Generalized Matrix Functions: Theory and Computation

Michele Benzi

Department of Mathematics and Computer Science

Emory University

Atlanta, Georgia, USA

NASCA‘18
Kalamata, Greece
July 2-6, 2018

Outline

Acknowledgements

- Francesca Arrigo (Strathclyde), Caterina Fenu (Cagliari)
- Jared Aurentz (ICM Madrid), Anthony Austin (Argonne National Laboratory), Vassilis Kalantzis (U. of Minnesota)
- Ru Huang (Emory)
- Support: NSF (Grant DMS-1719578)
- Thanks to the organizers for the invitation

Introduction

Generalized matrix functions were originally introduced by J. B. Hawkins and A. Ben-Israel in 1973 in an attempt to extend the notion of a matrix function to rectangular matrices. The idea was to parallel the construction of the (Moore-Penrose) generalized inverse, using the SVD.

J. B. Hawkins and A. Ben-Israel, *On generalized matrix functions*, Linear and Multilinear Algebra, 1 (1973), pp. 163–171.

This paper is purely theoretical and does not mention any applications.

Introduction (cont.)

Note that the use of the term “generalized" is somewhat misleading, since this notion of matrix function does not reduce to the usual one when A is square, except in special situations.

The name was dropped in the treatment given later in

A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, Second Ed., Springer, New York, 2003.

Introduction (cont.)

This notion has gone largely unnoticed for many years. Nevertheless, generalized matrix functions, usually unrecognized as such, have apppeared repeatedly in different contexts in the literature and do have important applications, for instance to matrix optimization and low-rank approximation problems arising in

- compressed sensing
- computer vision (photometric stereo and optical flow)
- regularization of discrete ill-posed problems
- MRI
- control theory
- complex frequency estimation

The notion of generalized matrix function also arises in the analysis of directed networks and in the computation of (standard) functions of skew-symmetric matrices.

Definition of generalized matrix function

Definition

Let $A \in \mathbb{C}^{m\times n}$ be a rank r matrix and let $A = U_r \Sigma_r V_r^*$ be its compact SVD. Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a scalar function such that $f(\sigma_i)$ is defined for all $i = 1, 2, \ldots, r$. The generalized matrix function $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ induced by f is defined as

$$f^\diamond(A) := U_r f(\Sigma_r) V_r^*,$$

where $f(\Sigma_r)$ is defined for the $r \times r$ matrix Σ_r in the standard way:

$$f(\Sigma_r) = \mathrm{diag}(f(\sigma_1), f(\sigma_2), \ldots, f(\sigma_r)).$$

Note: This notion of matrix function reduces to the usual one when A is Hermitian positive definite, or if A is positive semidefinite and f satisfies $f(0) = 0$.

Basic properties

For $A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^* = \sum_{i=1}^{r} \sigma_i E_i$, let

$$E := \sum_{i=1}^{r} E_i = U_r V_r^*.$$

Note that $EE^* = P_{R(A)}$ and $E^*E = P_{R(A^*)}$.

Proposition

Let $f, g, h : \mathbb{R} \to \mathbb{R}$ be scalar functions and let $f^\diamond, g^\diamond, h^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the corresponding generalized matrix functions. Then:

(i) *if $f(z) = k$, then $f^\diamond(A) = kE$;*

(ii) *if $f(z) = z$, then $f^\diamond(A) = A$;*

(iii) *if $f(z) = z^{-1}$, then $f^\diamond(A) = (A^\dagger)^*$;*

(iv) *if $f(z) = g(z) + h(z)$, then $f^\diamond(A) = g^\diamond(A) + h^\diamond(A)$;*

(v) *if $f(z) = g(z)h(z)$, then $f^\diamond(A) = g^\diamond(A)E^*h^\diamond(A)$.*

Basic properties (cont.)

Proposition

Let $A \in \mathbb{C}^{m\times n}$ be a matrix of rank r. Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a scalar function and let $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the induced generalized matrix function, assumed to be defined at A. Then the following properties hold true.

(i) $[f^\diamond(A)]^* = f^\diamond(A^*)$;

(ii) *let $X \in \mathbb{C}^{m\times m}$ and $Y \in \mathbb{C}^{n\times n}$ be two unitary matrices, then* $f^\diamond(XAY) = X[f^\diamond(A)]Y$;

(iii) *if $A = \mathrm{diag}(A_{11}, A_{22}, \ldots, A_{kk})$, then*

$$f^\diamond(A) = \mathrm{diag}(f^\diamond(A_{11}), f^\diamond(A_{22}), \ldots, f^\diamond(A_{kk}));$$

(iv) $f^\diamond(I_k \otimes A) = I_k \otimes f^\diamond(A)$;

(v) $f^\diamond(A \otimes I_k) = f^\diamond(A) \otimes I_k$.

