Linear Algebra and Matrix Calculus

Appendix C

Matrices and Linear Algebra

This appendix addresses briefly various matrix/linear-algebra basis, really attempting to be a reference for the reader, as opposed to a development. Many of the operations reviewed here like matrix/tensor definition , basic operations, inversion, and factorization nominally implement with routines in implementation packages for design (like Matlab). Further, this section attempts to enumerate and explain briefly the salient relevant aspects of vector/matrix calculus that supports the designs in Chapters **??** - 5 or adaptive design implementation in Chapter 7.

Section C.1 reviews matrix basics. Many early topics are likely familiar to this text's reader in advance, but this section reviews them in terminology common to this text's other chapters and thus removes some ambiguities. Some reference to tensors, and alternative products like Hadamard and Kronecker, to help avert direct consideration of tensors. Common heavily used design factorizations like QR, Cholesky, singular-value, and eigenvalue appear in common terminology along with some common matrix inversion recursions. Oder is also addressed along with the Frobenious norm and inner product.

Section C.2 progresses to matrix calculus, focusing first on real matrices and functions, but expanding then to complex matrices. Various useful derivatives are listed in Section C.2, with attempt also to explain and hopefully avert some common mistakes with complex convex optimization that sometimes appear in widely used software packages. Section C.3 visits optimization of scalar and matrix functions, particularly looking at convex procedures and the appropriate descent algorithms' necessary "gradient" or "steepest descent" directional calculations. https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

C.1 Vectors and Matrices

This text most often uses matrices in conjunction with MMSE designs (Appendix D addresses MMSE) and many related matrix and system concepts. In that context and in all this text's main chapters, the notation of $\boldsymbol{x}^*$ or A^* for conjugate transpose of a vector or matrix respectively is sufficient, averting special separate notation for each of conjugate (no transpose), transpose, and both. However, general matrix calculus creates situations where the extra notation is necessary. This notation may be necessary to help software algorithms (See Appendix G) internal code that may supply results into use in main chapters where subsequent use only needs the consistent conjugate and transpose notation, $\boldsymbol{x}^*$. For this Appendix (C) and understanding detailed use in software, the notation $\boldsymbol{x}^t$ and A^t means transpose with no conjugate and $\boldsymbol{x}^{t*}$ and A^{t*} means take the transpose twice and the conjugate of that, or equivalently just replace every element with its conjugate.

An $m \times n$ matrix $A \in \mathbb{C}^{mn}$ has column-vector characterization[1]

$$A = [\boldsymbol{a}_1 \; \boldsymbol{a}_2 \; ... \; \boldsymbol{a}_n] \tag{C.1}$$

and of course then

$$A^t = \begin{bmatrix} \boldsymbol{a}_1^t \\ \boldsymbol{a}_2^t \\ \vdots \\ \boldsymbol{a}_n \end{bmatrix} \quad . \tag{C.2}$$

where the superscript of t denotes transpose, so A^t is the $n \times m$ matrix with i, j^{th} element equal to a_{ji}. A could also be written with row vectors $\tilde{\boldsymbol{a}}_{i=1,...,m}$ as

$$A = \begin{bmatrix} \tilde{\boldsymbol{a}}_1^t \\ \tilde{\boldsymbol{a}}_2^t \\ \vdots \\ \tilde{\boldsymbol{a}}_m \end{bmatrix} \quad , \tag{C.3}$$

where it is clear that often $\tilde{\boldsymbol{a}}_i^t \neq \boldsymbol{a}_i^t$ and that reocrding causes any deviation. This ordering can become important when tensors effectively become elongated vectors.

A **tensor** adds additional indices (third, fourth) etc to a matrix so that there are columns, rows, pages, ... A vector is a special case of a matrix where either m or n is one. Similarly a matrix is a special case of tensor and so on. Matlab calls tensors "arrays."

C.1.1 matrix multiplication and inverses

The **matrix product** of $m \times n$ matrix A and $n \times r$ matrix B is the $m \times r$ matrix

$$A \cdot B = \begin{bmatrix} \tilde{\boldsymbol{a}}_1^t \cdot \boldsymbol{b}_1 & \tilde{\boldsymbol{a}}_1^t \cdot \boldsymbol{b}_2 & ... & \tilde{\boldsymbol{a}}_1^t \cdot \boldsymbol{b}_r \\ \tilde{\boldsymbol{a}}_2^t \cdot \boldsymbol{b}_1 & \tilde{\boldsymbol{a}}_2^t \cdot \boldsymbol{b}_2 & ... & \tilde{\boldsymbol{a}}_2^t \cdot \boldsymbol{b}_r \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\boldsymbol{a}}_m^t \cdot \boldsymbol{b}_1 & \tilde{\boldsymbol{a}}_m^t \cdot \boldsymbol{b}_2 & ... & \tilde{\boldsymbol{a}}_m^t \cdot \boldsymbol{b}_r \end{bmatrix} \quad . \tag{C.4}$$

Matrix multiplication of a vector produces another vector, where the initial vector specifies the weights of the matrix' column vectors in a linear transformation. Through multiplication a matrix transforms the unit hypercube, for instance, into a polytope. The determinant of a square matrix, $|A|$, measures that polytope's volume. The determinant is the sum of the products of any row (column) vector's elements with their cofactors. The cofactors are equal to $(-1)^{i+j}$ times the determinant of the submatrix remaining after the i^{th} row and j^{th} column have been deleted. This creates a recursive way to compute

[1] The ordering here is similar to Matlab with $1, 1$ entry in upper left and m, n entry in lower right, which differs from this text's time/frequency and space orderings in various chapters. The text's order may simply map all column vectors according $\boldsymbol{a} \rightarrow J \cdot \boldsymbol{a}$ where J is the Hankel identity with all ones on the antidiagonal and zeros elsewhere.

the determinant because a simple scalar is its own determinant, allowing the 2×2 determinant of the matrix $|[a\ b\,;c\ d]| = a \cdot c - b \cdot d$. The adjoint matrix $adj(A)$ replaces each element with its cofactor and takes the transpose.

The inverse of a square matrix A^{-1} exists when $|A| \neq 0$ and is $A^{-1} \triangleq \frac{adj(A)}{|A|}$, and $A \cdot A^{-1} = A^{-1} \cdot A = I$. The trace of square matrix, $trace\{A\}$, is the sum of its diagonal elements.

The following statements are all true (with $\alpha \in \mathbb{C}$):

1. $trace\{A + B\} = trace\{A\} + trace\{B\}$,
2. $trace\{\alpha A\} = \alpha \cdot trace\{A\}$,
3. $trace\{A \cdot B\} = trace\{B \cdot A\}$
4. $|A \cdot B| = |A| \cdot |B|$,
5. $|\alpha \cdot A| = |\alpha|^m \cdot |A|$,
6. $|A^{-1}| = 1/|A|$, and
7. $|I + \boldsymbol{u} \cdot \boldsymbol{v}^*| = (1 + \boldsymbol{u}^* \cdot \boldsymbol{v})^* = 1 + \boldsymbol{u}^* \cdot \boldsymbol{v}$.

