Exploiting Localization in Matrix Computations
II. Functions of Matrices

Michele Benzi

Department of Mathematics and Computer Science

Emory University

Atlanta, Georgia, USA

CIME-EMS Summer School in Applied Mathematics

Exploiting Hidden Structure in Matrix Computations:
Theory and Applications

Cetraro, 22-26 June, 2015

Outline

Outline

Important examples

Let $A \in \mathbb{C}^{n\times n}$, and let $z \notin \sigma(A)$. The *resolvent* of A at z is defined as

$$R(A; z) = (zI - A)^{-1}.$$

The resolvent is central to the definition of matrix functions via the contour integral approach. As a special case, the resolvent can be used to define the spectral projector onto the eigenspace of a matrix or operator corresponding to an isolated eigenvalue $\lambda_0 \in \sigma(A)$:

$$P_{\lambda_0} := \frac{1}{2\pi i} \int_{|z-\lambda_0|=\varepsilon} (zI - A)^{-1} \mathrm{d}z$$

where $\varepsilon > 0$ is small enough so that no other eigenvalue of A falls within ε of λ_0.

Remarks: More generally, one can define the spectral projector onto the invariant subspace of A corresponding to a set of selected eigenvalues by integrating $R(A; z)$ along a countour surrounding those eigenvalues and excluding the others.

The spectral projector is orthogonal if and only if A is normal.

Important examples (cont.)

Also extremely important is the matrix exponential

$$\mathrm{e}^{A} = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!}A^k$$

which is defined for arbitrary $A \in \mathbb{C}^{n\times n}$.

Just as the resolvent is central to spectral theory, the matrix exponential is fundamental to the solution of differential equations. For example, the solution to the inhomogeneous system

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay + f(t,y), \quad y(0) = y_0, \quad y \in \mathbb{C}^n, \quad A \in \mathbb{C}^{n\times n}$$

is given (implicitly!) by

$$y(t) = \mathrm{e}^{tA}y_0 + \int_0^t \mathrm{e}^{A(t-s)}f(s,y(s))\mathrm{d}s.$$

In particular, $y(t) = \mathrm{e}^{tA}y_0$ when $f = 0$.

It is important to recall that $\lim_{t\to\infty} \mathrm{e}^{tA} = 0$ if and only if A is a *stable* matrix: $Re(\lambda) < 0$, for all $\lambda \in \sigma(A)$.

Important examples (cont.)

When $f(t, y) = b \in \mathbb{C}^n$ (=const.), the solution can also be expressed as

$$y(t) = t\psi_1(tA)(b + Ay_0) + y_0$$

where

$$\psi_1(z) = \frac{\mathrm{e}^z - 1}{z} = 1 + \frac{z}{2!} + \frac{z^2}{3!} + \cdots$$

Trigonometric functions and square roots of matrices are also important in applications. For example, the solution to the second-order system

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + Ay = 0, \qquad y(0) = y_0, \qquad y'(0) = y_0'$$

(where A is SPD) can be expressed as

$$y(t) = \cos(\sqrt{A}t)\, y_0 + (\sqrt{A})^{-1} \sin(\sqrt{A}t)\, y_0' \,.$$

Important examples (cont.)

If $A, B \in \mathbb{C}^{n\times n}$ commute ($AB = BA$), then the identity

$$\mathrm{e}^{A+B} = \mathrm{e}^{A}\mathrm{e}^{B}$$

holds true, but in general $\mathrm{e}^{A+B} \neq \mathrm{e}^{A}\mathrm{e}^{B}$.

A useful identity that is always true is

$$\mathrm{e}^{A} = \cosh(A) + \sinh(A)\,,$$

where

$$\cosh(A) = \frac{\mathrm{e}^{A} + \mathrm{e}^{-A}}{2} = \sum_{k=0}^{\infty} \frac{1}{(2k)!} A^{2k}$$

and

$$\sinh(A) = \frac{\mathrm{e}^{A} - \mathrm{e}^{-A}}{2} = \sum_{k=0}^{\infty} \frac{1}{(2k+1)!} A^{2k+1}\,.$$

Important examples (cont.)