Basic properties (cont.)

The next three propositions describe the relation between generalized and standard matrix functions.

Proposition

Let $A \in \mathbb{C}^{m\times n}$ be a rank r matrix and let $f : \mathbb{R}_+ \to \mathbb{R}$ be a scalar function. Let $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the induced generalized matrix function. Then

$$f^\diamond(A) = \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i}\mathbf{u}_i\mathbf{u}_i^*\right) A = A\left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i}\mathbf{v}_i\mathbf{v}_i^*\right), \tag{1a}$$

or, equivalently,

$$f^\diamond(A) = f(\sqrt{AA^*})(\sqrt{AA^*})^\dagger A = A(\sqrt{A^*A})^\dagger f(\sqrt{A^*A}). \tag{1b}$$

Basic properties (cont.)

Proposition

Let $A \in \mathbb{C}^{m\times n}$ have the polar decomposition

$$A = PH$$

with $P \in \mathbb{C}^{m\times n}$ having orthonormal columns and $H \in \mathbb{C}^{n\times n}$ Hermitian positive semidefinite. If $f^\diamond(A)$ is defined, then

$$f^\diamond(A) = Pf(H)\,, \tag{2}$$

where $f(H)$ is a standard matrix function of H.

Note: recall that $H = \sqrt{A^*A}$.

Basic properties (cont.)

Proposition

Let f be defined on $A = M^*M$ where $M \in \mathbb{C}^{r\times n}$ has rank $r \le n$; if $r < n$ we assume that $f(0) = 0$. If $M = U_r\Sigma_r V_r^*$, then

$$f(A) = V_r g(\Sigma_r)^2 V_r^* = [V_r g(\Sigma_r) U_r^*]\,[U_r g(\Sigma_r) V_r^*] = [g^\diamond(M)]^*\,[g^\diamond(M)]\,, \tag{3}$$

where $g(x) := \sqrt{f(x^2)}$.

Note that $g = f$ when $f(x) = x^\alpha$, $\alpha \in \mathbb{R}$.

Basic properties (cont.)

When f is analytic, it is also possible to express $f^\diamond(A)$ in terms of *generalized power series* and also in terms of contour integrals. We will not need these representations here.

The notion of generalized matrix function also extends to compact operators on infinite-dimensional separable Hilbert spaces (since the SVD does).

F. Andersson, M. Carlsson, and K.-M. Perfekt, *Operator-Lipschitz estimates for the singular value functional calculus*, Proceedings of the AMS, 144(5), pp. 1867–1875, 2016.

Structure preservation

When designing numerical methods for computing matrix functions it is often useful to know in advance whether certain structural properties of A are preserved. For instance, it is well known that a (standard) matrix function $B = f(A)$ is triangular, or circulant, if A is triangular or circulant.

Another important result is the fact that the matrix exponential maps skew-Hermitian matrices to unitary ones, or Hamiltonian matrices to symplectic ones. Here the original matrix structure is not preserved, but it is transformed into another structure by a specific function, in this case the exponential.

It turns out that while the triangular form is not preserved by GMFs, many other important structural properties are.

Structure preservation (cont.)

In the following, $\mathcal{M}$ denotes one of the following matrix classes:

Normal, Hermitian, Skew-Hermitian, Pseudo-Hermitian, Pseudo-skew-Hermitian, Symmetric, Skew-symmetric, Pseudo-symmetric, Pseudo-skew-symmetric, Persymmetric, Perskew-symmetric, Hamiltonian, Skew-Hamiltonian, J-Jermitian, J-skew-Hermitian, J-symmetric, J-skew-symmetric, Centro-Hermitian, Centro-skew-Hermitian, Circulant, Block Circulant with Circulant Blocks.

Proposition

Let $A \in \mathcal{M}$ and assume that $f^{\diamond}(A)$ is well-defined. Then $f^{\diamond}(A) \in \mathcal{M}$.

Structure preservation (cont.)

In other cases a given structure is preserved provided that f satisfies certain conditions.

For example, let $\mathcal{Q}$ denote one of the following matrix classes:

Orthogonal, Pseudo-orthogonal, Unitary, Pseudo-unitary, Symplectic, Conjugate Symplectic.

Proposition

Let $A \in \mathcal{Q}$ and assume that $f^{\diamond}(A)$ is well-defined. If f satisfies $f(\frac{1}{x})f(x) = 1$ for $x > 0$, then $f^{\diamond}(A) \in \mathcal{Q}$.

The proposition applies to all matrix powers $f(x) = x^{\alpha}$, for α real.

Structure preservation (cont.)

