Journal of Computational Science 16 (2016) 128–139

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Numeric tensor framework: Exploiting and extending Einstein notation

Adam P. Harrison*, Dileepan Joseph

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada

ARTICLE INFO

Article history:
Received 5 December 2015
Received in revised form 27 April 2016
Accepted 7 May 2016
Available online 24 May 2016

Keywords:
Tensor algebra
Tensor computations
Tensor inversion
C++ classes
MATLAB classes

ABSTRACT

The numeric tensor (NT) framework addresses and unifies a growing body of work on high-dimensional algebra and software for technical computing. Its NT algebra exploits and extends Einstein notation, offering unmatched capabilities, including *N*-dimensional operators, associativity, commutativity, entrywise products, and linear invertibility. High-performance C++ and MATLAB NT software allows practitioners to directly program with NT algebra. The advantages of NT algebra are highlighted using the example of canonical-polyadic (CP) tensor decomposition. Corresponding dense benchmarks demonstrate that the NT software matches or surpasses leading competitors, *i.e.*, the MATLAB Tensor Toolbox, NumPy, and Blitz++, while supporting a more general set of arithmetic operations.

1. Introduction

Technical computing is a cornerstone of modern scientific practice. The wide use of matrix-vector (MV) algebra and the robustness of MV computations, used to represent and manipulate linear mappings applied to vectors, are a major manifestation of technical computing's success.

Underscoring the MV paradigm's integral role, the publication of von Neumann and Goldstine's 1947 paper [1], "Numerical Inverting of Matrices of High Order", is often credited as the starting point of modern numerical analysis [2]. Since then, MV computations have continued to mature, *e.g.*, the widely influential LAPACK and BLAS numerical libraries. In parallel with these computational advances, practitioners have keenly expressed the desire for direct support of algebra within their programming environments. As early as 1954, Backus and Herrick predicted that eventually "a programmer might not be considered too unreasonable if he were willing only to produce the formulas for the numerical solution of his problem" [3]. Arguably, the advent of operator-overloaded object-oriented programming languages, *e.g.*, C++, and very high-level languages for technical computing, *e.g.*, MATLAB, have come close to fulfilling this vision, wedding MV algebra directly with MV computations.

The MV paradigm exemplifies the intimate partnership between formalism and constructivism, *i.e.*, algebra and software, in technical computing. The crosstalk between algebra and software is so potent that Åhlander et al. advocate choosing mathematical abstractions through the lens of software engineering considerations [4]. Eriksson et al. echo this argument, asserting that constructivism and formalism inform each other [5]. Illustrating this, the Society for Industrial and Applied Mathematics has published books reframing the Fast Fourier Transform [6] and graph algorithms [7] using MV algebra, each emphasising that MV notation provides a powerful means to understand and develop these respective algorithms. This interplay is bi-directional, *e.g.*, efficient sparse computations have paved the way for practical MV-based graph techniques [7]. For these reasons, we use "technical computing framework" to describe a numeric algebra accompanied by a body of mature software routines.

Despite the success of the MV paradigm, there exists a cross-disciplinary need for operations upon high-dimensional data that fall outside its natural purview. This has spurred work on alternative algebras for high-dimensional data [12,8–11,13–20] and on software supporting the algebra of such formalisms [10,21–23,9,24–30]. These investigations are motivated by the need to perform arithmetic upon high-dimensional data, to differentiate expressions involving high-dimensional data, to execute or invert high-dimensional linear mappings, to represent multilinear or polynomial mappings, and/or to decompose high-dimensional data. Yet, while much of the cited work meets specific demands stemming from their application areas, their capabilities do not

* Corresponding author at: Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada.
E-mail addresses: adam.p.harrison@gmail.com (A.P. Harrison), dil.joseph@ualberta.ca (D. Joseph).

http://dx.doi.org/10.1016/j.jocs.2016.05.004
1877-7503/

often extend beyond those specific needs. This fracturing contributes to the impression that this body of work requires distinct sets of technical computing frameworks, which has a deleterious effect on knowledge translation and widespread use of these advancements.

To combat this lack of universality, this paper argues that an extended version of Einstein notation can serve as the foundation of a comprehensive technical computing framework for high-degree data[1]. We call this a numeric tensor (NT) framework, underscoring that the focus is on *numeric* multi-index arrays invested with a set of arithmetic operations. We identify key algebraic qualities necessary toward a universal framework, which include *N*-degree support, associativity, commutativity, entrywise products, and linear invertibility. To best satisfy these characteristics, we exploit and extend Einstein notation to develop the NT algebra of this work.

An algebra for numeric high-degree data is orphaned without effective supporting software. For this reason, we also introduce fast and efficient NT software, embodied by C++ and MATLAB libraries, called LibNT and NTToolbox, respectively. Mirroring the MV paradigm's capabilities, these libraries allow NT algebra to be used directly within a programming environment. The software employs a novel lattice data structure, which enables any combination of inner, entrywise, and outer products, as well as corresponding inverses, across the indices of two NTs. Both libraries rely on high-performance kernels and fully support any mixture of dense and sparse NT data. This paper will restrict attention to dense NTs, comparing the performance of LibNT and NTToolbox to leading software solutions [10,21,22,31].