A symmetric (square) matrix has $A = A^t$, while a conjugate-symmetric, or Hermetian symmetric, matrix has $A = A^*$. A unitary matrix has $A^{-1} = A^*$.

The rank of a matrix ϱ_A is the maximum number of linearly independent columns or rows (see Appendix D for linear independence) of A, and $\varrho_A \leq \min(m, n)$.

Cramers Matrix Inversion: Any nonsingular square matrix

$$A = \begin{bmatrix} a_{n,n} & a_{n-1,n} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,2} & a_{n-1,2} & \cdots & a_{1,2} \\ a_{n,1} & a_{n-1,1} & \cdots & a_{1,1} \end{bmatrix} \tag{C.5}$$

has an inverse that can be found by the formula

$$A^{-1} = \frac{1}{|A|} \cdot \begin{bmatrix} (-1)^{(n+n)} \cdot |A_{n,n}| & (-1)^{(n-1+n)} \cdot |A_{n-1,n}| & \cdots & (-1)^{(1+n)} |A_{1,n}| \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^{(n+2)} \cdot |A_{n,2}| & (-1)^{(n-1+2)} \cdot |A_{n-1,2}| & \cdots & (-1)^{(1+2)} \cdot |A_{1,2}| \\ (-1)^{(n+1)} \cdot |A_{n,1}| & (-1)^{(n-1+1)} \cdot |A_{n-1,1}| & \cdots & (-1)^{(1+1)} \cdot |A_{1,1}| \end{bmatrix}^t \tag{C.6}$$

where $A_{i,j}$ is the **minor matrix** formed by deleting the i^{th} row and j^{th} column of A. So, basically replace each element by $\pm$ times its **minor** matrix' determinant and **take the transpose** before dividing by the overall determinant. The $\pm$ sign is always positive on the diagonal and alternates in sign in moving each position horizontally (or vertically) from the diagonal. The determinants without the plus/minus are often called **co-factors**.

C.1.1.1 Singularity and pseudoinverse

The matrix inversion lemma (or Woodbury's Lemma) states that

$$[A + B \cdot C \cdot D]^{-1} = A^{-1} - A^{-1} \cdot B \cdot \left[C^{-1} + D \cdot A^{-1} \cdot B\right]^{-1} \cdot D \cdot A^{-1} \ , \tag{C.7}$$

which of course requires A^{-1} and C^{-1} to exist (are nonsingular). This formula provides a means to update an inverse when more information becomes available or the matrix changes. For instance if $B = \boldsymbol{x}$, $C = I$, and $D = \boldsymbol{y}^*$, then this reduces to

$$(A + \boldsymbol{x} \cdot \boldsymbol{y}^*)^{-1} = A^{-1} - \frac{(A^{-1} \cdot x) \cdot (\boldsymbol{y}^* \cdot A^{-1})}{1 + \boldsymbol{y}^* \cdot A^{-1} \cdot \boldsymbol{x}} \ , \tag{C.8}$$

which considerably reduces complexity to update the inverse with respect using a full matrix inversion. Another recursive form follows from

$$\begin{bmatrix} A & B \\ D & C \end{bmatrix}^{-1} = \begin{bmatrix} (A - B \cdot C^{-1} D)^{-1} & -A^{-1} \cdot B \cdot (C - D \cdot A^{-1} \cdot B)^{-1} \\ -C^{-1} \cdot D \cdot (A - B \cdot C^{-1} \cdot D)^{-1} & (C - D \cdot A^{-1} \cdot B)^{-1} \end{bmatrix} \tag{C.9}$$

and builds larger-size inverses from smaller ones. Computation of an $m \times m$ inverse without any recursion requires the order of $O(m^3)$ calculations. The matlab command inv(A) provides the matrix inverse for nonsingular inputs.

The pseudoinverse A^+ can be used when$|A| = 0$ or when $m \neq n$, recognizing that such a situation corresponds to the equation $A \cdot \boldsymbol{x} = \boldsymbol{b}$ having either no exact solution (over-determined) or many solutions (under-determined). In the over-determined case, the solution is the one that minimizes the norm $\|A \cdot \boldsymbol{x} - \boldsymbol{b}\|^2$ in terms of solutions for $\boldsymbol{x}$, while the under-determined case finds the solution with minimum norm $\|\boldsymbol{x}\|^2$ among the many solutions. The unique pseudoinverse satisfies the following

1. $A \cdot A^+ \cdot A = A$,
2. $A^+ \cdot A \cdot A^+ = A^+$,
3. $A \cdot A^+ = A^{+*} \cdot A^*$, and
4. $A^+ \cdot A = A^* \cdot A^{+*}$.

$A^+ = A^{-1}$ when A is square nonsingular. The matlab command for pseudoinverse is pinv(A). However, when solving the equation $A \cdot \boldsymbol{x} = b$, the pseudoinverse solution is given by x = A \ b.

C.1.1.2 Factorizations

There are many useful matrix factorizations in data-transmission design.

QR Factorization: QR factorization takes any $m \times n$ matrix H and factors it so that

$$H = Q \cdot R \tag{C.10}$$

where $Q \cdot Q^* = Q^* \cdot Q = I$ is $m \times m$ unitary and R is $m \times n$ upper triangular. Sometimes it is convenient to write this as $Q \cdot L^*$ where $L = R^*$ is lower triangular. For non-square matrices, the nonzero entries of R are in the upper left (unless $H = 0$). The Matlab command for QR is qr(H). The R will be a full triangle if $\varrho_H \leq n = \min(m, n)$ and will otherwise look "trapezoidal" in nonzero entries.

```
>> [Q , R] = qr([ 1 2 3 ; 4 5 6])
Q =
   -0.2425   -0.9701
   -0.9701    0.2425
R =
   -4.1231   -5.3358   -6.5485
         0   -0.7276   -1.4552

[Q , R] = qr([ 1 2 3 ; 4 5 6]')
Q =
   -0.2673    0.8729    0.4082
   -0.5345    0.2182   -0.8165
   -0.8018   -0.4364    0.4082
R =
   -3.7417   -8.5524
         0    1.9640
         0         0
```

This text (see Appendix G) also has an rq.m matlab command that produces $H = R \cdot Q^*$ for convenience of use to certain designs. QR factorizations most often use either Householder Transformations or Givens Rotations. Chapter 7 has more on the Givens approach. These algorithms proceed sequentially with residual columns of H and can reorder them in a hidden way so that there are many $H = Q \cdot R$ factorizations. Matlab's QR factorization can track this ambiguity with respect to the order of diagonal elements because it places the largest-magnitude diagonal elements in the upper left (something useful with sparse matrices). More explicity, while $H = Q \cdot R$, it is possible also that (with J as a permutation matrix with inverse $J^* = J^t$ reversing the reordering)

$$H = \underbrace{(Q \cdot J)}_{Q1} \cdot \underbrace{(J^* \cdot R)}_{R1} = Q1 \cdot R1 \quad , \tag{C.11}$$

which with upper-triangular $R1$ essentially hides the reordering by J. If separate processing uses the Q (or $Q1$) it is sometimes useful to know this reorder that occurred within the algorithm. Matlab recorders this reordering explicity with the command [Q,R,J] = qr (H) so then $H \cdot J = Q \cdot R$ and the user can then back-out the reordering. This operation is particularly important with so-called multiuser broadcast channels where the reordering would confuse the labels of different user-receiver locations.