Proposition

(1) Let A be real and nonnegative. If f is the odd part of an analytic function $f(x) = \sum_{k=0}^{\infty} c_k x^k$ with $c_{2k+1} \geq 0$ for $k = 0, 1, \ldots$ and $f^{\diamond}(A)$ is well-defined, then $f^{\diamond}(A)$ is a nonnegative matrix.

(2) Let A be doubly stochastic. If f is as in (1) and in addition $f(1) = 1$, then $f^{\diamond}(A)$ is also doubly stochastic.

Functions of block matrices

In network science and in the numerical solution of systems of ODEs it is required io compute (standard) matrix functions of matrices of the form

$$\mathcal{A} = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \quad \text{or} \quad \mathcal{B} = \begin{bmatrix} 0 & -A \\ A^* & 0 \end{bmatrix}.$$

In particular, exponentials of matrices of the form $\mathcal{A}$ arise in the analysis of bipartite and directed networks, and exponentials of matrices of the form $\mathcal{B}$ arise, for example, in the numerical integration of the wave equation, the Korteveg-de Vries equation, and other Hamiltonian systems.

M. Benzi, E. Estrada, and C. Klymko, *Ranking hubs and authorities using matrix functions*, LAA, 438 (2013), pp. 2447–2474.

N. Del Buono, L. Lopez, and R. Peluso, *Computation of the exponential of large sparse skew-symmetric matrices*, SISC, 27 (2005), pp. 278–293.

Functions of block matrices (cont.)

It is easy to check that the exponentials of $\mathcal{A}$ and $\mathcal{B}$ are given by

$$\exp(\mathcal{A}) = \begin{bmatrix} \cosh(\sqrt{AA^*}) & \sinh^\diamond(A) \\ \sinh^\diamond(A^*) & \cosh(\sqrt{A^*A}) \end{bmatrix}$$

and

$$\exp(\mathcal{B}) = \begin{bmatrix} \cos(\sqrt{AA^*}) & -\sin^\diamond(A) \\ \sin^\diamond(A^*) & \cos(\sqrt{A^*A}) \end{bmatrix},$$

respectively. Similar expressions hold for other functions of $\mathcal{A}$ and $\mathcal{B}$, leading to even functions of $\sqrt{AA^*}$ and $\sqrt{A^*A}$ on the diagonal and to generalized odd functions of A and A^* in the off-diagonal positions.

Hence, generalized matrix functions occur as submatrices of (standard) functions of certain block 2×2 matrices.

Functions of block matrices (cont.)

Proposition

Let $A \in \mathbb{C}^{m\times n}$ *and*

$$\mathcal{A} = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}.$$

If f *is defined at* $\mathcal{A}$*, then*

$$f(\mathcal{A}) = \begin{bmatrix} f_{even}(\sqrt{AA^*}) & f_{odd}^{\diamond}(A) \\ f_{odd}^{\diamond}(A^*) & f_{even}(\sqrt{A^*A}) \end{bmatrix}, \qquad (4)$$

where $f = f_{even} + f_{odd}$ *is the decomposition of a function in its even and odd parts,* $f_{even}(x) = \frac{1}{2}(f(x) + f(-x))$ *and* $f_{odd}(x) = \frac{1}{2}(f(x) - f(-x))$*.*

Functions of block matrices (cont.)

It follows that if f is an odd function, then

$$f(\mathcal{A}) = \begin{bmatrix} 0 & f^{\diamond}(A) \\ f^{\diamond}(A)^* & 0 \end{bmatrix}. \tag{5}$$

Note that every function $f(x)$ defined on the positive singular values of A can be extended to an odd function defined on the eigenvalues of $\mathcal{A}$ by setting

$$f(-\sigma_i) = -f(\sigma_i), \qquad f(0) = 0.$$

With this definition of f, formula (5) can always be used. It can also be regarded as an alternative definition of $f^{\diamond}(A)$.

Two applications

Let A be the adjacency matrix of a directed network $G = (V, E)$, possibly weighted. The total hub communicability of node $i \in V$ is given by

$$C_H(i) = [f^\diamond(A)\mathbf{1}]_i = \mathbf{e}_i^T f^\diamond(A)\mathbf{1}\,,$$

where $f(x) = \sinh(x)$ and $\mathbf{1}$ denotes the vector of all ones. It is a measure of how well node i communicates with other nodes in the networks, when regarded as a broadcaster of information ('hub').

Hence, the vector $f^\diamond(A)\mathbf{1}$ (column sums of $f^\diamond(A)$) contains the total hub communicabilities of all nodes in the network.

Likewise, The total authority communicability of node $i \in V$ is given by

$$C_A(i) = [f^\diamond(A^T)\mathbf{1}]_i = \mathbf{1}^T f^\diamond(A)\,\mathbf{e}_i\,.$$

It is a measure of how well node i communicates with other nodes in the networks, when regarded as a receiver of information ('authority').

