Convex Relaxations of Options & Greeks w/ AD

Neil Kichler, M.Sc. RWTH Aachen

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

Problem: Uncertainty

Computation of sensitivities (*the Greeks*) in option pricing is based on parameters being fixed.
↪ Uncertainty in parameters not accounted for.

Key Idea

Combine Algorithmic Differentiation (AD) [5] with Relaxation techniques (McCormick relaxations [3, 4]).
↪ Allows reuse of existing generic numerical programs and evaluates it via operator overloading.

Insights:

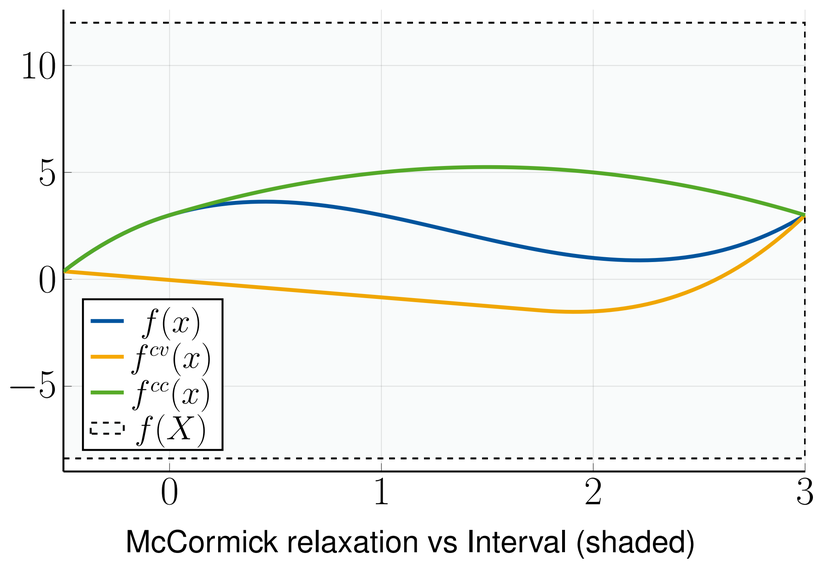
- Obtain information over entire intervals [2].
- McCormick relaxations provide much tighter convex relaxations compared to interval arithmetic.

Examples:

- Convex relaxation of Call Price, Δ, Θ, etc.
- Delta with uncertain S_0, r, ξ, κ, etc.

McCormick Relaxations

Relaxations are typically used in deterministic global optimization to bound non-convex functions [1].



McCormick relaxation vs Interval (shaded)

Technical Setup

Implemented as open-source CUDA C++ libraries:

CuInterval, CuMcCormick, CuTangent.

Combinations thereof result in desired relaxations of sensitivity information.