Singular Value Decomposition (SVD): Any $m \times n$ matrix H has a singular value decomposition (SVD) such that

$$H = F \cdot \Lambda \cdot M^* \tag{C.12}$$

where the $m \times m$ unitary matrix F satisfies $F \cdot F^* = F^* \cdot F = I_m$ and the $n \times n$ unitary matrix M satisfies $M \cdot M^* = M^* \cdot M = I_n$, while Λ is an $m \times n$ diagonal matrix with real nonnegative entries (even if H is complex). The singular values are unique, and Matlab's SVD command [F, Lambda, M] = svd(H) arranges them largest to smallest from the upper left corner down. Matlab's "economy" option will keep only the columns of F and M that correspond to nonzero singular values, but $H = F \cdot \Lambda \cdot M^*$ still holds. An algorithm for computing SVD using MMSE methods appears in Chapter 7.

When H is square nonsingular, then $|H| = e^{\jmath\theta} \cdot \prod_{i=1}^{m} \lambda_i$, with the ambiguity on sign arising from the fact that a unitary matrix may have determinant equal to $e^{\jmath\theta_F}$ where $0 \leq \theta < 2\pi$, and thus $\theta = \theta_F - \theta_M$.

Eigendecomposition: Eigenvalues and eigenvectors are useful in many engineering areas. However, this text uses SVD for matrices that are not conjugate symmetric autocorrelation matrices. This text only uses eigendecomposition for positive semi-definite autocorrelation matrices (see Appendix D). In this case such a matrix has eigendecomposition

$$R = V \cdot \boldsymbol{\mathcal{E}} \cdot V^* \tag{C.13}$$

where $V \cdot V^* = V^* \cdot V = I$ is unitary and $\boldsymbol{\mathcal{E}}$ is a positive semi-definite diagonal matrix of eigenvalues such that

$$R \cdot \boldsymbol{v}_n = \boldsymbol{\mathcal{E}}_n \cdot \boldsymbol{v}_n \tag{C.14}$$

where $V = [v_1 \ ... v_{\varrho_R}]$. More generally the V^* is replaced by V^{-1} for any square matrix R. For the autocorrelation matrix, all the eigenvalues are nonnegative and then $|R| = \prod_{i=1}^{m} \boldsymbol{\mathcal{E}}_i$. In general $trace\{R\} = \sum_{i=1}^{m} \boldsymbol{\mathcal{E}}_i$.

The nonzero eigenvalues of the symmetric positive semidefinite matrices $H \cdot H^*$ and $H^* \cdot H$ are the same and equal to the squared singular values of H.

Cholesky Factorization Cholesky factorization of a nonsingular square autocorrelation matrix R is

$$R = G \cdot G^* \tag{C.15}$$

where G is upper triangular[2]. Often in this book, the related factorization (which is attained by use of the ldl_rev_idx.m program)

$$R = G \cdot S_x \cdot G^* \tag{C.16}$$

[2]This differs from conventional definition slightly (and Matlab) where G would be lower triangular; the difference is related to ordering, so the factorization provided from matlab can be reversed by using the command lohc.m when desired.

is used where G is monic (has ones on the diagonal) and upper triangular (so then $|G| = 1$). The diagonal elements of S_x are all positive real and are the the matrix R's **Cholesky Factors**. Note then $|R| = |S_x|$ when R is nonsingular. Chapter 5 generalizes Cholesky factorization to the singular case. The factorization in this form (lower times upper) is the inverse of what matlab's chol command produces, so this course's website has an lohc.m matlab command that provides the desired factorization. Matlab's chol command is however also directly useful in Chapter 5's canonical factorization of the backward channel model. Clearly $R^{-1} = G^{-*} \cdot G^{-1}$.

Schur Compliment: Schur Compliments essentially are a block Cholesky factorization of a matrix

$$\begin{bmatrix} A & B \\ D & C \end{bmatrix} = \begin{bmatrix} I & B \cdot C^{-1} \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} (A - B \cdot C^{-1} \cdot D) & 0 \\ 0 & C \end{bmatrix} \cdot \begin{bmatrix} I & 0 \\ C^{-1} \cdot D & I \end{bmatrix} \tag{C.17}$$

and so then

$$\begin{bmatrix} A & B \\ D & C \end{bmatrix}^{-1} = \begin{bmatrix} I & 0 \\ -C^{-1} \cdot D & I \end{bmatrix} \cdot \begin{bmatrix} (A - B \cdot C^{-1} \cdot D)^{-1} & 0 \\ 0 & C^{-1} \end{bmatrix} \cdot \begin{bmatrix} (I & -B \cdot C^{-1} \\ 0 & I \end{bmatrix} \quad . \tag{C.18}$$

The Schur method lends itself readily to recursive implementation and determination of a block Cholesky factorization by adding successive blocks to R (and enlarging its dimensionality to include increasingly larger sets of dimensions).

C.1.2 reordering and products

The function $vec(A)$ transforms an two-dimensional $m \times n$ matrix A into a one-dimensional $mn \times 1$ column vector

$$vec(A) = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_m \end{bmatrix} \quad . \tag{C.19}$$

The function $rvec(A)$ instead creates the $1 \times nm$ row vector

$$rvec(A) = \begin{bmatrix} \tilde{\boldsymbol{a}}_1^t & \tilde{\boldsymbol{a}}_2^t & ... & \tilde{\boldsymbol{a}}_n^t \end{bmatrix} \quad . \tag{C.20}$$

The ordering is such that

$$rvec(A) = \left[vec(A^t) \right]^t \quad , \tag{C.21}$$

which can be cumbersome as notation and also to implement. Alternately, there is a special permutation matrix[3], sometimes called a commutation matrix K_{mn}, such that

$$K_{mn} \cdot vec(A) = vec(A^t) \quad . \tag{C.22}$$

It is clear that $K_{mn} = K_{nm}^t = K_{nm}^{-1}$. The commutation matrix can be written with the Kronecker product to be defined precisely later as

$$K_{mn} = \sum_{j=1}^{n} (\boldsymbol{e}_j^t \otimes I_m \otimes e_j) \quad , \tag{C.23}$$

which may be convenient for computer-software implementation.

