Collected matrix derivative results for forward and reverse mode AD

M. B. Giles

Abstract This paper collects together a number of matrix derivative results which are very useful in forward and reverse mode AD. It highlights in particular the remarkable contribution of a 1948 paper by Dwyer and Macphail which derives the linear and adjoint sensitivities of a matrix product, inverse and determinant, and a number of related results motivated by applications in multivariate analysis in statistics.

Key words: Forward mode, reverse mode, numerical linear algebra

1 Introduction

As the title suggests, there are no new theoretical results in this paper. Instead, it is a collection of results on derivatives of matrix functions, expressed in a form suitable for both forward and reverse mode algorithmic differentiation [8] of basic operations in numerical linear algebra. All results are derived from first principles, and it is hoped this will be a useful reference for the AD community.

The paper is organised in two sections. The first covers the sensitivity analysis for matrix product, inverse and determinant, and other associated results. Remarkably, most of these results were first derived, although presented in a slightly different form, in a 1948 paper by Dwyer and Macphail [4]. Comments in a paper by Dwyer in 1967 [3] suggest that the "Dwyer/Macphail calculus" was not widely used in the intervening period, but thereafter it has been used extensively within statistics, appearing in a number of books [11, 14, 15, 17] from the 1970's onwards. For a more extensive bibliography, see the notes at the end of section 1.1 in [12].

M.B. Giles
Oxford University Computing Laboratory, Oxford, OX1 3QD, United Kingdom
e-mail: giles@comlab.ox.ac.uk

The second section discusses Maximum Likelihood Estimation which was one of the motivating applications for Dwyer's work, and also comments on how the form of the results in Dwyer and Macphail's paper relates to the AD notation used in this paper.

An expanded version of this paper [6] also contains material on the sensitivity of eigenvalues and eigenvectors, singular values and singular vectors, and associated results for matrix norms.

2 Matrix product, inverse and determinant

2.1 Preliminaries

We consider a computation which begins with a single scalar input variable S_I and eventually, through a sequence of calculations, computes a single scalar output S_O. Using standard AD terminology, if A is a matrix which is an intermediate variable within the computation, then $\dot{A}$ denotes the derivative of A with respect to S_I, while $\overline{A}$ (which has the same dimensions as A, as does $\dot{A}$) denotes the derivative of S_O with respect to each of the elements of A.

Forward mode AD starts at the beginning and differentiates each step of the computation. Given an intermediate step of the form

$$C = f(A,B)$$

then differential calculus expresses infinitesimal perturbations to this as

$$\mathrm{d}C = \frac{\partial f}{\partial A}\,\mathrm{d}A + \frac{\partial f}{\partial B}\,\mathrm{d}B. \tag{1}$$

Taking the infinitesimal perturbations to be due to a perturbation in the input variable S_I gives

$$\dot{C} = \frac{\partial f}{\partial A}\,\dot{A} + \frac{\partial f}{\partial B}\,\dot{B}.$$

This defines the process of forward mode AD, in which each computational step is differentiated to determine the sensitivity of the output to changes in S_I.

Reverse mode AD computes sensitivities by starting at the end and working backwards. By definition,

$$\mathrm{d}S_O = \sum_{i,j} \overline{C}_{i,j}\,\mathrm{d}C_{i,j} = \mathrm{Tr}(\overline{C}^T\,\mathrm{d}C),$$

where $\mathrm{Tr}(A)$ is the trace operator which sums the diagonal elements of a square matrix. Inserting (1) gives

$$\mathrm{d}S_O = \mathrm{Tr}\left(\overline{C}^T \frac{\partial f}{\partial A}\,\mathrm{d}A\right) + \mathrm{Tr}\left(\overline{C}^T \frac{\partial f}{\partial B}\,\mathrm{d}B\right).$$

Assuming A and B are not used in other intermediate computations, this gives

$$\overline{A} = \left(\frac{\partial f}{\partial A}\right)^T \overline{C}, \qquad \overline{B} = \left(\frac{\partial f}{\partial B}\right)^T \overline{C}.$$

This defines the process of reverse mode AD, working backwards through the sequence of computational steps originally used to compute S_O from S_I. The key therefore is the identity

$$\mathrm{Tr}(\overline{C}^T \mathrm{d}C) = \mathrm{Tr}(\overline{A}^T \mathrm{d}A) + \mathrm{Tr}(\overline{B}^T \mathrm{d}B). \tag{2}$$

