nag

www.nag.com

Exact First- and Second-Order Greeks by Algorithmic Differentiation

The Numerical Algorithms Group (NAG) work very closely with Uwe Naumann[1] to help users take advantage of Algorithmic Differentiation methods.

Abstract

Algorithmic (also known as Automatic) differentiation (AD) is a method for computing sensitivities of outputs of numerical programs with respect to its inputs both accurately (to machine precision) and efficiently. The two basic modes of AD – forward and reverse – and combinations thereof yield products of a vector with the Jacobian, its transpose, or the Hessian, respectively.

Numerical simulation plays a central role in computational finance as well as computational science and engineering. Gradients, (projected) Jacobians, (projected) Hessians or even higher-order sensitivities are required in order to make the highly desirable transition from pure simulation to optimization of the numerical model or its parameters. Such quantities can be computed to machine accuracy by AD [5] as demonstrated by a large number of successful applications [1, 2, 3].

AD deals with implementations of multivariate nonlinear vector functions $F : \mathbb{R}^n \to \mathbb{R}^m$ as computer programs. Let $\mathbf{y} = F(\mathbf{x})$. Denote the Jacobian by $F' \equiv F'(\mathbf{x})$ and the Hessian by $F'' \equiv F''(\mathbf{x})$. The forward mode of AD transforms F into the tangent-linear model

$$\dot{F}(\underset{\downarrow}{\mathbf{x}}, \underset{\downarrow}{\dot{\mathbf{x}}}, \underset{\downarrow}{\dot{\mathbf{y}}}) \quad \text{where} \quad \dot{\mathbf{y}} = F' \cdot \dot{\mathbf{x}} \quad .$$

An overset downarrow marks an input. Outputs are marked with an underset downarrow. The columns of F' can be computed by letting $\dot{\mathbf{x}}$ range over the Cartesian basis vectors in $\mathbb{R}^n$. The computational complexity of accumulating the whole Jacobian using a tangent-linear model is of the same order $O(n)$ as that of numerical approximation using finite difference quotients ("bumping"). A single directional first-order sensitivity (e.g. a projection of Delta in some direction) can be obtained at roughly twice to three times the cost of evaluating F.

The reverse mode of AD transforms F into the adjoint model

$$\bar{F}(\underset{\downarrow}{\mathbf{x}}, \underset{\downarrow}{\bar{\mathbf{x}}}, \underset{\downarrow}{\bar{\mathbf{y}}}) \quad \text{where} \quad \bar{\mathbf{x}} = \bar{\mathbf{x}} + \bar{\mathbf{y}} \cdot F' \quad .$$

[1] LuFG Informatik 12: Software and Tools for Computational Engineering (STCE), Department of Computer Science, RWTH Aachen University, 52056 Aachen, Germany

F' is accumulated by setting $\bar{\mathbf{x}} = 0$ on input and by letting $\bar{\mathbf{y}}$ range over the Cartesian basis vectors in $\mathbb{R}^m$. Gradients of scalar functions can be obtained at a constant[2] multiple of the computational complexity of F. The ability to compute gradients cheaply is essential for sensitivity-based methods for large-scale nonlinear optimization ($n \gg 1$). Otherwise, the previously mentioned transition from simulation to optimization is impossible for most relevant real-world problems.

Second-order Greeks (e.g. Gamma) can be computed by any of the four combinations of forward and reverse modes of AD. For notational simplicity we assume that $m = 1$ thus avoiding the definition of products involving the Hessian tensor. The function value y and all its total sensitivities become scalars denoted by non-bold symbols. Forward-over-reverse mode is a preferred option for Hessian-vector products. It transforms F into the second-order adjoint model

$$\overset{\downarrow}{\dot{F}}(\overset{\downarrow}{\mathbf{x}}, \overset{\downarrow}{\dot{\mathbf{x}}}, \underset{\downarrow}{\overset{\downarrow}{\bar{\mathbf{x}}}}, \overset{\downarrow}{\bar{y}}, \overset{\downarrow}{\dot{\bar{y}}}) \quad \text{where} \quad \dot{\bar{\mathbf{x}}} = \dot{\bar{\mathbf{x}}} + \dot{\bar{y}} * F'(\mathbf{x}) + \bar{y} * F''(\mathbf{x}) * \dot{\mathbf{x}} \quad .$$

Products of the Hessian with a vector $\dot{\mathbf{x}}$ can be computed by setting $\dot{\bar{\mathbf{x}}} = \mathbf{0}$, $\dot{\bar{y}} = 0$, and $\bar{y} = 1$ at roughly twice the cost of running the adjoint code. Alternatively, reverse-over-forward mode produces the same result at a similar computational cost.