Apart from the contour integration formula, the matrix exponential and the resolvent are also related through the *Laplace transform*: there exists an $\omega \in \mathbb{R}$ such that $z \notin \sigma(A)$ for $Re(z) > \omega$ and

$$(zI - A)^{-1} = \int_0^\infty \mathrm{e}^{-zt}\mathrm{e}^{tA}\mathrm{d}t = \int_0^\infty \mathrm{e}^{-t(zI-A)}\mathrm{d}t\,.$$

Recall also that if $|z| > \rho(A)$, the following *Neumann series expansion* of the resolvent is valid:

$$(zI - A)^{-1} = z^{-1}(I + z^{-1}A + z^{-2}A^2 + \cdots) = z^{-1}\sum_{k=0}^{\infty} z^{-k}A^k.$$

A useful variant of this expression is

$$(I - \alpha A)^{-1} = I + \alpha A + \alpha^2 A^2 + \cdots = \sum_{k=0}^{\infty} \alpha^k A^k\,,$$

where $0 < \alpha < \frac{1}{\rho(A)}$.

The exponential and resolvent in network analysis

Let $G = (V, E)$ be a graph, or network, assumed to be unweighted and without self-loops, and let A be the corresponding $n \times n$ adjacency matrix ($n = |V|$):

$$a_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E, \\ 0, & \text{else.} \end{cases}$$

A walk of length k in G is a set of nodes $v_{i_1}, v_{i_2}, \ldots v_{i_k}, v_{i_{k+1}} \in V$ such that for all $1 \leq j \leq k$, there is an edge between v_{i_j} and $v_{i_{j+1}}$.

A closed walk is a walk where $v_{i_1} = v_{i_{k+1}}$.

It can be easily shown that

- $[A^k]_{ii} = \#$ of closed walks of length k based at node v_i,
- $[A^k]_{ij} = \#$ of walks of length k that connect nodes v_i and v_j.

The exponential and resolvent in network analysis (cont.)

Consider now a matrix function given by a convergent power series:

$$f(A) = a_0 I + a_1 A + a_0 A^2 + \cdots = \sum_{k=0}^{\infty} a_k A^k,$$

where the coefficients are assumed to be positive. Note that $a_k \to 0$ as $k \to \infty$ since the series converges.

Then, the entries of $f(A)$ are a weighted sum of the number of walks between vertices in V, where the weights are chosen so as to penalize longer walks.

Choosing for instance $a_k = \alpha^k$, where $0 < \alpha < 1/\rho(A)$, corresponds to using the resolvent $f(A) = (I - \alpha A)^{-1}$.

Choosing instead $a_k = 1/k!$ corresponds to using the matrix exponential $f(A) = \mathrm{e}^A$.

The exponential and resolvent in network analysis (cont.)

Consider now the i-th diagonal entry of $f(A)$:

$$[f(A)]_{ii} = \sum_{k=0}^{\infty} a_k [A^k]_{ii}, \quad 1 \leq i \leq n.$$

The diagonal entries of $f(A)$ provide a measure of how important each node is the network, in terms of how central a role that node plays in controlling the flow of information in the network.

Therefore, the diagonal entries of $f(A)$ can be used to rank the nodes of the graph in order of "importance".

Resolvent-based centrality is known as Katz centrality, whereas exponential-based centrality is known as subgraph centrality.

They have both found widespread use in network analysis, side-by-side with other techniques.

The exponential and resolvent in network analysis (cont.)