To express things in this desired form, the following identities will be useful:

$$\begin{aligned} \mathrm{Tr}(A^T) &= \mathrm{Tr}(A), \\ \mathrm{Tr}(A+B) &= \mathrm{Tr}(A) + \mathrm{Tr}(B), \\ \mathrm{Tr}(AB) &= \mathrm{Tr}(BA). \end{aligned}$$

In considering different operations $f(A,B)$, in each case we first determine the differential identity (1) which immediately gives the forward mode sensitivity, and then manipulate it into the adjoint form (2) to obtain the reverse mode sensitivities. This is precisely the approach used by Minka [13] (based on Magnus and Neudecker [11]) even though his results are not expressed in AD notation, and the reverse mode sensitivities appear to be an end in themselves, rather than a building block within an algorithmic differentiation of a much larger algorithm.

2.2 Elementary results

2.2.1 Addition

If $C = A + B$ then obviously

$$\mathrm{d}C = \mathrm{d}A + \mathrm{d}B$$

and hence in forward mode

$$\dot{C} = \dot{A} + \dot{B}.$$

Also,

$$\mathrm{Tr}(\overline{C}^T \mathrm{d}C) = \mathrm{Tr}(\overline{C}^T \mathrm{d}A) + \mathrm{Tr}(\overline{C}^T \mathrm{d}B)$$

and therefore in reverse mode

$$\overline{A} = \overline{C}, \qquad \overline{B} = \overline{C}.$$

2.2.2 Multiplication

If $C = A\,B$ then

$$\mathrm{d}C = \mathrm{d}A\;B + A\;\mathrm{d}B$$

and hence in forward mode

$$\dot{C} = \dot{A}\,B + A\,\dot{B}.$$

Also,

$$\mathrm{Tr}(\overline{C}^T\,\mathrm{d}C) \;=\; \mathrm{Tr}(\overline{C}^T\mathrm{d}A\,B) + \mathrm{Tr}(\overline{C}^T A\,\mathrm{d}B) \;=\; \mathrm{Tr}(B\overline{C}^T\mathrm{d}A) + \mathrm{Tr}(\overline{C}^T A\,\mathrm{d}B),$$

and therefore in reverse mode

$$\overline{A} = \overline{C}\,B^T, \qquad \overline{B} = A^T\,\overline{C}.$$

2.2.3 Inverse

If $C = A^{-1}$ then

$$C\,A = I \quad\Longrightarrow\quad \mathrm{d}C\,A + C\,\mathrm{d}A = 0 \quad\Longrightarrow\quad \mathrm{d}C = -C\;\mathrm{d}A\;C.$$

Hence in forward mode we have

$$\dot{C} = -C\,\dot{A}\,C.$$

Also,

$$\mathrm{Tr}(\overline{C}^T\,\mathrm{d}C) \;=\; \mathrm{Tr}(-\overline{C}^T A^{-1}\mathrm{d}A\,A^{-1}) \;=\; \mathrm{Tr}(-A^{-1}\overline{C}^T A^{-1}\mathrm{d}A)$$

and so in reverse mode

$$\overline{A} = -A^{-T}\,\overline{C}A^{-T} = -C^T\,\overline{C}\,C^T\,.$$

2.2.4 Determinant

If we define $\widetilde{A}$ to be the matrix of co-factors of A, then

$$\det A = \sum_j A_{i,j}\widetilde{A}_{i,j}, \qquad A^{-1} = (\det A)^{-1}\widetilde{A}^T.$$

for any fixed choice of i. If $C = \det A$, it follows that

$$\frac{\partial C}{\partial A_{i,j}} = \widetilde{A}_{i,j} \quad\Longrightarrow\quad \mathrm{d}C \;=\; \sum_{i,j}\widetilde{A}_{i,j}\,\mathrm{d}A_{i,j} \;=\; C\,\mathrm{Tr}(A^{-1}\mathrm{d}A).$$

Hence, in forward mode we have

$$\dot{C} = C\,\mathrm{Tr}(A^{-1}\dot{A}),$$

while in reverse mode C and $\overline{C}$ are both scalars and so we have

$$\overline{C}\,\mathrm{d}C \;=\; \mathrm{Tr}(\overline{C}CA^{-1}\mathrm{d}A)$$

and therefore

$$\overline{A} = \overline{C}CA^{-T}.$$

Note: in a paper in 1994 [10], Kubota states that the result for the determinant is well known, and explains how reverse mode differentiation can therefore be used to compute the matrix inverse.