github.com/neilkichler/<cuinterval, cumccormick, cutangent>

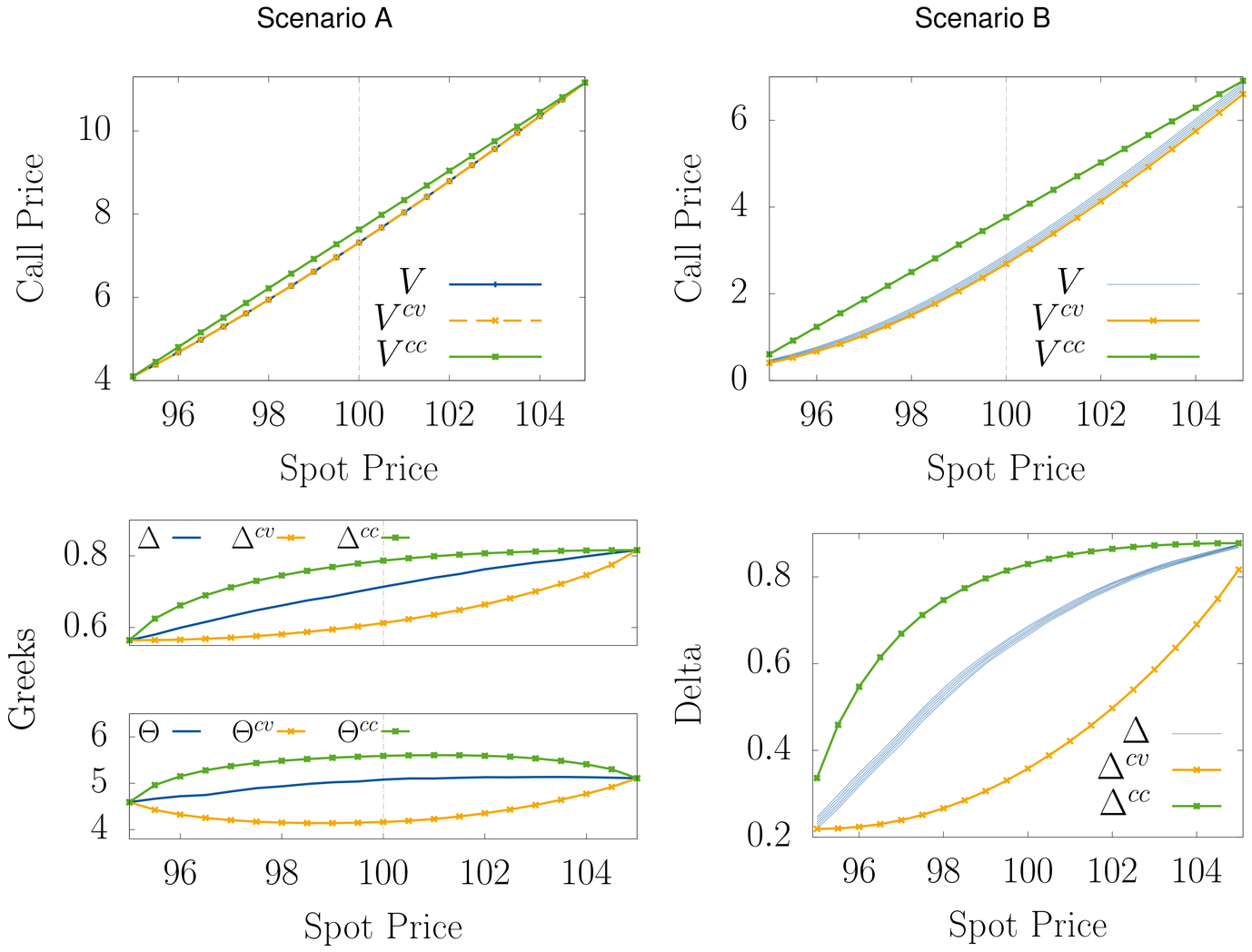
References

1. J. Deussen and U. Naumann. “Subdomain separability in global optimization”. *Journal of Global Optimization* 3, 2023, pp. 573–588.
2. N. Kichler et al. “Towards Sobolev Pruning”. *PASC ’24*. 2024, pp. 1–11.
3. G. P. McCormick. “Computability of global solutions to factorable nonconvex programs: Part I”. *Mathematical programming* 1, 1976, pp. 147–175.
4. A. Mitsos et al. “McCormick-Based Relaxations of Algorithms”. *SIAM Journal on Optimization* 2, 2009, pp. 573–601.
5. U. Naumann. *The art of differentiating computer programs.* SIAM, 2011.

Sample Code

```
#include <cumccormick/cumccormick.cuh>
#include <cutangent/cutangent.cuh>
...
using T = cu::tangent<cu::mccormick<double>>;
heston::parameters<T> xs{}; T res[n];
// dummy scenario
xs[i].r = 0.0319; xs[i].tau = 1.0; xs[i].K = 95.0; ...
// specify input bounds and point where to obtain relaxation
value(xs[i].S0) = { .lb = 95.0, .cv = 100.0,
                    .cc = 100.0, .ub = 105.0 };
// seeding to compute derivative w.r.t. S0 (i.e., Delta)
derivative(xs[i].S0) = 1.0;
heston::parameters<T> *d_xs; T *d_res;
cudaMalloc(&d_xs, n * sizeof(*xs));
cudaMalloc(&d_res, n * sizeof(*res));
cudaMemcpy(d_xs, xs, n * sizeof(*xs), cudaMemcpyHostToDevice);
your_pricing_kernel<<<n, 1>>>(d_xs, d_res, n); // unaltered
cudaMemcpy(res, d_res, n * sizeof(*res), cudaMemcpyDeviceToHost);
std::cout << "Price & Delta: " << *res << std::endl;
...
```

Results for Heston Monte Carlo



Using: $T = 1, \quad r = 0.04$ $\qquad$ $T = \frac{3}{12}, \quad r \in [0.0325, 0.045]$

$K = 100, \quad \rho = -0.7, \quad \kappa = 6.21, \quad \theta = 0.019, \quad \xi = 0.61, \quad v_0 = 0.010201$

Future Directions

Many opportunities:

- Applicability to other methods for option pricing:
  PDEs via finite-differences, semi-closed form, etc.
- Exotic options.

Beyond option pricing:

- Non-convex deterministic global optimization in finance?
- Quant applications for sensitivities of relaxations?
  ↪ i.e., using `cu::mccormick<cu::tangent<T>>`.

Poster online:

Contact: kichler@stce.rwth-aachen.de Website: stce.rwth-aachen.de