[3]Permutation matrices are square unitary matrices that have exactly one nonzero entry, which is 1 in every row and column. The nonzero entries determine the reordering after application to a column vector input.

C.1.2.1 Hadamard Product

The Hadamard matrix product is

$$(A \odot B)_{i,j} = (A)_{ij} \cdot (B)_{ij} \tag{C.24}$$

It has properties

$$\begin{aligned} A \odot B &= B \odot A && \text{(C.25)} \\ A \odot (B \odot C) &= (A \odot B) \odot C && \text{(C.26)} \\ A \odot (B + C) &= A \odot B + A \odot C && \text{(C.27)} \\ A \odot 0 &= 0 \odot A = 0 \quad . && \text{(C.28)} \end{aligned}$$

Of particular importance is the relation (with $D_{\boldsymbol{x}}$ being a matrix with vector $\boldsymbol{x}$'s elements along its diagonal)

$$\boldsymbol{x}^* \cdot (A \odot B) \cdot \boldsymbol{y} = trace\{D^*_{\boldsymbol{x}} \cdot A \cdot D_{\boldsymbol{y}} \cdot B^t\} \tag{C.29}$$

In particular by setting $\boldsymbol{x} = \boldsymbol{y}$, this form shows that the Hadamaard product of two positive semi-definite matrices must be positive semi-definite and vice versa. Other properties are

$$rank(A \odot B) \leq rank(A) \cdot rank(B) \tag{C.30}$$

and (with $\lambda_i(A)$ as the i^{th} largest eigenvalue of positive definite A and also a positive-definite B)

$$\prod_{i=k}^{n} \lambda_i(A \odot B) \geq \prod_{i=k}^{n} \lambda_i(A \cdot B) \; \forall k = 1, ..., n \quad . \tag{C.31}$$

also,

$$|A \odot B| \geq |A| \cdot |B| \tag{C.32}$$

for positive semi-definite A and B. If the matrices are vectors then

$$\boldsymbol{a} \odot \boldsymbol{b} = D_{\boldsymbol{a}} \cdot \boldsymbol{b} = D_{\boldsymbol{b}} \cdot \boldsymbol{a} \quad . \tag{C.33}$$

Also

$$\text{Diag}\{\boldsymbol{a}\} = (\boldsymbol{a} \cdot \boldsymbol{1}^*) \odot I \quad . \tag{C.34}$$

C.1.2.2 Frobenius Norm

The Frobenius norm of a matrix A with singular values $\lambda_i(A)$ is

$$\|A\|_F \triangleq \sqrt{trace\{A^*A\}} = \sqrt{\sum_{i=1}^{\min(m,n)} \lambda_i^2(A)} \quad . \tag{C.35}$$

Similarly, the Frobenius inner product is

$$\langle A, B \rangle_F = trace\{A^* \cdot B\} \tag{C.36}$$

C.2 Matrix Calculus

This section reviews derivatives with respect to vectors and matrices. Good source for this Peterson and Pederson's The Matrix Cookbook [3], the online Complex Analysis [2], and Zhang's matrix approach to artifical intelligence [4], and the more in-depth references therein. The reader presumably understands well derivative of a real scalar function $f(x)$ with respect to a real scalar variable x, $\frac{df(x)}{dx} \triangleq f_x(x)$ and its inverse indefinite integration $x = \int f \cdot df_x \cdot df + C$ (C a constant that disappears with definite integrals that have limits). Most communication engineers will also recall the partial derivatives of a real scalar function with respect to multiple real inputs, say $f(x, y, z)$ as $\frac{\partial f}{\partial x} \triangleq f_x(x, y, z)$, $\frac{\partial f}{\partial y} \triangleq f_y(x, y, z)$, and $\frac{\partial f}{\partial z} \triangleq f_z(x, y, z)$ often organized in a gradient vector (with $\boldsymbol{x}$ now a column vector containing x, y, and z)

$$\frac{df}{d\boldsymbol{x}} \triangleq \nabla f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \quad . \tag{C.37}$$

Subsection C.2.1 further pursues this and expands to when the function itself is also a vector. Subsection C.2.2 yet further expands to when either or both of the function and the inputs are matrices.

C.2.1 Derivative with respect to a vector

While a real vector is a special case of a real matrix, first examination of derivative with respect to a vector is helpful to build intuition. A scalar function $f(\boldsymbol{x}) \in \mathbb{R}$ has derivative with respect to a vector $\boldsymbol{x} \in \mathbb{R}^{L_x}$ that is an $L_x \times 1$ **gradient** vector and there are L_x derivative components (that are the components) in $df/d\boldsymbol{x}$:

$$\nabla f_{\boldsymbol{x}} = \frac{df}{d\boldsymbol{x}} = \begin{bmatrix} f_{x_{L_x}} \\ \vdots \\ \\ f_{x_1} \end{bmatrix} \quad . \tag{C.38}$$

The gradient of an $L_y \times 1$ vector $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{L_f}$ with respect to the vector $\boldsymbol{x} \in \mathbb{R}^{L_x}$, is an $L_f \times L_x$ matrix

$$\nabla_{\boldsymbol{x}} \boldsymbol{f} = \left[\left(\frac{\partial f_{L_f}}{\partial \boldsymbol{x}} \right) \; \cdots \; \left(\frac{\partial f_1}{\partial \boldsymbol{x}} \right) \right] = \begin{bmatrix} \frac{\partial f_{L_f}}{\partial x_{L_x}} & \cdots & \frac{\partial f_1}{\partial x_{L_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{L_f}}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_1} \end{bmatrix} \quad . \tag{C.39}$$

The transpose of this matrix is sometimes called the **Jacobian** matrix $J_f = [\nabla_{\boldsymbol{x}} \boldsymbol{f}]^t$. Block vectors or "cell arrays" follow the same definitions but with possible variable elements corresponding to the individual element dimensionalities. The gradient of a scalar linear combination is

$$\nabla_{\boldsymbol{x}} (\boldsymbol{a}^t \cdot \boldsymbol{x}) = \boldsymbol{a} = \nabla_{\boldsymbol{x}} (\boldsymbol{x}^t \cdot \boldsymbol{a}) \quad , \tag{C.40}$$

while the $L_f \times L_x$ matrix derivative corresponding to an $L_f \times 1$ vector linear combination $A \cdot \boldsymbol{x}$ is (A is $L_f \times L_x$)

$$\frac{d(A \cdot \boldsymbol{x})}{d\boldsymbol{x}} = A^t \quad , \tag{C.41}$$

but the $L_f \times L_x$ gradient of the $1 \times L_f$ row-vector linear combination $\boldsymbol{x}^t \cdot B$ is (B is $L_x \times L_f$)

$$\frac{d(\boldsymbol{x}^t \cdot B)}{d\boldsymbol{x}} = B \quad . \tag{C.42}$$

C.2.2 Derivative with respect to a matrix

A scalar function's derivative with respect to a matrix is a matrix of the same dimension with the partial derivative with respect to each element of the matrix in its corresponding (same) position. A vector of course immediately also corresponds to a special case of a matrix. Again in this text: capitalized letters denote matrices (in context, sometimes capitals also correspond to Fourier, Laplace, or D transforms). Vectors are boldface lower case.