Hence, the vector $\mathbf{1}^T f^\diamond(A)$ (row sums of $f^\diamond(A)$) contains the total authority communicabilities of all nodes in the network.

Two applications (cont.)

Let now $G = (V, E)$ be an undirected network and let L be the graph Laplacian, $L = D - A$, of G. The wave equation on G is the $n \times n$ system of second order ODEs

$$\ddot{\mathbf{u}}(t) = -L\mathbf{u}(t), \tag{6}$$

together with prescribed initial conditions $\mathbf{u}(0) = \mathbf{u}_0$, $\dot{\mathbf{u}}(0) = \mathbf{v}_0$.

Writing $L = BB^T$ where B is the $n \times m$ incidence matrix of the network and defining $\dot{\mathbf{v}}(t) = B^T\mathbf{u}(t)$ we can rewrite the second-order problem (6) as the first-order system in $n + m$ unknowns

$$\begin{bmatrix} \dot{\mathbf{u}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix} = \begin{bmatrix} 0 & -B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}(t) \\ \mathbf{v}(t) \end{bmatrix}. \tag{7}$$

We emphasize that here B is rectangular, in general.

Two applications (cont.)

Taking the exponential yields the solution of (7):

$$\begin{bmatrix} \mathbf{u}(t) \\ \mathbf{v}(t) \end{bmatrix} = \begin{bmatrix} \cos(t\sqrt{BB^T}) & -\sin^\diamond(tB) \\ \sin^\diamond(tB^T) & \cos(t\sqrt{B^TB}) \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{v}_0 \end{bmatrix}.$$

Hence, the solution of the wave equation is given by

$$\mathbf{u}(t) = \cos(t\sqrt{BB^T})\mathbf{u}_0 - \sin^\diamond(tB)\mathbf{v}_0$$

which, in the special case $\mathbf{u}_0 = \mathbf{0}$, reduces to

$$\mathbf{u}(t) = -\sin^\diamond(tB)\mathbf{v}_0.$$

Typically, this expression has to be computed for different values of t.

Algorithms

Many problems involving generalized matrix functions boil down to one of the following problems:

- Computing $f^\diamond(A)\mathbf{v}$ for a given matrix A and vector $\mathbf{v}$;
- Evaluating sesquilinear forms of the type $\mathbf{z}^* f^\diamond(A)\mathbf{w}$ for given A, $\mathbf{w}$ and $\mathbf{z}$;
- Block variants of the first two problems.

We have developed two approaches for solving these problems, one based on Gaussian quadrature and Golub–Kahan bidiagonalization, the other one based on Chebyshev polynomial interpolation.

F. Arrigo, M. Benzi, and C. Fenu, *Computation of generalized matrix functions*, SIAM Journal on Matrix Analysis and Applications, 37(3), pp. 836–860, 2016.

J. Aurentz, A. Austin, M. Benzi, and V. Kalantzis, *Stable computation of generalized matrix functions via polynomial interpolation*, Preprint, submitted to SIMAX (2018).

Algorithms (cont.)

The choice of method depends on the properties of the function f and on the distribution of the singular values of A.

The Golub–Kahan–Lanczos (GKL) type methods are expected to converge rapidly when f takes on large values on the extreme singular values of A and A is well-conditioned (no tiny singular values). This is the case of $f(x) = \sinh(x)$, for example.

On the other hand, such methods cannot be expected to work well if f is not small on the interior singular values of A. In this case Chebyshev polynomial based interpolation can be expected to give good results. This is the case of $f(x) = \sin(x)$, for example. For this example small singular values are not a problem, since $\sin(x) \approx 0$ for $x \approx 0$.

Gaussian quadrature

Consider the computation of the following sesquilinear form:

$$\mathbf{z}^* f^\diamond(A)\mathbf{w}. \tag{8}$$

Owing to the identities

$$\mathbf{z}^* f^\diamond(A)\mathbf{w} = \mathbf{z}^* \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{u}_i \mathbf{u}_i^* \right) \widetilde{\mathbf{w}} = \widetilde{\mathbf{z}}^* \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{v}_i \mathbf{v}_i^* \right) \mathbf{w},$$

where $\widetilde{\mathbf{w}} = A\mathbf{w}$, and $\widetilde{\mathbf{z}} = A^*\mathbf{z}$, we can rewrite (8) as

$$\mathbf{z}^* f^\diamond(A)\mathbf{w} = \widetilde{\mathbf{z}}^* \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{v}_i \mathbf{v}_i^* \right) \mathbf{w} = \widetilde{\mathbf{z}}^* g(A^*A)\mathbf{w}, \tag{9a}$$

$$\mathbf{z}^* f^\diamond(A)\mathbf{w} = \mathbf{z}^* \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{u}_i \mathbf{u}_i^* \right) \widetilde{\mathbf{w}} = \mathbf{z}^* g(AA^*)\widetilde{\mathbf{w}}, \tag{9b}$$

where in both cases $g(t) = (\sqrt{t})^{-1} f(\sqrt{t})$.