2.3 Additional results

Other results can be obtained from combinations of the elementary results.

2.3.1 Matrix inverse product

If $C = A^{-1}B$ then

$$\mathrm{d}C = \mathrm{d}A^{-1}\,B + A^{-1}\,\mathrm{d}B = -A^{-1}\mathrm{d}A\,A^{-1}B + A^{-1}\,\mathrm{d}B = A^{-1}(\mathrm{d}B - \mathrm{d}A\,C),$$

and hence

$$\dot{C} = A^{-1}(\dot{B} - \dot{A}\,C),$$

and

$$\begin{aligned}\mathrm{Tr}(\overline{C}^T\,\mathrm{d}C) &= \mathrm{Tr}(\overline{C}^T A^{-1}\mathrm{d}B) - \mathrm{Tr}(\overline{C}^T A^{-1}\mathrm{d}A\,C)\\ &= \mathrm{Tr}(\overline{C}^T A^{-1}\mathrm{d}B) - \mathrm{Tr}(C\overline{C}^T A^{-1}\mathrm{d}A)\\ \implies \qquad \overline{B} = A^{-T}\,\overline{C}, &\qquad \overline{A} = -A^{-T}\,\overline{C}C^T = -\overline{B}C^T.\end{aligned}$$

2.3.2 First quadratic form

If $C = B^T A\,B$, then

$$\mathrm{d}C = \mathrm{d}B^T A\,B + B^T\,\mathrm{d}A\,B + B^T A\,\mathrm{d}B.$$

and hence

$$\dot{C} = \dot{B}^T A\,B + B^T\dot{A}\,B + B^T A\,\dot{B},$$

and

$$\begin{aligned}\mathrm{Tr}(\overline{C}^T \mathrm{d}C) &= \mathrm{Tr}(\overline{C}^T \mathrm{d}B^T A\,B) + \mathrm{Tr}(\overline{C}^T B^T \mathrm{d}A\,B) + \mathrm{Tr}(\overline{C}^T B^T A\,\mathrm{d}B) \\ &= \mathrm{Tr}(\overline{C} B^T A^T\,\mathrm{d}B) + \mathrm{Tr}(B\overline{C}^T B^T \mathrm{d}A) + \mathrm{Tr}(\overline{C}^T B^T A\,\mathrm{d}B)\end{aligned}$$

$$\implies \qquad \overline{A} = B\overline{C}\,B^T, \qquad \overline{B} = A\,B\overline{C}^T + A^T B\overline{C}.$$

2.3.3 Second quadratic form

If $C = B^T A^{-1} B$, then similarly one gets

$$\dot{C} = \dot{B}^T A^{-1} B - B^T A^{-1} \dot{A} A^{-1} B + B^T A^{-1} \dot{B},$$

and

$$\overline{A} = -A^{-T} B\overline{C}\,B^T A^{-T}, \qquad \overline{B} = A^{-1} B\overline{C}^T + A^{-T} B\overline{C}.$$

2.3.4 Matrix polynomial

Suppose $C = p(A)$, where A is a square matrix and $p(A)$ is the polynomial

$$p(A) = \sum_{n=0}^{N} a_n A^n.$$

Pseudo-code for the evaluation of C is as follows:

```
C := a_N I
for n from N-1 to 0
    C := A C + a_n I
end
```

where I is the identity matrix with the same dimensions as A.

Using standard forward mode AD with the matrix product results gives the corresponding pseudo-code to compute $\dot{C}$:

```
Ċ := 0
C := a_N I
for n from N-1 to 0
    Ċ := Ȧ C + A Ċ
    C := A C + a_n I
end
```

Similarly, the reverse mode pseudo-code to compute $\overline{A}$ is:

$C_N := a_N I$
`for` n `from` $N-1$ `to` 0
  $C_n := AC_{n+1} + a_n I$
`end`

$\overline{A} := 0$
`for` n `from` 0 `to` $N-1$
  $\overline{A} := \overline{A} + \overline{C}C_{n+1}^T$
  $\overline{C} := A^T \overline{C}$
`end`

Note the need in the above code to store the different intermediate values of C in the forward pass so that they can be used in the reverse pass.