When the function itself is a vector or a matrix, then the derivatives essentially become potentially 3-dimensional and 4-dimensional tensors respectively. In these situations, typically the last sections Kronecker product instead characterizes the tensors as matrices. When this happens, typically the indices for the two function outputs $\boldsymbol{Y}$ are k, ℓ while for the function input $\boldsymbol{X}$ are i, j. For unconstrained matrices (so then NOT symmetric, positive definite, Toeplitz, etc), direct expressions can be obtained when the matrix variable X has its entries $x_{k,\ell}$ chosen so that when taking partial derivatives with respect to any matrix element $x_{i,j}$ that

$$\frac{\partial x_{k,\ell}}{\partial x_{i,j}} = \delta_{ik} \cdot \delta_{\ell j} \ . \tag{C.43}$$

Some familiar rules apply largely from scalar calculus:

1. linearity
$$\nabla_X \left[\alpha_f \cdot \boldsymbol{f}(X) + \alpha_g \cdot \boldsymbol{g}(X)\right] = \alpha_f \cdot \nabla_X \boldsymbol{f}(X) + \alpha_g \cdot \nabla_X \boldsymbol{g}(X) \tag{C.44}$$

2. Product (also true for Hadamard and Kronecker products)
$$\nabla_X \left[f(X) \cdot g(X)\right] = \nabla_X f(X) \cdot \nabla_X g(X) + g(X) \cdot \nabla_X g(X) \tag{C.45}$$

3. Quotient
$$\nabla_X \frac{f(X)}{g(X)} = \frac{1}{g^2(X)} \cdot \left[g(X) \cdot \nabla_X f - f(X) \cdot \nabla_x g(X)\right] \tag{C.46}$$

4. Chain Rule
$$\nabla_X f(g(X)) = \frac{df}{dg} \cdot \nabla_X g(X) \tag{C.47}$$

5. Chain Rule for matrix function G that is $p \times q$
$$\left[\nabla_X f(G(X))\right]_{ij} = \sum_{k=1}^{p} \sum_{\ell=1}^{q} \frac{\partial f(G)}{\partial g_{k\ell}} \cdot \frac{\partial g_{k\ell}}{x_{ij}} \ . \tag{C.48}$$

This text has interest in some common scalar functions and their derivatives with respect to matrix $\boldsymbol{X}$ (square when directly the argument of *trace* or determinant:

$$f(X) \quad | \quad \nabla_X f \tag{C.49}$$
$$- - - - - - - - \quad \quad - - - - - - - - - - - - \tag{C.50}$$
$$trace\{X\} \quad | \quad I \ \text{(square } X \text{ only)} \tag{C.51}$$
$$trace\{X^{-1}\} \quad = \quad -X^{-2} \ \text{(square } X \text{ only)} \tag{C.52}$$
$$trace\{A \cdot X\} \quad = \quad A^t \tag{C.53}$$
$$trace\{X^k\} \quad = \quad k \cdot (X^t)^{k-1} \ \text{(square } X \text{ only)} \tag{C.54}$$
$$trace\{X \cdot A \cdot X \cdot B\} \quad | \quad B^t \cdot X^t \cdot A^t + A^t \cdot X^t \cdot B^t \tag{C.55}$$
$$trace\{X \cdot A \cdot X^t \cdot B\} \quad | \quad B^t \cdot X \cdot A^t + B \cdot X \cdot A \tag{C.56}$$
$$trace\{A \cdot X \cdot X^{-t} \cdot B\} \quad | \quad (B \cdot A + A^t \cdot B^t) \cdot X \tag{C.57}$$
$$trace\{A \cdot X^{-1} \cdot B\} \quad | \quad -X^t \cdot A^t \cdot B^t \cdot X^{-t} \tag{C.58}$$
$$|X| \quad | \quad X^{-t} \cdot |X| \ \text{(square } X \text{ only)} \tag{C.59}$$

$$\ln|X| \quad | \quad X^{-t} \tag{C.60}$$
$$|X| \quad | \quad X^{-t} \cdot |X| \tag{C.61}$$
$$\left|X^{-1}\right| \quad | \quad \frac{X^{-t}}{|X|} \tag{C.62}$$
$$\left|X^{k}\right| \quad | \quad k \cdot \left|X^{k}\right| \cdot X^{-t} \tag{C.63}$$
$$\ln\left|X \cdot X^{t}\right| \quad | \quad 2 \cdot X \cdot \left[X^{t} \cdot X\right]^{-1} \cdot \left|X \cdot X^{t}\right| \tag{C.64}$$
$$\ln\left|X \cdot A \cdot X^{t}\right| \quad | \quad \left[A \cdot X^{t} \cdot \left(X^{t} \cdot A \cdot X\right)^{-1} + X \cdot A^{t} \cdot \left(X \cdot A \cdot X^{t}\right)^{-1}\right] \cdot \left|X \cdot A \cdot X^{t}\right| \tag{C.65}$$
$$|A \cdot X \cdot B| \quad | \quad |A \cdot X \cdot B| \cdot A^{t}(AXB)^{-t} \cdot B^{t} \tag{C.66}$$

If A, B, and/or X are such that any trace above becomes a scalar, then this is also the gradient of that scalar function because trace of a scalar is that scalar.