If reverse mode is not available, then second-order sensitivity information must be computed by applying forward mode to the tangent-linear model of F leading to the second-order tangent-linear model

$$\overset{\sim}{\tilde{F}}(\overset{\downarrow}{\mathbf{x}}, \overset{\downarrow}{\dot{\mathbf{x}}}, \overset{\downarrow}{\tilde{\mathbf{x}}}, \underset{\downarrow}{\tilde{y}}) \quad \text{where} \quad \tilde{y} = \dot{\mathbf{x}}^T \cdot F''(\mathbf{x}) \cdot \tilde{\mathbf{x}} + F'(\mathbf{x}) \cdot \tilde{\dot{\mathbf{x}}} \quad .$$

The Hessian is accumulated entry-by-entry by setting $\tilde{\dot{\mathbf{x}}} = \mathbf{0}$ and initializing $\dot{\mathbf{x}}^T$ and $\tilde{\mathbf{x}}$ to the corresponding Cartesian basis vectors in $\mathbb{R}^n$. This approach yields a computational complexity of $O(n^2)$, which is likely to be prohibitive in practical situations.

Robust and efficient implementations of AD are crucial ingredients of state-of-the-art numerical simulation methods. The use of this technology will shorten the simulation software development cycle, increase both efficiency and robustness of the numerical schemes, and improve long-term maintainability of the simulation codes.

Three reasons for using AD. In certain cases the approximation of sensitivities by finite differences may be satisfactory. However there are at least the following three situations where AD is the only feasible approach to computing Greeks.

1. The inaccuracy of finite differences has a negative impact on the convergence of your algorithm. **You need exact sensitivities.**

2. The dimension (n) of your problem is too large for finite differences or forward sensitivities to be feasible approaches to the accumulation of the gradient. **You need cheap gradients**.

[2]... three or more depending on the implementation and the used programming language; A factor of 50 is typically rather straight-forward to get. Factors below 20 may require more substantial insight into the structure of the given implementation of F. In order to reach the theoretical minimum of around three a highly tuned adjoint is required. Its generation may turn out to be highly challenging for complex problems. The minimization of the factor is the subject of various ongoing research and development efforts world-wide. Refer to `www.autodiff.org` for references.

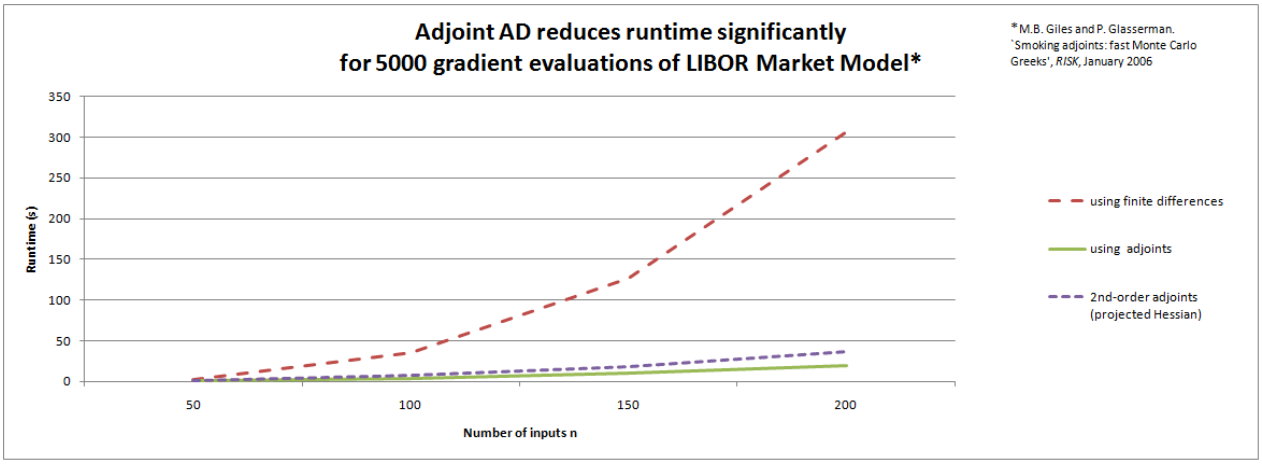

Figure 1: Superiority of Adjoints

For illustration we apply reverse mode AD to the LIBOR Market Model from [4] for calculating the sensitivities of securities. The time required in evaluating 5,000 deltas grows with n when using finite differences but increases only slowly (independent of n) with adjoints. The results are illustrated in Figure 1.

3. Your numerical model changes over time due to new insight resulting from ongoing research and development. The corresponding sensitivity codes need to be updated too. Nontrivial man hours may be involved. **You need AD software to (partially) automate the generation of sensitivity codes.**

References

[1] C. Bischof, M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors. *Advances in Automatic Differentiation*, number 64 in Lecture Notes in Computational Science and Engineering, Berlin, 2008. Springer.

[2] M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, editors. *Automatic Differentiation: Applications, Theory, and Tools*, number 50 in Lecture Notes in Computational Science and Engineering, Berlin, 2005. Springer.

[3] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors. *Automatic Differentiation of Algorithms – From Simulation to Optimization*, New York, 2002. Springer.

[4] M. Giles and P. Glasserman. Smoking adjoints: Fast monte carlo greeks. *RISK*, January 2006.

[5] A. Griewank and A. Walther. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation. Second Edition*. Number 19 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

For details please contact infodesk@nag.com.