Section 2 begins by reviewing current efforts toward developing high-degree algebras, making the case that an extended version of Einstein notation is best suited to meet the requirements of this wide-ranging body of work. Section 3 describes such a solution, which is called NT algebra. This is followed by Section 4, which outlines NT software designed to directly support the NT algebra. Finally, Section 5 concludes the work.

2. Exploiting Einstein notation

Practitioners have introduced or advocated for various algebras to perform arithmetic upon numeric high-degree data. These researchers come from backgrounds that include numerical analysis [9,24], computational chemistry [26–28], applied mathematics [32], image processing [18], computer vision [33], systems and control theory [16], signal processing [34–36], geodesy [37], econometrics [38], database and information systems [39], psychometrics [14], informatics [10,21,40], and statistics [19,20].

Kronecker algebra is a major player in such formalisms, furnishing MV algebra with a tensor product and typically operating in tandem with vectorising operations [43]. Practitioners have also extended MV algebra with entrywise operations, *e.g.*, the Hadamard and Khatri-Rao products. Like Liu and Trenkler [13], we group these entrywise matrix products together with Kronecker algebra, using our own label of extended matrix/vector (EMV) algebra.

Other algebras depart more completely from MV formalism. The *n*-mode product notation is a popular formalism often used in tensor decomposition literature [8,34,40,44]. For core operations in tensor decomposition, *e.g.*, multiplying a tensor on all sides with matrices, *n*-mode product notation supplies dedicated notation. Less-prevalent conventions exist to express general tensor-times-tensor products [10,33,45], which rely on numerals to designate which NT index is undergoing an inner product. We label this augmented version *n*-mode$^+$ notation.

Fields other than tensor decomposition have also introduced algebras with similar characteristics. R-matrix notation [15], originating from geodesy, represents one prominent example. Predating *n*-mode notation, its algebra relies on similar numerical designations. Suzuki and Shimizu's array algebra [16], introduced within systems and control, is another example with similar capabilities.

Einstein notation represents the final high-degree algebra. Enjoying deep roots in physics [42], the notation can be grouped within the larger category of index notations, which includes Tait's array algebra [17] and Antzoulatos and Sawchuck's hypermatrix algebra [18]. We outline many of its powerful conventions for numeric calculations in Section 3's discussion of NT algebra.

Table 1 outlines these high-degree algebras, which all embody certain characteristics that have proven beneficial. We focus on these characteristics in detail below. Based on this analysis, we argue that Einstein notation can serve as the algebraic foundation for a unifying high-degree technical computing framework.

2.1. N-*degree support*

MV algebra is unable to naturally represent *N*-degree linear mappings and multilinear mappings outside of bilinear forms, which has spurred the promotion of other numeric algebras. Einstein notation, along with array algebra and *n*-mode$^+$ notation, is naturally able to represent data and mappings of arbitrary degree, an obvious strength for high-degree algebras.

Opting to stay within the MV domain, EMV blurs the natural boundaries of the MV paradigm by supporting certain multilinear and *N*-degree linear mappings through flattening operations. However, flattening *N*-degree data requires settling on a lexicographic order in the notation, which elevates a consideration normally confined to software into the realm of algebra. This can create confusion. For instance, the Kronecker product itself has two different definitions, *e.g.*, see Van Loan [41] and Regalia and Mitra [46]. Or see Magnus [47] on the competing views on how to best lexicographically lay out partial derivatives of vector or matrix-valued functions. These issues magnify as the degree increases because for *N*-degree data there are *N*! vectorising options.

Many authors [9,16–18,24,36,48] have explicitly made a case against casting *N*-degree problems into the MV domain, describing some of the requisite identities, rearrangements, and manipulations as "troublesome" [16], "cumbersome" [18], and "awkward" [9]. Even those articulating how MV algebra can be extended note that the requisite complications can be avoided "if one is willing to abandon matrix notation" [49]. Thus, for applications that express mappings not naturally supported by MV algebra, there is an acute need for algebras that can innately work with *N*-degree data and mappings.

2.2. *Associativity and commutativity*

Einstein notation is unique in being completely associative and commutative, imbuing the algebra with unrivalled abilities to manipulate algebraic expressions. Authors have long recognised the importance of these characteristics. For instance, associativity is fundamental to the MV paradigm, as lacking it would be "unbearable" [50]. However, MV algebra is not commutative. This is often highlighted by champions of Einstein notation, *e.g.*, Papastavridis' particularly pugnacious quote that "whatever cosmetic or aesthetic advantages that [MV] notation may have, they are far outweighed by the merciless straightjacket of *noncommutativity*" [42].

[1] We use "degree" to describe the number of indices of an NT. Other possible choices include "dimension" and "order". Because "dimensionality" is widely used to describe the number of scalar entries in an array, "dimension" is avoided for this orthogonal purpose. Because "order" is frequently used to describe how non-zero data is arranged in a sparse tensor, which is an important concept [21], this term is also avoided.