For derivatives with respect to a matrix, the function *vec* takes a matrix input column by column. There are also some vector and matrix gradients of interest (in the matrix case, Kronecker products implicity expand the dimensionality)

$$F(X) \quad | \quad \nabla_X f \tag{C.67}$$
$$- - - - - - - - - \quad \quad - - - - - - - - - - - - \tag{C.68}$$
$$X^{t} \cdot A \cdot X \quad | \quad (A^{t} \cdot X) \otimes I + K_{nm} \cdot (A \cdot X) \otimes I \tag{C.69}$$
$$A \cdot X^{t} \cdot B \quad | \quad K_{nm} \cdot (A^{t} \otimes B) \tag{C.70}$$
$$A \cdot X^{t} \cdot B \cdot X \cdot C \quad | \quad K_{nm} \cdot A^{t} \otimes (B \cdot X \cdot C) + (B^{t} X \cdot A^{t}) \otimes C \tag{C.71}$$
$$X^{-1} \quad | \quad -\left(X^{-t} \otimes X\right) \text{ (square } X \text{ only)} \tag{C.72}$$
$$X^{k} \quad | \quad \sum_{j=1}^{k} (X^{t})^{j-1} \otimes X^{k-j} \text{ (square } X \text{ only)} \tag{C.73}$$
$$\ln(X) \quad | \quad X^{-t} \otimes I \text{ (square } X \text{ only)} \tag{C.74}$$
$$e^{X} \quad | \quad \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \cdot \sum_{j=0}^{k} (X^{t})^{j} \otimes X^{k-j} \tag{C.75}$$
$$\tag{C.76}$$

The chain rule as well as some basic principles below can be used with these to construct more sophisticated matrix derivatives.

Another use of the *vec* function is to write the gradient of a scalar with respect to a matrix as an enlarged vector, creating the $mn \times 1$ gradient $\nabla_{vec(X)} f(vec(X))$. This rearranges the gradient from an $m \times n$ matrix to an $mn \times 1$ vector. This may seem semantical until viewing the gradient of a matrix function of a matrix like $p \times q$ matrix function $F(X)$ where then

$$\nabla_{vec(X)} F(X) \triangleq \frac{\partial vec(F(X))^{t}}{\partial vec(X)} \quad , \tag{C.77}$$

is an $mn \times pq$ gradient matrix, avoiding a 4-dimensional tensor with a two-dimensional matrix that is partitioned into$n \cdot q$ $m \times p$ submatrices.

Chain Rule Expressions and other Derivative Calculating Aids chain rule, products, division, distributive.

C.2.2.1 Derivative with respect to a complex variable

Complex variables and numbers are in reality just a way to simplify notation for two-dimensional-real vectors (real part and the imaginary part are both real amplitudes, but the $\jmath$ is a basis vector in a direction orthogonal to the basis vector 1), namely

$$z = x + \jmath \cdot y \quad . \tag{C.78}$$

A function f may correspondingly have nonzero real and imaginary parts so that

$$f(z) = u(z) + \jmath \cdot v(z) = u(x,y) + \jmath \cdot v(x,y) \quad , \tag{C.79}$$

essentially a 2×2 functional transformation on the real magnitudes underlying the complex-variable notational simplication. This "simplification" introduces some complication for limits, and in particularly derivatives; for instance the definition of a derivative

$$\frac{df}{dz}(z) \triangleq \lim_{\delta z \to 0} \frac{f(z+\delta z) - f(z)}{\delta z} \quad , \tag{C.80}$$

has an ambiguity in terms of from which direction $\delta z \to 0$. When exclusively real, the direction is limited to one direction; when complex, the direction could be from any angle towards z, see for instance [2]. In the real-axis approach, the derivative becomes with real h

$$\frac{df}{dz}(z) \triangleq \lim_{h \to 0} \frac{f(u(x+h,y) - f(x,y)) - u(x,y) + \jmath \cdot v(x+h,y) - v(x,y)}{\delta z} \tag{C.81}$$

$$= \frac{\partial u(x,y)}{\partial x} + \jmath \cdot \frac{\partial v(x,y)}{\partial x} \quad . \tag{C.82}$$

However, a different real g, not necessarily equal to h, could represent the approach along the imaginary axis to complex variable z and correspondingly

$$\frac{df}{dz}(z) \triangleq \lim_{g \to 0} \frac{f(u(x,y+g) - f(x,y)) - u(x,y) + \jmath \cdot v(x,y+g) - v(x,y)}{\delta z} \tag{C.83}$$

$$= \frac{\partial u(x,y)}{\partial y} + \jmath \cdot \frac{\partial v(x,y)}{\partial y} \quad . \tag{C.84}$$

Clearly we'd like these two paths into z to have the same derivative, but that only occurs if

$$\begin{aligned} \frac{\partial u(x,y)}{\partial x} &= \frac{\partial v(x,y)}{\partial y} \text{ and} \\ \frac{\partial u(x,y)}{\partial y} &= -\frac{\partial v(x,y)}{\partial y} \quad , \end{aligned} \tag{C.85}$$

the **Cauchy-Riemann** equations [3], [2]. This **Cauchy-Rieman Equation** satisfaction implies a certain symmetry of $f(z)$ with respect to all directions (which would be linear combinations of the real and imaginary paths) so that all derivatives are the same. Figure C.1 illustrates this

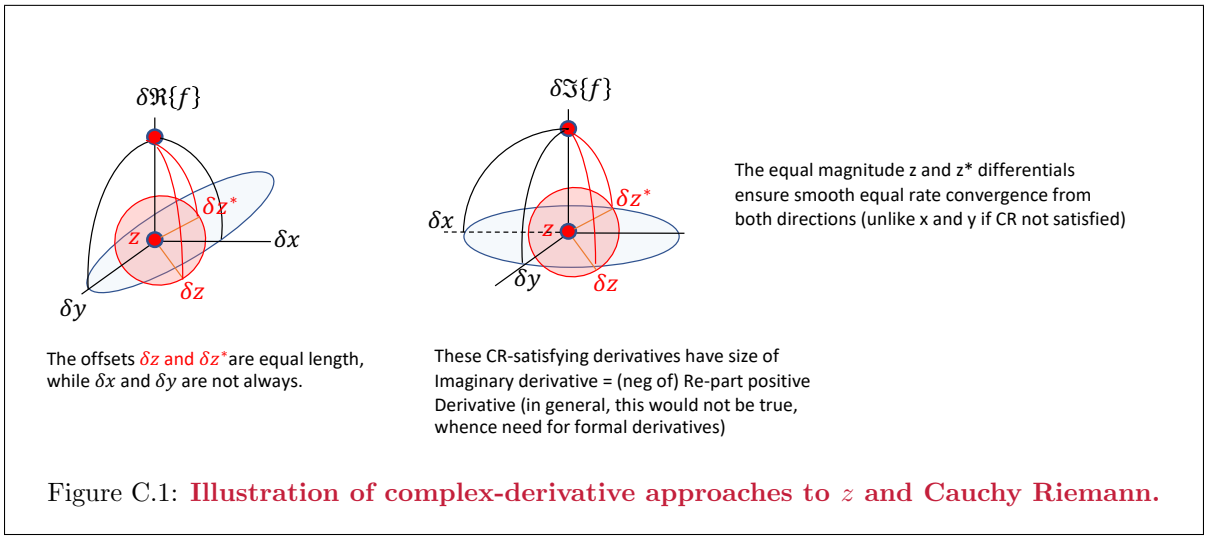

Figure C.1: **Illustration of complex-derivative approaches to z and Cauchy Riemann.**

Not all functions $f(z)$ satisfy the Cauchy-Riemann equation, and then the definition of a complex derivative becomes confused. Figure C.1 also shows that if the two approach directions are δz and δz^*, which are also orthogonal directions (clearly one is 90-degree rotation from the other) that the magnitude of the δz remains constant even if $|h| \neq |g|$. Thus a formal derivative definition that uses these two directions is an improvement and would always have two orthogonal components, one along δz and the other along $\delta z*$. For instance the function $f(z) = z^2$ satsisies CR with $z^2 = (x^2 - y^2) + \jmath(2xy)$ so that $u_x = 2x = v_y$ and $u_y = -2y = -v_x$. However, the function $f(z) = |z|^2 = x^2 + y^2$ clearly has $v = 0$ and $u_x = 2x$, $u_y = 2y$ which satisfies CR only at the origin. The latter function thus would have a confused derivative.