See, for instance,

L. Katz, *A new status index derived from sociometric analysis*, Psychometrika, 18 (1953), pp. 39–43.

E. Estrada and J. A. Rodríguez-Velázquez, *Subgraph centrality in complex networks*, Phys. Rev. E, 71 (2005), 056103.

E. Estrada and D. J. Higham, *Network properties revealed through matrix functions*, SIAM Rev., 52 (2010), pp. 671–696.

The exponential and resolvent in network analysis (cont.)

Similarly, the (i, j) entry of $f(A)$ can be regarded as a measure of how well two nodes in G communicate. When $f(A) = \mathrm{e}^A$, we obtain the communicability between nodes i and j:

$$C(i, j) = [\mathrm{e}^A]_{ij} = \sum_{k=0}^{\infty} \frac{[A^k]_{ij}}{k!}.$$

Also of interest in applications to network science is the total communicability of a node $v_i \in V$, defined as

$$\sum_{j=1}^{n} C(i, j) = [\mathrm{e}^A \mathbf{1}]_i = i\text{-th row sum of } \mathrm{e}^A,$$

and the total network communicability:

$$TC(G) = \sum_{i=1}^{n} \sum_{j=1}^{n} C(i, j) = \mathbf{1}^T \mathrm{e}^A \mathbf{1}.$$

The exponential and resolvent in network analysis (cont.)

The total communicability of a node is similar to subgraph centrality, but it is much easier to compute for large graphs, since it requires the evaluation of the product $e^A \mathbf{1}$, whereas subgraph centrality requires computing the diagonal entries of e^A, a much harder task.

Similarly, the total communicability $TC(G) = \mathbf{1}^T e^A \mathbf{1}$ is easy to compute. Note that no entry of e^A is explicitly needed.

E. Estrada and N. Hatano, *Communicability in complex networks*, Phys. Rev. E, 77 (2008), 036111.

M. Benzi and C. Klymko, *Total communicability as a centrality measure*, J. Complex Networks, 1(2), 2013, pp. 124–149.

Matrix functions in quantum physics

The matrix exponential plays an especially important role in quantum mechanics. Consider for instance the time-dependent Schrödinger equation:

$$i\frac{\partial \Psi}{\partial t} = H\Psi, \quad t \in \mathbb{R}, \quad \Psi(0) = \Psi_0,$$

where $\Psi_0 \in L^2$ is a prescribed initial state with $\|\Psi_0\|_2 = 1$. Here $H = H^*$ is the Hamiltonian, or energy operator.

The solution is given esplicitly by $\Psi(t) = \mathrm{e}^{-itH}\Psi_0$, for all $t \in \mathbb{R}$; note that since itH is skew-Hermitian, the propagator $U(t) = \mathrm{e}^{-itH}$ is unitary, which guarantees that the solution has unit norm for all t:

$$\|\Psi(t)\|_2 = \|U(t)\Psi_0\|_2 = \|\Psi_0\|_2 = 1, \quad \forall t \in \mathbb{R}.$$

Matrix functions in quantum physics (cont.)

Also very important in many-body quantum mechanics is the Fermi–Dirac operator, defined as

$$f(H) := (I + \exp(\beta(H - \mu I)))^{-1},$$

where $\beta = (\kappa_B T)^{-1}$ is the inverse temperature, κ_B the Boltzmann constant, and μ is the Fermi level, separating the eigenvalues of H corresponding to the first N_e eigenfunctions from the rest, where N_e is the number of electrons. We will come back to this in Lecture IV.

Finally, in statistical quantum mechanics the state of a system is completely described (statistically) by the density operator:

$$\rho := \frac{\mathrm{e}^{-\beta H}}{Z}, \quad \text{where } Z = \mathrm{Tr}(\mathrm{e}^{-\beta H}).$$

$Z = Z(\beta)$ is known as the partition function of the system.

