Algorithmic Differentiation of Nonsmooth and Discontinuous Functions

Jonathan Hüser and Uwe Naumann
Software and Tools for Computational Engineering
RWTH Aachen University

Abstract

Adjoint algorithmic differentiation (AAD) is exact up to machine precision and does not capture sensitivity to nearby nonsmoothness or discontinuities. Smoothing the indicator function produces a regularization effect similar to bumping the input while maintaining the efficiency of AAD. The operator-overloading tool dco/c++ supports smoothing through extensible adjoint code patterns for nonsmooth and discontinuous functions.

Adjoint Algorithmic Differentiation

- For \( g = f(x) \) with \( f : \mathbb{R}^n \to \mathbb{R} \), efficiently compute \( \frac{\partial g}{\partial x} \)
- AAD bumping
- dco/c++ is an operator-overloading AAD tool
- partial derivatives are stored in tape during computation
- no separate maintenance of primal and derivative code required

Nonsmooth and Discontinuous Functions

Examples

- Vanilla call payoff (nonsmooth)
  \[ P(S, K) = \begin{cases} S - K & \text{if } S > K \\ 0 & \text{otherwise.} \end{cases} \]
- Digital payoff (discontinuous)
  \[ P(S, K) = \begin{cases} 100 & \text{if } S > K \\ 0 & \text{otherwise.} \end{cases} \]

- Nonsmooth and discontinuous functions usually piecewise defined
- \( f(x) = \sum_{i=1}^{N} \alpha_i x_i \) can be challenging (e.g. Monte Carlo)
- Derivatives near \( y(x) = 0 \) can be challenging
- Other methods for nonsmooth and discontinuous functions
- Differentiable univariate quadrature (for stochastic case)
- Direct evaluation of piecewise linearization

Barrier Option Monte Carlo (Case Study)

- Payoff for discretized path given in stochastic INF
  \[ P(S_0, B, r, \sigma, \mathcal{Z}) = \sum_{i=1}^{N} I[B = 0; S_i - 0; S_i - K > 0] \cdot (S_i - K) \]
  with \( S_i = S_i(t_i, r, \sigma, \mathcal{Z}) \)

Function Regularization

- Replace indicator by continuous/smooth approximation
- Bandwidth parameter \( \delta \) controls approximation error (bias)

Uniform Distribution (Call Spread)

\[ \frac{\partial g}{\partial \delta} | g > 0 \approx \frac{1}{\sqrt{\delta}} \exp \left( -\frac{g^2}{2\delta} \right) \]

Example:

Euler-Maruyama Path with dco/c++

- Project sensitivity \( \nabla_{\lambda_i} A_i / Q_i \) for different \( \delta \)

Regularized Projected Monte Carlo Delta Estimator

- \( \delta \) with 20080 samples, \( N = 1000 \)
- \( \delta \) uniform: 0.2345 0.2398 0.2397
- \( \delta \) normal: 0.2345 0.2398 0.2397
- Low probability events \( \rightarrow \) Variance reduction still necessary

Nearest Correlation Matrix (Case Study)

- Pairwise correlations can be inconsistent
- Find nearest \( n \times n \) correlation matrix that is positive definite
- \( A \) projection \( \lambda_i = \max(0, \lambda_i) \) of eigenvalues

- Convex optimization problem \( \rightarrow \) use implicit function theorem

Adjoint Code Patterns in dco/c++

- Test matrix positive definite but numerically singular
- NCM algorithms maps \( A \) to itself
- Regularization gives sensitivity on implicit function theorem
- Projection sensitivity \( \nabla_{\lambda_i} A_i / Q_i \) for different \( \delta \)

Conclusions

- Monte Carlo sensitivities for discontinuous payoffs
- Smoothing of auxiliary functions and implicit function theorem
- Local sensitivity enables bandwidth calibration
- dco/c++ supports extensible adjoint code patterns for regularization

Contact Information:
LuFG Informatik 12, STCE
RWTH Aachen University D-52056 Aachen, Germany
Phone: +49 (0)241 80 29224
Email: husser@stce.rwth-aachen.de