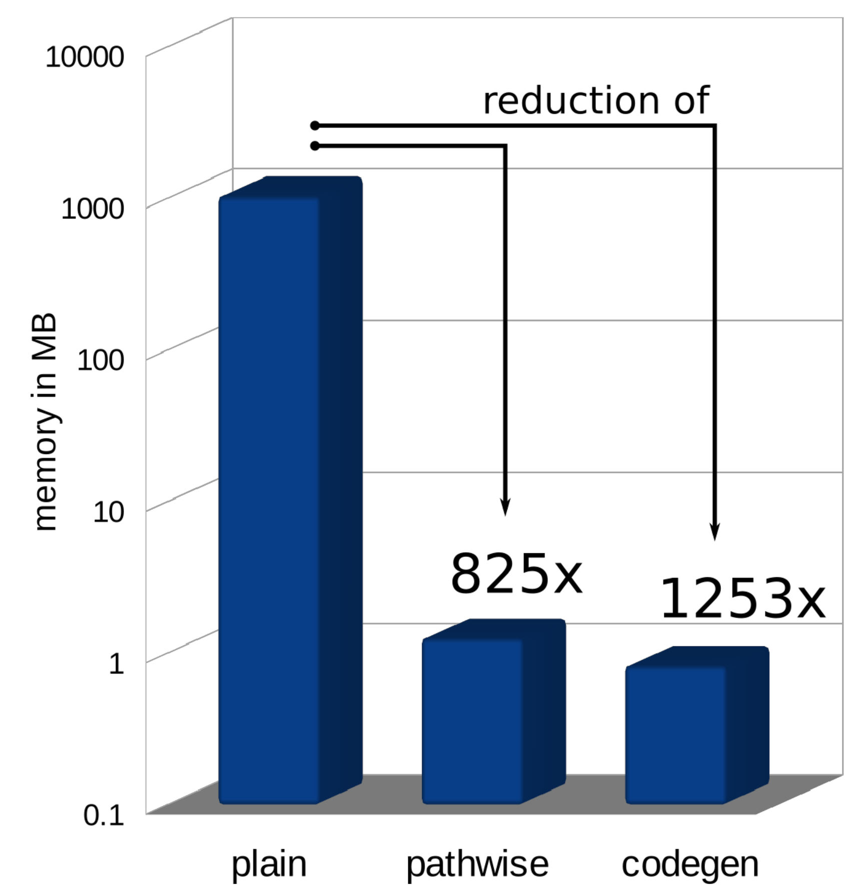
- Monte Carlo core:

```
//** mcpath(X, XP, Z)
  type mcdt = X.T / (XP.M-1);
  type logS = log(X.S0), t = 0;

  for(int i = 0; i < XP.M; ++i, t += mcdt) {
    type volS = sqrt(X.sigmaSq(logS, t));
    logS += (X.r-0.5*volS*volS)*mcdt
         + volS*sqrt(mcdt)*Z[i];
  }
  type ST = exp(logS);
  return dco::condition(ST < X.K, 0,
                        exp(-X.r*X.T)*(ST-X.K));

//** price(X, XP, Y, YP)
  std::vector<double> Z(XP.M);
  for(int p = 0; p < XP.N; p++) {
    randNormal(XP.M, XP.rngseed, Z);
    Y.V += mcpath(X, XP, Z);
  }
  Y.V = Y.V / XP.N;
```

Performance and Memory Use

For computing the gradient of above example code, we compare plain dco/c++ with pathwise adjoints and code generation approaches ⇒ 10k Monte Carlo paths and 360 Euler steps in each path.





- Parallel taping makes use of unused, idling cores during tape recording. **Speedup up to 4x for 32 cores**.
- *Dedicated adjoint vector* **reduces the required memory by factors of 10 to 1M** (depending on structure).
- Explicit vectorization **increases performance 2x (AVX2) to 4x (AVX-512)** compared to the auto-vectorizer.

dco/c++ gold: Achieve significant performance increase and reduction in memory use.