Gaussian quadrature (cont.)

Note that if $\mathbf{z}, \mathbf{w}$ are vectors such that $\widetilde{\mathbf{z}} \neq \mathbf{w}$, then we can use the polarization identity:

$$\widetilde{\mathbf{z}}^* g(A^*A)\mathbf{w} = \frac{1}{4}\left[(\widetilde{\mathbf{z}}+\mathbf{w})^* g(A^*A)(\widetilde{\mathbf{z}}+\mathbf{w}) - (\widetilde{\mathbf{z}}-\mathbf{w})^* g(A^*A)(\widetilde{\mathbf{z}}-\mathbf{w})\right]$$

to reduce the evaluation of the sesquilinear form of interest to the evaluation of two Hermitian forms.

Hence, we can assume that $\widetilde{\mathbf{z}} = \mathbf{w}$.

See G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, Princeton, NJ, 2010.

Gaussian quadrature (cont.)

Let $\widetilde{\mathbf{z}} = \mathbf{w}$ be a unit vector (i.e., $\|\mathbf{w}\|_2 = 1$). We can rewrite the quantity (9a) as a Riemann–Stieltjes integral using the eigendecomposition of A^*A:

$$\mathbf{w}^* g(A^*A)\mathbf{w} = \mathbf{w}^* V_r g(\Sigma_r^2) V_r^* \mathbf{w} = \sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} |\mathbf{v}_i^* \mathbf{w}|^2 = \int_{\sigma_r^2}^{\sigma_1^2} g(t)\, d\alpha(t),$$

where $\alpha(t)$ is a piecewise constant step function with jumps at the positive eigenvalues $\{\sigma_i^2\}_{i=1}^r$ of A^*A, defined as follows:

$$\alpha(t) = \begin{cases} 0, & \text{if } t < \sigma_r^2 \\ \sum_{i=j+1}^{r} |\mathbf{v}_i^* \mathbf{w}|^2, & \text{if } \sigma_{j+1}^2 \le t < \sigma_j^2 \\ \sum_{i=1}^{r} |\mathbf{v}_i^* \mathbf{w}|^2, & \text{if } t \ge \sigma_r^2. \end{cases}$$

We use Gaussian quadrature to approximate the above Stieltjes integral.

Gaussian quadrature (cont.)

In turn, the quadrature formulas can be directly obtained from the Golub–Kahan bidiagonalization of A, with starting vector $\mathbf{w}$. That is, the quadratic form (Stieltjes integral) is approximated with an ℓ point Gauss quadrature rule given by the following expression:

$$\mathcal{G}_\ell := \mathbf{e}_1^T g\left(B_\ell^* B_\ell\right)\mathbf{e}_1 = \mathbf{e}_1^T \left(\sqrt{B_\ell^* B_\ell}\right)^\dagger f\left(\sqrt{B_\ell^* B_\ell}\right)\mathbf{e}_1, \tag{10}$$

where B_ℓ is the bidiagonal matrix obtained after ℓ steps of Golub–Kahan bidiagonalization of A, with starting vector $\mathbf{w}$.

One can also prescribe some of the nodes (Gauss–Radau/Gauss–Lobatto rules).

Gaussian quadrature (cont.)

After ℓ steps, the Golub–Kahan bidiagonalization of the matrix A with initial vector $\mathbf{w}$ yields the decompositions

$$AQ_\ell = P_\ell B_\ell, \qquad A^* P_\ell = Q_\ell B_\ell^T + \gamma_\ell \mathbf{q}_\ell \mathbf{e}_\ell^T, \tag{11}$$

where the matrices

$$Q_\ell = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{\ell-1}] \in \mathbb{C}^{n\times\ell} \quad \text{and} \quad P_\ell = [\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{\ell-1}] \in \mathbb{C}^{m\times\ell}$$

have orthonormal columns, the matrix

$$B_\ell = \begin{pmatrix} \omega_1 & \gamma_1 & & \\ & \ddots & \ddots & \\ & & \omega_{\ell-1} & \gamma_{\ell-1} \\ & & & \omega_\ell \end{pmatrix} \in \mathbb{R}^{\ell\times\ell}$$

is upper bidiagonal, and the first column of Q_ℓ is $\mathbf{w}$.