2.3.5 Matrix exponential

In MATLAB, the matrix exponential

$$\exp(A) \equiv \sum_{n=0}^{\infty} \frac{1}{n!} A^n,$$

is approximated through a scaling and squaring method as

$$\exp(A) \approx \left(p_1(A)^{-1} p_2(A) \right)^m,$$

where m is a power of 2, and p_1 and p_2 are polynomials such that $p_2(x)/p_1(x)$ is a Padé approximation to $\exp(x/m)$ [9]. The forward and reverse mode sensitivities of this approximation can be obtained by combining the earlier results for the matrix inverse product and polynomial.

3 MLE and the Dwyer/Macphail paper

A d-dimensional multivariate Normal distribution with mean vector μ and covariance matrix Σ has the joint probability density function

$$p(x) = \frac{1}{\sqrt{\det \Sigma}\ (2\pi)^{d/2}}\ \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right).$$

Given a set of N data points x_n, their joint probability density function is

$$P = \prod_{n=1}^{N} p(x_n).$$

Maximum Likelihood Estimation infers the values of μ and Σ from the data by choosing the values which maximise P. Since

$$\log P = \sum_{n=1}^{N} \left\{ -\tfrac{1}{2}\log(\det\Sigma) - \tfrac{1}{2}d\log(2\pi) - \tfrac{1}{2}(x_n-\mu)^T\Sigma^{-1}(x_n-\mu) \right\},$$

the derivatives with respect to μ and Σ are

$$\frac{\partial \log P}{\partial \mu} = -\sum_{n=1}^{N} \Sigma^{-1}(x_n-\mu),$$

and

$$\frac{\partial \log P}{\partial \Sigma} = -\tfrac{1}{2}\sum_{n=1}^{N} \left\{ \Sigma^{-1} - \Sigma^{-1}(x_n-\mu)\,(x_n-\mu)^T\Sigma^{-1} \right\}.$$

Equating these to zero gives the maximum likelihood estimates

$$\mu = N^{-1}\sum_{n=1}^{N} x_n,$$

and

$$\Sigma = N^{-1}\sum_{n=1}^{N} (x_n-\mu)\,(x_n-\mu)^T.$$

Although this example was not included in Dwyer and Macphail's original paper [4], it is included in Dwyer's later paper [3]. It is a similar application concerning the Likelihood Ratio Method in computational finance [7] which motivated the present author's investigation into this subject.

Returning to Dwyer and Macphail's original paper [4], it is interesting to note the notation they used to express their results, and the correspondence to the results presented in this paper. Using $\langle\cdot\rangle_{i,j}$ to denote the $(i,j)^{th}$ element of a matrix, and defining $J_{i,j}$ and $K_{i,j}$ to be matrices which are zero apart from a unit value for the $(i,j)^{th}$ element, then their equivalent of the equations for the matrix inverse are

$$\frac{\partial A^{-1}}{\partial \langle A\rangle_{i,j}} = -A^{-1}J_{i,j}A^{-1},$$
$$\frac{\partial \langle A^{-1}\rangle_{i,j}}{\partial A} = -A^{-T}K_{i,j}A^{-T}.$$

In the forward mode, defining the input scalar to be $S_I = A_{i,j}$ for a particular choice (i,j) gives $\dot{A} = J_{i,j}$ and hence, in our notation with $B = A^{-1}$,

$$\dot{B} = -A^{-1}\dot{A}A^{-1}.$$

Similarly, in reverse mode, defining the output scalar to be $S_O = (A^{-1})_{i,j}$ for a particular choice (i, j) gives $\overline{B} = K_{i,j}$ and so

$$\overline{A} = -A^{-T}\,\overline{B}\,A^{-T},$$

again matching the result derived previously.

4 Validation

All results in this paper have been validated with a MATLAB code which performs two checks.

The first check uses a wonderfully simple technique based on the Taylor series expansion of an analytic function of a complex variable [16]. If $f(x)$ is analytic with respect to each component of x, and $y = f(x)$ is real when x is real, then

$$\dot{y} = \lim_{\varepsilon \to 0} \mathscr{I}\{\varepsilon^{-1} f(x + \mathrm{i}\varepsilon\dot{x})\}.$$

Taking $\varepsilon = 10^{-20}$ this is used to check the forward mode derivatives to machine accuracy. Note that this is similar to the use of finite differences, but without roundoff inaccuracy.