When the CR Equation is satisfied, the function is said to be **analytic** (See Appendix D for more on analytic functions), which means that over any input domain where CR holds, the function and all its derivatives exist (are bounded)[4]

The derivative generalization needed is the Complex Derivative and the Conjugate Derivative , showing explicitly the dependency on both z and z^* as per [4],

$$\textbf{Complex Derivative} \quad \frac{df(z,z^*)}{dz} \triangleq \frac{1}{2}\left\{\frac{\partial f(z,z^*)}{\partial \Re z} - \jmath \cdot \frac{\partial f(z,z^*)}{\partial \Im z}\right\} \tag{C.86}$$

$$\textbf{Conjugate Derivative} \quad \frac{df(z,z^*)}{dz^*} \triangleq \frac{1}{2}\left\{\frac{\partial f(z,z^*)}{\partial \Re z} + \jmath \cdot \frac{\partial f(z,z^*)}{\partial \Im z}\right\} \ . \tag{C.87}$$

Note in this case that for $f(z,z^*) = |z|^2 = z \cdot z^*$ that $\frac{df(z,z)}{dz} = z^*$ and $\frac{df(z,z^*)}{dz^*} = z$. The complex derivative and conjugate derivative always exist as long as the real functions u_x, u_y, v_x, and v_y are all well defined. The generalized complex derivative reduces to the usual derivative with the Cauchy-Reinman equation's satisfaction. It is simple to show that

$$\frac{dz}{dz^*} = \frac{dz^*}{dz} = 0 \ , \tag{C.88}$$

confirming the two approach directions to z are independent. The **complex-variable derivative's chain rule** also generalizes to

$$\frac{\partial f(w,w^*)}{\partial z} = \frac{\partial f(w,w^*)}{\partial w} \cdot \frac{\partial w(z,z^*)}{\partial z} + \frac{\partial f(w,w^*)}{\partial w^*} \cdot \frac{\partial w^*(z,z^*)}{\partial z} \tag{C.89}$$

$$\frac{\partial f(w,w^*)}{\partial z^*} = \frac{\partial f(w,w^*)}{\partial w} \cdot \frac{\partial w(z,z^*)}{\partial z^*} + \frac{\partial f(w,w^*)}{\partial w^*} \cdot \frac{\partial w^*(z,z^*)}{\partial z^*} \ , \tag{C.90}$$

which has second terms zero for analytic f.

The rules in (C.44) - (C.48) apply to both generalized gradients, as do the $F(X)$ vec-function notational generalizations

The **complex gradient vector**'s then follows by using the complex derivative for each complex-input dimension[5] (column vectors, both)

$$\nabla f_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}^{t*}) \triangleq \frac{\partial f(\boldsymbol{z}, \boldsymbol{z}^{t*})}{\partial \Re \boldsymbol{z}} + \jmath \cdot \frac{\partial f(\boldsymbol{z}, \boldsymbol{z}^{t*})}{\partial \Im \boldsymbol{z}} \tag{C.91}$$

$$\nabla f_{\boldsymbol{z}^{t*}}(\boldsymbol{z}, \boldsymbol{z}^{t*}) \triangleq \frac{\partial f(\boldsymbol{z}, \boldsymbol{z}^{t*})}{\partial \Re \boldsymbol{z}} + \jmath \cdot \frac{\partial f(\boldsymbol{z}, \boldsymbol{z}^{t*})}{\partial \Im \boldsymbol{z}} \ . \tag{C.92}$$

Similarly for matrix Z:

$$\nabla f_Z(Z, Z^{t*}) \triangleq \frac{\partial f(Z, Z^{t*})}{\partial \Re Z} + \jmath \cdot \frac{\partial f(Z, Z^{t*})}{\partial \Im Z} \tag{C.93}$$

$$\nabla f_{Z^{t*}}(Z, Z^{t*}) \triangleq \frac{\partial f(Z, Z^{t*})}{\partial \Re Z} + \jmath \cdot \frac{\partial f(Z, Z^{t*})}{\partial \Im Z} \ . \tag{C.94}$$

[4]This is somewhat subtle, but satisfaction of CR implies that all paths in must have all derivatives bounded, but only the first (partial) derivatives needed to be checked - a somewhat striking property left here to mathematicians to explain further.

[5]Note the factor of 1/2 is already implicit in the conjugate derivative definition so it need not be repeated.

These complex forms are used in gradient descent (Chapters 3 - 6) and other optimization, particularly also the conjugate gradient form $\nabla f_{\boldsymbol{z}^{t*}}(\boldsymbol{z}, \boldsymbol{z}^{t*})$ for nonanalytic function's descent.