Gaussian quadrature (cont.)

All the $\{\gamma_j\}_{j=1}^{\ell-1}$ and $\{\omega_j\}_{j=1}^{\ell}$ can be assumed to be real and nonzero. With this assumption, the CSVD of the bidiagonal matrix B_ℓ coincides with its SVD:

$$B_\ell = \mathcal{U}_\ell \Theta_\ell \mathcal{V}_\ell^T,$$

where $\mathcal{U}_\ell = [\upsilon_1, \upsilon_2, \ldots, \upsilon_\ell] \in \mathbb{R}^{\ell\times\ell}$ and $\mathcal{V}_\ell = [\nu_1, \nu_2, \ldots, \nu_\ell] \in \mathbb{R}^{\ell\times\ell}$ are orthogonal, and $\Theta_\ell = \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_\ell) \in \mathbb{R}^{\ell\times\ell}$.

Combining the equations in (11) leads to

$$A^*AQ_\ell = Q_\ell B_\ell^T B_\ell + \gamma_\ell \omega_\ell \mathbf{q}_\ell \mathbf{e}_\ell^T,$$

where $\mathbf{q}_\ell$ denotes the Lanczos vector computed at iteration $\ell+1$. The matrix

$$T_\ell = B_\ell^T B_\ell$$

is thus symmetric and tridiagonal and coincides (in exact arithmetic) with the matrix obtained when the Lanczos algorithm is applied to A^*A.

Gaussian quadrature (cont.)

The previous approach requires computing the standard matrix function $f(\sqrt{T_\ell})$, with $T_\ell = B_\ell^* B_\ell$ tridiagonal. In alternative, we can work directly with B_ℓ:

Proposition

The ℓ-point Gauss quadrature rule $\mathcal{G}_\ell$ is given by

$$\mathcal{G}_\ell = \mathbf{e}_1^T B_\ell^\dagger f^\diamond(B_\ell)\mathbf{e}_1, \qquad \textit{if } \ \widetilde{\mathbf{z}} = \mathbf{w},$$

or

$$\mathcal{G}_\ell = \mathbf{e}_1^T f^\diamond(B_\ell) B_\ell^\dagger \mathbf{e}_1, \qquad \textit{if } \ \mathbf{z} = \widetilde{\mathbf{w}}.$$

While mathematically equivalent, the two approaches can behave rather differently in practice.

Gaussian quadrature (cont.)

A third approach uses the Golub–Kahan decomposition $A = P_r B_r Q_r^*$:

$$\mathbf{z}^* f^\diamond(P_r B_r Q_r^*)\mathbf{w} = \mathbf{z}^* f^\diamond(P_r \mathcal{U}_r \Sigma_r \mathcal{V}_r^T Q_r^*)\mathbf{w} = \mathbf{z}^*(P_r \mathcal{U}_r) f(\Sigma_r)(Q_r \mathcal{V}_r)^* \mathbf{w},$$

hence $\mathbf{z}^* f^\diamond(A)\mathbf{w} = \widehat{\mathbf{z}}^* f^\diamond(B_r)\mathbf{e}_1$, with $\widehat{\mathbf{z}} = P_r^* \mathbf{z}$ and $Q_r^* \mathbf{w} = \mathbf{e}_1$.

Assume now that $\ell < r$. We can truncate the bidiagonalization process and approximate $f^\diamond(A)\mathbf{w}$ as

$$f^\diamond(A)\mathbf{w} \approx P_\ell f^\diamond(B_\ell)\mathbf{e}_1$$

and then obtain the approximation to the bilinear form of interest as

$$\mathbf{z}^* f^\diamond(A)\mathbf{w} \approx \mathbf{z}^* P_\ell f^\diamond(B_\ell)\mathbf{e}_1.$$

The quality of the approximation will depend in general on the distribution of the singular values of A and on the particular choice of f.

Chebyshev polynomial interpolation

The basic idea is to approximate f by an odd polynomial obtained by interpolating f at Chebyshev nodes, and then to approximate $f^\diamond(A)$ by the corresponding generalized matrix polynomial. Note that this requires first scaling A so that its largest singular value is 1, since the Chebyshev approximation works on the interval $[-1, 1]$.

More precisely, after scaling we approximate $f^\diamond(A)\mathbf{v}$ by a (generalized) polynomial of degree $2k + 1$,

$$p_k^\diamond(A)\mathbf{v} = \left(\sum_{i=0}^{k} \alpha_{2i+1} q_i(AA^*)\right) A\mathbf{v}.$$

Here q_i is a polynomial such that $T_{2i+1}(x) = q_i(x^2)x$ where T_{2i+1} is the Chebsyhev polynomial of degree $2i + 1$.