Outline

Computational tasks

There are essentially two types of computational problems involving matrix functions:

(1) Computing individual entries of $f(A)$

(2) Computing the action of $f(A)$ on a vector: $f(A)\mathbf{v}$

The first problem can be further divided into two cases:

(1a) All entries of $f(A)$ are needed

(1b) Only selected entries are needed (e.g., on or near the diagonal)

Problem (1a) is only feasible for A small (up to $n = \mathcal{O}(10^3)$, say).

In the other two cases A is typically large and sparse. In some applications high accuracy is not always required or warranted.

Computational tasks (cont.)

When A is of moderate size and high accuray is important, there exist specialized algorithms for the most important functions, such as the exponential, the logarithm, the p-th root, etc.

The following resources are highly recommended:

N. J. Higham, *Functions of Matrices. Theory and Computation*, SIAM, Philadelphia, 2008.

C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev., 20 (1978), 801–836.

C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.

Note: Our main focus will be on problems of the type (1b) and (2).

Computational tasks (cont.)

Many computations can be formulated as evaluation of expressions of the form $\mathbf{u}^T f(A)\,\mathbf{v}$ for suitably chosen vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and for suitable functions $f(x)$.

For instance, computing individual entries of $f(A)$ requires computing the bilinear form

$$[f(A)]_{ij} = \mathbf{e}_i^T f(A)\,\mathbf{e}_i.$$

Similarly, computing the i-th row sum of $f(A)$ amounts to evaluating

$$\sum_{j=1}^{n} [f(A)]_{ij} = [f(A)\mathbf{1}]_i = \mathbf{e}_i^T f(A)\mathbf{1}.$$

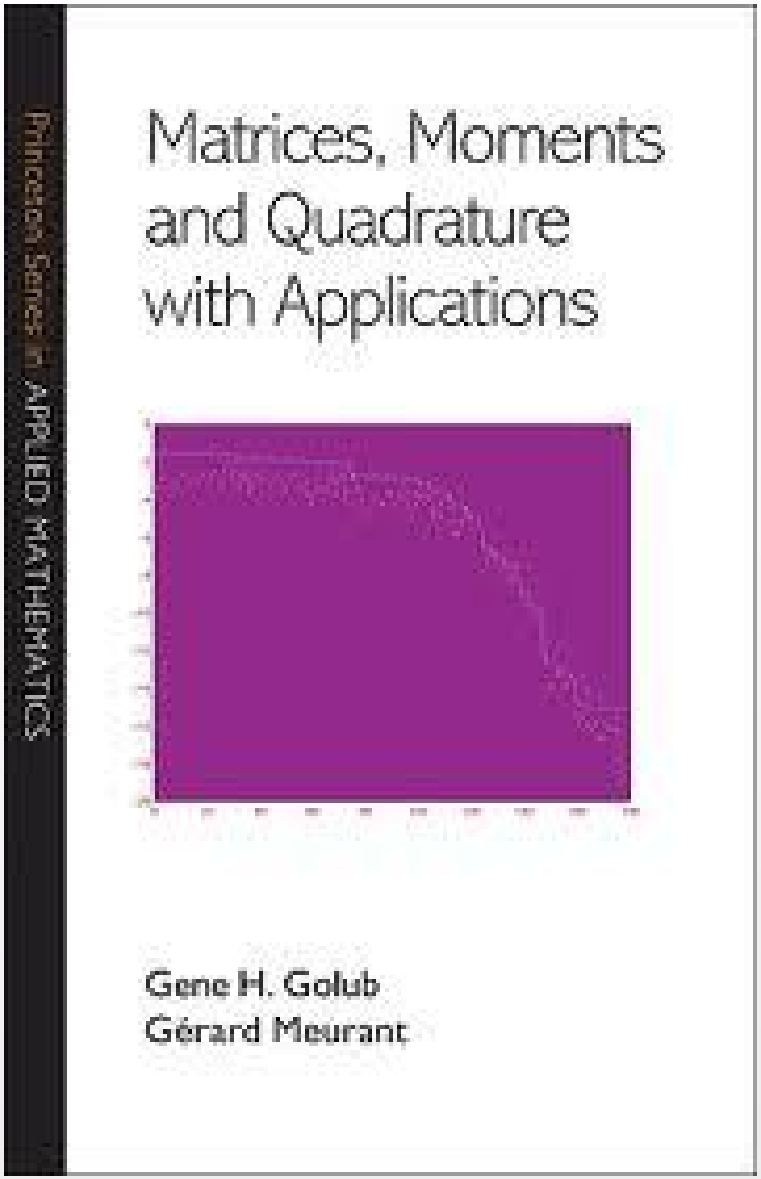
As a further example, the total network communicability requires computing $TC(G) = \mathbf{1}^T \exp(A)\,\mathbf{1}$.