The earlier gradients generalize in the complex case to

$f(Z)$	$\nabla_Z f$	$\nabla_{Z^{t*}} f$	
$- - - - - - - -$	$- - - - - - - - - - -$	$- - - - - - - - - - - - -$	(C.96)
$trace\{A \cdot Z\}$	A^t	0	(C.97)
$trace\{A \cdot Z^*\}$	0	A	(C.98)
$trace\{A \cdot Z^{-1}\}$	$-Z^{-t} \cdot A^t \cdot Z^{-t}$	0	(C.99)
$trace\{A \cdot Z^{-*}\}$	0	$-Z^{-*} \cdot A \cdot Z^{-*}$	(C.100)
$trace\{Z \cdot A \cdot Z^t \cdot B\}$	$B^t \cdot Z \cdot A^t + B \cdot Z \cdot A$	0	(C.101)
$trace\{Z \cdot A \cdot Z \cdot B\}$	$(A \cdot Z \cdot B + B \cdot Z \cdot A)^t$	0	(C.102)
$trace\{Z \cdot A \cdot Z^{t*} \cdot B\}$	$B^t \cdot Z^* \cdot A^t$	$A^t \cdot Z^t \cdot B^t$	(C.103)
$trace\{Z \cdot A \cdot Z^* \cdot B\}$	$B^t \cdot Z^{t*} \cdot A^t$	$B \cdot Z \cdot A$	(C.104)
$trace\{Z^k\}$	$k \cdot (Z^t)^{k-1}$	0	(C.105)
$trace\{(Z^*)^k\}$	0	$k \cdot (Z^*)^{k-1}$	(C.106)
$\lvert Z \rvert$	$Z^{-t} \cdot \lvert Z \rvert$	0	(C.107)
$\lvert Z^* \rvert$	0	$Z^{-*} \cdot \lvert Z^* \rvert$	(C.108)
$\lvert Z \cdot Z^t \rvert$	$2(Z \cdot Z^t)^{-1} \cdot Z \cdot \lvert Z \cdot Z^t \rvert$	0	(C.109)
$\lvert Z \cdot Z^{*t} \rvert$	$(Z^* \cdot Z^t)^{-1} \cdot Z^* \cdot \lvert Z \cdot Z^{t*} \rvert$	$Z^t \cdot (Z^* \cdot Z^t)^{-1} \cdot \lvert Z \cdot Z^{t*} \rvert$	(C.110)
$\lvert Z \cdot Z^* \rvert$	$(Z^{t*} \cdot Z^t)^{-1} \cdot Z^{t*} \cdot \lvert Z \cdot Z^* \rvert$	$(Z \cdot Z^*)^{-1} \cdot Z \cdot \lvert Z \cdot Z^* \rvert$	(C.111)
$\lvert Z^k \rvert$	$k \cdot \lvert Z \rvert^k \cdot Z^{-t}$	0	(C.112)

$F(Z, Z^{t*})$	$\nabla_{vec(Z)} F\left(vec(Z), vec(Z^{t*})\right)$	$\nabla_{vec(Z^{t*})} F\left(vec(Z), vec(Z^{t*})\right)$	
$F(Z, Z^{t*})$	$\nabla_{vec(Z)} F\left(vec(Z), vec(Z^{t*})\right)$	$\nabla_{vec(Z^{t*})} F\left(vec(Z), vec(Z^{t*})\right)$	(C.113)
$- - - - - - - -$	$- - - - - - - - - - - -$	$- - - - - - - - - -$	(C.114)
$\boldsymbol{a}^t \cdot \boldsymbol{z} + \boldsymbol{b}^t \boldsymbol{z}^{t*}$	$\boldsymbol{a}$	$\boldsymbol{b}$	(C.115)
$A \cdot Z \cdot B + C \cdot Z^{t*} \cdot D$	$B \otimes A^t$	$D \otimes C^t$	(C.116)
$A \cdot Z^t \cdot B + C \cdot Z^* \cdot D$	$K_{nm} \cdot (B \otimes A)$	$K_{nm} \cdot (D \otimes C^t)$	(C.117)

C.3 Matrix Functional Optimization

Global and local minima in the matrix cases (scalar function of vector/matrix, or matrix function of matrix) follow the intuition of scalar function minimization, basically the first-derivative (gradient) needs to be zero and the second derivative needs to be positive/negative for a minimum/maximum. Iterative algorithmic generation of such optima usually increments in the gradient's revealed direction of maximum descent/ascent.

Section C.2's use of the vec function expands here to simplify generalization to the matrix case.

C.3.1 The Hessian - Generalized 2nd Derivative

Recalling from Section C.2 that $\nabla_{vec(X)} f(X)$, the **Hessian matrix** for a scalar function of a matrix, $f(X)$, is the $mn \times mn$ matrix

$$\nabla^2_{vec(X)} f(X) \triangleq \frac{\partial^2 f(X)}{\partial vec(X)\, \partial vec(X)^t} \text{ real case .} \tag{C.118}$$

for the complex case with generalized derivatives, this Hessian expands properly to

$$\nabla^2_{vec(X)} f(X) \triangleq \begin{bmatrix} \frac{\partial^2 f(vec(Z), vec(Z^{t*}))}{\partial vec(Z^{t*})\, \partial vec(Z)^t} & \frac{\partial^2 f(vec(Z), vec(Z^{t*}))}{\partial vec(Z^{t*})\, \partial vec(Z)^*} \\ \\ \frac{\partial^2 f(vec(Z), vec(Z^{t*}))}{\partial vec(Z)\, \partial vec(Z)^t} & \frac{\partial^2 f(vec(Z), vec(Z^{t*}))}{\partial vec(Z)\, \partial vec(Z)^*} \end{bmatrix} \tag{C.119}$$

C.3.2 Local Optima

The extrema (local minimum and/or maximum) of the scalar function can occur when

$$\nabla_{vec(Z^{t*})} f(Z, Z^{t*}) = \mathbf{0} \ . \tag{C.120}$$

The point Z is a local minimum if

$$\nabla^2_{vec(Z), vec(Z^{t*})} f(Z, Z^{t*}) \succeq 0 \tag{C.121}$$

(unique if strict positive definite) and a local maximum if

$$\nabla^2_{vec(Z), vec(Z^{t*})} f(Z, Z^{t*}) \preceq 0 \ , \tag{C.122}$$

(unique if strict negative definite). The simplification to real case is obvious. When the Hessians are positive (negative) definite for all Z in a domain, then the resultant functional values are the global minimum (maximum) for that domain.,

C.3.3 Gradient Descent

Gradient descent algorithms essentially follow the form

$$Z \leftarrow Z - \mu \cdot \nabla_{vec(Z^{t*})} f(Z, Z^{t*}) \quad , \tag{C.123}$$

where the updating term is added if the algorithm is steepest ascent to a maximum. So called Newton's methods, deflect the gradient by the Hessian's psedoinverse so

$$Z \leftarrow Z - \mu \cdot \left[\nabla^2_{vec(Z), vec(Z^{t*})} f(Z, Z^{t*}) \right]^{-1} \cdot \nabla_{vec(Z^{t*})} f(Z, Z^{t*}) \quad , \tag{C.124}$$

and speed algorithm convergence.

Bibliography

[1] Introduction à l'Analyse des lignes Courbes algébriques Gabriel Cramer, Geneva:Europeana 1750, pp. 656-659.

[2] Complex-Analysis Book: A visual and Interactive Introduction Juan Carlos Ponce Campuzano https://complex-analysis.com/content/complex_differentiation.html Chapter 2, 2019-2022

[3] Kaare Brandt Petersen and Michael Syskind Pedersen *"The Matrix Cookbook"*. https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf . November 15, 2012.

[4] Xian-Da Zhang. *"A Matrix Algebra Approach to Artificial Intelligence"*. Springer, Sinapore. ISBN 978-981-15-2769-2. 2020.

Index

1

2

3

4

5

6

7

8