The evaluation of this polynomial can be performed in a backward stable manner by means of a version of Clenshaw's algorithm. See the paper for details and proofs.

Chebyshev polynomial interpolation (cont.)

Generally speaking, the Chebyshev-based approach requires far less storage and has more inherent parallelism than the GKL-base approach (which needs reorthogonalization). However, it requires knowledge of the largest singular value of A, which can be estimated by a few steps of GKL.

In our test problems, the number of GKL steps necessary to estimate $\sigma_1(A)$ was generally small compared to the number of steps necessary to obtain a sufficiently good approximation of $f^\diamond(A)\mathbf{w}$, at least when the function f was of the form $f(x) = \sin(tx)$ (that is, not negligible on the interior singular values).

The package Chebfun was used to construct a Chebyshev interpolant of degree large enough to ensure a relative error (in the sup norm) less than 10^{-5}.

Numerical experiments

We present some total communicability computations on two directed networks: ITwiki and SLASHDOT. Here $m = n$ and the computed quantities are row sums:

$$C(i) = [\sinh^{\diamond}(A)\mathbf{1}]_i = \mathbf{e}_i^T \sinh^{\diamond}(A)\mathbf{1}\,.$$

ITwiki is the Italian Wikipedia. Its adjacency matrix A is $49,728 \times 49,728$ and has $941,425$ nonzeros, and there is a link from node i to node j in the graph if page i refers to page j.

SLASHDOT is a social news website on science and technology (aka “news for nerds"). There is a connection from node i to node j if user i indicated user j as a friend or a foe. Its adjacency matrix A is $82,168 \times 82,168$ matrix with $948,464$ nonzeros.

Numerical experiments (cont.)

We approximate $C(i)$ for ten different choices of i using Gauss quadrature with ℓ nodes using the stopping criterion

$$\mathcal{R}_\ell = \frac{\left|x^{(\ell)} - x^{(\ell-1)}\right|}{\left|x^{(\ell)}\right|} \leq \mathsf{tol}$$

and $x^{(\ell)}$ is the approximation to $C(i)$ obtained with ℓ steps of the method being tested.

We also check the relative error

$$\mathcal{E}_\ell = \frac{|x^{(\ell)} - C(i)|}{|C(i)|},$$

where $C(i)$ is the “exact" quantity, computed using $\gg \ell$ terms.

The following computations were carried out with MATLAB Version 7.10.0.499 (R2010a) 64-bit for Linux, in double precision arithmetic, on an Intel Core i5 computer with 4 GB RAM.

Results for $f(x) = \sinh(x)$, network ITwiki

Table: Network: ITwiki, $f(x) = \sinh(x)$ (tol $= 10^{-6}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	5	3.88e-08	5	2.90e-08	6	8.02e-09
2	10	4.72e-05	9	4.68e-05	7	1.27e-08
3	5	3.20e-08	5	3.17e-08	6	7.01e-09
4	7	2.31e-05	9	2.33e-05	8	4.31e-09
5	8	4.20e-05	20	5.77e-05	8	5.91e-09
6	9	2.19e-04	24	2.13e-04	8	2.70e-08
7	6	4.26e-07	6	5.85e-07	7	3.15e-09
8	14	1.91e-04	29	2.24e-04	8	3.38e-09
9	5	8.57e-08	5	9.31e-08	6	5.07e-09
10	9	9.36e-06	8	1.12e-05	8	3.22e-10

Results for $f(x) = \sinh(x)$, network SLASHDOT

Table: Network: SLASHDOT, $f(x) = \sinh(x)$ $(\mathrm{tol} = 10^{-6})$.

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	6	4.31e-07	6	5.61e-07	9	2.45e-08
2	9	3.24e-05	15	2.26e-06	9	1.56e-08
3	7	1.24e-06	8	1.75e-06	9	1.04e-07
4	14	2.21e-04	8	2.12e-04	10	1.74e-08
5	7	2.24e-05	7	2.35e-05	10	5.16e-09
6	10	4.84e-04	19	3.72e-04	10	1.99e-08
7	7	1.20e-06	7	1.20e-06	9	6.47e-08
8	7	7.11e-07	7	7.66e-07	9	7.68e-09
9	7	5.53e-06	7	5.98e-06	9	1.32e-09
10	6	6.98e-07	6	4.92e-07	8	8.68e-09

For this function and rhese graphs, the Chebyshev-based approach was found to be non-competitive with these three methods.

Numerical experiments (cont.)

Next, we present some results for the computation of $\sin^\diamond(tB)\mathbf{v}$, where B is the incidence matrix of a network. Here B is rectangular. This problem arises in the numerical solution of the wave equation on networks. Two values of the time t are considered, $t = 1$ and $t = 4$.

