

# On the use of direct vs. iterative sparse linear solvers in adjoint computations

Klaus Leppkes      Uwe Naumann\*

## Abstract

We discuss a simplification of a known computationally hard combinatorial optimization problem in Automatic Differentiation (AD) [6], that is the efficient evaluation of adjoints of numerical programs involving the solution of sparse linear systems [5]. Our results are based on work by other members of the CSC community on direct solvers for such systems [3, 4].

More specifically we are interested in the efficient computation of  $\bar{A} = \frac{\partial x}{\partial A} \cdot \bar{x}$  and  $\bar{b} = \frac{\partial x}{\partial b} \cdot \bar{x}$  for a given  $x = x(t)$  where  $A(t) \cdot x = b(t)$  and  $A \in R^{n \times n}$ ,  $x, b \in R^n$ . Our target application uses an iterative solver for the computation of  $x$ . Classical AD by operator overloading records the entire computation on a tape [1] yielding a worst-case computational complexity of  $O(n^3)$  for the adjoint propagation. We aim to reduce this cost by an order of magnitude by replacing the iterative solver by a direct solver whose factorization of  $A$  can be reused for the adjoint propagation as shown in the following.

Both the adjoints  $\bar{A}$  of  $A$  and  $\bar{b}$  of  $b$  are functions of the adjoint  $\bar{x}$  of  $x$ .  $\bar{A}(\bar{x})$  is computed efficiently based on the following straight forward algebraic manipulations:

$$\begin{aligned} A \cdot x &= b \\ \frac{dA}{dA} \cdot x + A \cdot \frac{dx}{dA} &= \frac{db}{dA} = 0 \\ A \cdot \frac{dx}{dA} &= -\frac{dA}{dA} \cdot x \\ \bar{x}^T \cdot \frac{dx}{dA} &= \bar{x}^T \cdot A^{-1} \cdot -\frac{dA}{dA} \cdot x \quad . \end{aligned}$$

$\alpha := \bar{x}^T \cdot A^{-1}$  is calculated by solving  $A^T \cdot \alpha = \bar{x}$ . A given LU or QR decomposition of  $A$  can be reused yielding a computational cost of  $O(n^2)$ .

The cost of computing  $\alpha \cdot -\frac{dA}{dA} \cdot x$  is  $\eta(A) \cdot O(1)$ , where  $\eta(A)$  denotes the number of nonzeros in  $A$ . Obviously,

- $-\frac{dA}{dA(i,j)}$  is a  $n \times n$  matrix, where only the  $(i, j)$ -th entry equals  $-1$ .
- $u := -\frac{dA}{dA(i,j)} \cdot x$  is a  $n \times 1$  vector with  $u(i) = -x(j)$ .
- $\langle \alpha, u \rangle$  reduces to  $\alpha(i) \cdot -x(j)$ .

Hence  $\bar{A}(i, j) = \alpha(i) \cdot -x(j)$  for each nonzero entry of  $A$ .  $\bar{b}(\bar{x})$  is computed similarly:

$$\begin{aligned} A \cdot x &= b \\ \frac{dA}{db} \cdot x + A \cdot \frac{dx}{db} &= \frac{db}{db} \\ A \cdot \frac{dx}{db} &= \frac{db}{db} \\ \bar{x}^T \cdot \frac{dx}{db} &= \bar{x}^T \cdot A^{-1} \cdot \frac{db}{db} \quad . \end{aligned}$$

The previously computed vector  $\alpha := \bar{x}^T \cdot A^{-1}$  is reused. The derivative  $\frac{db}{db_i}$  is a  $n \times 1$  vector, whose  $i$ -th entry equals 1. The inner product  $\langle \alpha, \frac{db}{db_i} \rangle$  reduces to the  $i$ -th entry of  $\alpha$ .

---

\*LuFG Informatik 12, RWTH Aachen University, D-52056 Aachen, Germany.

Moreover  $\bar{b}(i) = t(i)$  for each entry of  $b$ . The cost is reduced to  $O(n)$  yielding an overall cost of  $O(n^2)$ .

This method has been applied successfully to ICON – a large general circulation model used in the climate sciences [2] (see also <http://icon.enes.org>). Large gradients are needed for adjoint error correction methods. These are calculated by using the taping approach implemented in the differentiation-enabled NAG Fortran compiler[1]. The originally used iterative solver takes only a few iterations to converge to the required accuracy. It was replaced with version 5.2 of the sparse direct solver UMFPACK [3].

For a given relevant problem size we get impressive results. Note that the iterative solver is slowed down considerably by producing 170,804,302 tape entries (each 36 Bytes=5.73GB). The direct solver is run without taping as the factorization of  $A$  is reused in the adjoint propagation.

	<b>taping</b>	<b>our approach</b>
<i>time for solving linear system (in sec.)</i>	7.2	1.0
<i>memory (in GB)</i>	5.73	0.077
<i>time for computing <math>\bar{A}(\bar{x})</math> and <math>\bar{b}(\bar{x})</math> (in sec.)</i>	9.4	0.1

We observe a local speedup of 14,8 and savings in memory of 98.7% for adjoining the linear solver. The iterative linear solver in the ICON shallow water module creates about 56% of all tape entries. We get an overall speedup of 3. More detailed results will be shown at the conference.

## References

- [1] Naumann, U. and Riehme, J.: *A Differentiation-Enabled Fortran 95 Compiler*, ACM Transactions on Mathematical Software 31(4), ACM, 2005.
- [2] Bonaventura, L., Kornblueh, L., Heinze, T. and Ripodas, P.: *A semi-implicit method conserving mass and potential vorticity for the shallow water equations on the sphere*,
- [3] Davis, T. A.: *Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method*, ACM Transactions on Mathematical Software 30(2), ACM, 2004. International Journal of Numerical Methods in Fluids (47), 2005.
- [4] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., and Liu, J. W. H.: *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Analysis and Applications 20(3), 720755, 1999
- [5] Naumann, U.: *DAG Reversal is NP-Complete*, J. Discr. Alg., Elsevier, 2008. Appeared Online.
- [6] Griewank, A. and Walther, A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Second Edition, SIAM, 2008.