The requirement that $f(x)$ be analytic can require some creativity in applying the check. For example, the singular values of a complex matrix are always real, and so they cannot be an analytic function of the input matrix. However, for real matrices, the singular values are equal to the square root of the eigenvalues of $A^T A$, and these eigenvalues are an analytic function of A.

The second check is that when inputs A, B lead to an output C, then the identity

$$\mathrm{Tr}(\overline{C}^T \dot{C}) = \mathrm{Tr}(\overline{A}^T \dot{A}) + \mathrm{Tr}(\overline{B}^T \dot{B}),$$

should be satisfied for all $\dot{A}, \dot{B}$ and $\overline{C}$. This check is performed with randomly chosen values for these matrices.

5 Conclusions

This paper has reviewed a number of matrix derivative results in numerical linear algebra. These are useful in applying both forward and reverse mode algorithmic differentiation at a higher level than the usual binary instruction level considered by most AD tools. As well as being helpful for applications which use numerical libraries to perform certain computationally intensive tasks, such as solving a system of simultaneous equations, it could be particularly relevant to those programming in MATLAB or developing AD tools for MATLAB [1, 2, 5, 18].

Acknowledgements I am grateful to Shaun Forth for the Kubota reference, Andreas Griewank for the Minka and Magnus & Neudecker references, and Nick Trefethen for the Mathai and Stewart & Sun references.

This research was funded in part by a research grant from Microsoft Corporation, and in part by a fellowship from the UK Engineering and Physical Sciences Research Council.

References

1. Bischof, C., Bücker, H., Lang, B., Rasch, A., Vehreschild, A.: Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs. In: Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002), pp. 65–72. IEEE Computer Society (2002)
2. Coleman, T., Verma, A.: ADMIT-1: Automatic differentiation and MATLAB interface toolbox. ACM Transactions on Mathematical Software **26**(1), 150–175 (2000)
3. Dwyer, P.: Some applications of matrix derivatives in multivariate analysis. Journal of the American Statistical Association **62**(318), 607–625 (1967)
4. Dwyer, P., Macphail, M.: Symbolic matrix derivatives. The Annals of Mathematical Statistics **19**(4), 517–534 (1948)
5. Forth, S.: An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. ACM Transactions on Mathematical Software **32**(2), 195–222 (2006)
6. Giles, M.: An extended collection of matrix derivative results for forward and reverse mode automatic differentiation. Tech. Rep. NA07/??, Oxford University Computing Laboratory (2007)
7. Glasserman, P.: Monte Carlo Methods in Financial Engineering. Springer-Verlag, New York (2004)
8. Griewank, A.: Evaluating derivatives : principles and techniques of algorithmic differentiation. SIAM (2000)
9. Higham, N.: The scaling and squaring method for the matrix exponential revisited. SIAM Journal on Matrix Analysis and Applications **26**(4), 1179–1193 (2005)
10. Kubota, K.: Matrix inversion algorithms by means of automatic differentiation. Applied Mathematics Letters **7**(4), 19–22 (1994)
11. Magnus, J., Neudecker, H.: Matrix differential calculus with applications in statistics and econometrics. John Wiley & Sons (1988)
12. Mathai, A.: Jacobians of matrix transformations and functions of matrix argument. World Scientific, New York (1997)
13. Minka, T.: Old and new matrix algebra useful for statistics. `http://research.microsoft.com/~minka/papers/matrix/` (2000)
14. Rao, C.: Linear statistical inference and its applications. Wiley, New York (1973)
15. Rogers, G.: Matrix derivatives. Marcel Dekker, New York (1980)
16. Squire, W., Trapp, G.: Using complex variables to estimate derivatives of real functions. SIAM Review **10**(1), 110–112 (1998)
17. Srivastava, M., Khatri, C.: An introduction to multivariate statistics. North Holland, New York (1979)
18. Verma, A.: ADMAT: automatic differentiation in MATLAB using object oriented methods. In: SIAM Interdiscplinary Workshop on Object Oriented Methods for Interoperability, pp. 174–183. SIAM (1998)