Network	Number of nodes	Number of edges
`SNAP/ca-HepTh`	9,877	25,998
`Newman/as-22july06`	22,963	48,436
`Gleich/usroads-48`	126,146	161,950
`SNAP/as-Skitter`	1,696,415	11,095,298
`DIMACS10/delaunay_n24`	16,777,216	50,331,601

Table: Test matrices for integrating the graph wave equation, along with the dimensions (number of nodes $\times$ number of edges) of their incidence matrices.

Relative runtime comparison for $f(x) = \sin(x)$

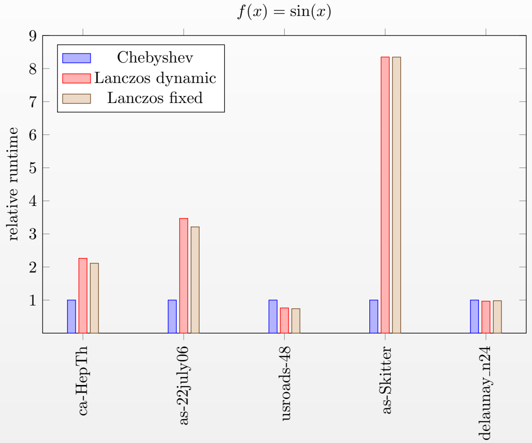

Figure 1: Plot of relative runtimes for approximating $\sin^{\diamond}(B)w$ using the three methods.

Relative runtime comparison for $f(x) = \sin(4x)$

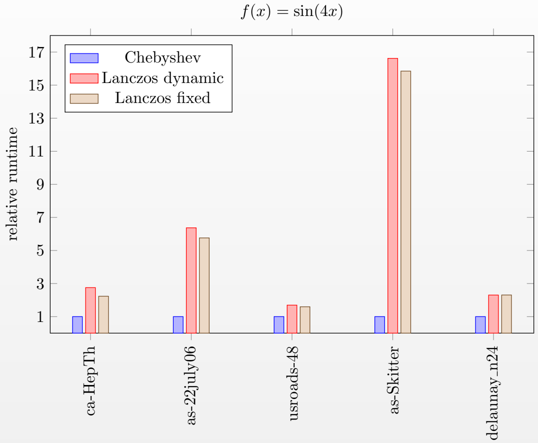

Figure 1: Plot of relative runtimes for approximating $\sin^{\diamond}(4B)w$ using the three methods.

Storage comparison for $f(x) = \sin(x)$

	Chebyshev		Lanczos		
Matrix	Degree	Memory	Steps	Memory	Ratio
`SNAP/ca-HepTh`	19	703 KB	10	2.9 MB	4.1
`Newman/as-22july06`	69	1.3 MB	25	14.3 MB	10.6
`Gleich/usroads-48`	11	4.9 MB	6	13.8 MB	2.8
`SNAP/as-Skitter`	221	280 MB	88	9.0 GB	32.3
`DIMACS10/delaunay_n24`	15	1.3 GB	7	3.6 GB	2.8

Table: Comparison of memory requirements for the Chebyshev and Lanczos based methods for $f(x) = \sin(x)$. The last column shows the ratio of memory required for the Lanczos-based method to that for the Chebyshev method.

Storage comparison for $f(x) = \sin(4x)$

	Chebyshev		Lanczos		
Matrix	Degree	Memory	Steps	Memory	Ratio
`SNAP/ca-HepTh`	51	703 KB	24	6.9 MB	9.8
`Newman/as-22july06`	229	1.3 MB	60	34.3 MB	25.4
`Gleich/usroads-48`	23	4.9 MB	12	27.7 MB	5.6
`SNAP/as-Skitter`	811	280 MB	240	24.6 GB	87.8
`DIMACS10/delaunay_n24`	37	1.3 GB	17	9.1 GB	6.8

Table: Comparison of memory requirements for the Chebyshev and Lanczos based methods for $f(x) = \sin(4x)$. The last column shows the ratio of memory required for the Lanczos-based method to that for the Chebyshev method

These comparisons were performed using MATLAB on a single compute node on the Mesabi Linux cluster at UMN, with two Intel Haswell E5-2680v3 processors and 64 GB of memory.

Conclusions and future work

The upshot:

- Generalized matrix functions arise in several important applications and are interesting mathematical objects to study.
- For large, sparse matrices, generalized matrix functions can be approximated via Gauss quadrature rules and Golub–Kahan bidiagonalization: convergence will be fast if f is large on the extreme singular values of A, and small on the rest.
- Chebyshev interpolation works better for more general functions (not small on the interior singular values).
- Chebyshev approach provably backward stable if properly implemented.
- Chebyshev also better in terms of storage and parallelism.