Question: How do we compute these quantities efficiently?

Golub & Meurant to the rescue!

Gene Golub and Gérard Meurant (Oxford, July 2007)

Golub & Meurant to the rescue!



Quadrature-based bounds

It turns out that Gaussian quadrature rules can be used to obtain bounds or estimates for bilinear forms involving functions of Hermitian matrices. These techniques can take full advantage of localization in $f(A)$, when present.

In particular, upper and lower bounds are available for the bilinear form

$$h(\mathbf{u}, \mathbf{v}) = \langle f(A)\mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T f(A)\mathbf{v}$$

with

- $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ (taking $\mathbf{u} = \mathbf{e}_i$, $\mathbf{v} = \mathbf{e}_j$ yields $[f(A)]_{ij}$)
- $A \in \mathbb{R}^{n \times n}$ symmetric
- $f(x)$ strictly completely monotonic on an interval containing the spectrum of A.

Quadrature-based bounds (cont.)

Definition

A real function $f(x)$ is strictly completely monotonic on an interval $I \subset \mathbb{R}$ if $f^{(2j)}(x) > 0$ and $f^{(2j+1)}(x) < 0$ on I for all $j \geq 0$.

Examples:

- $f(x) = 1/x^{\alpha}$ is s.c.m. on $(0, \infty)$, for all $\alpha > 0$,
- $f(x) = \mathrm{e}^{x}$ is not s.c.m.,
- $f(x) = \mathrm{e}^{-x}$ is s.c.m. on $\mathbb{R}$.

In particular, we can compute:

- bounds on $\mathrm{e}^{A} = \mathrm{e}^{-(-A)}$,
- bounds on B^{-1} if $B = I - \alpha A$ is positive definite (that is, if $0 < \alpha < \frac{1}{\lambda_{\max}(A)}$).

Quadrature-based bounds (cont.)

Consider the spectral decompositions

$$A = Q\Lambda Q^T, \qquad f(A) = Qf(\Lambda)Q^T.$$

For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ we have

$$\mathbf{u}^T f(A)\mathbf{v} = \mathbf{u}^T Q f(\Lambda) Q^T \mathbf{v} = \mathbf{p}^T f(\Lambda)\mathbf{q} = \sum_{i=1}^{n} f(\lambda_i) p_i q_i,$$

where $\mathbf{p} = Q^T\mathbf{u}$ and $\mathbf{q} = Q^T\mathbf{v}$. Rewrite this as a Riemann-Stieltjes integral:

$$\mathbf{u}^T f(A)\mathbf{v} = \int_a^b f(\lambda) d\mu(\lambda), \quad \mu(\lambda) = \begin{cases} 0 & \lambda < a = \lambda_1 \\ \sum_{j=1}^{i} p_j q_j & \lambda_i \le \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} p_j q_j & b = \lambda_n \le \lambda. \end{cases}$$

Caveat: Note the numbering of the eigenvalues of A here.

Gauss quadrature

The general Gauss-type quadrature rule is

$$\int_a^b f(\lambda) d\mu(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f],$$

where the nodes $\{z_k\}$ are prescribed.

- Gauss: $M = 0$,
- Gauss–Radau: $M = 1$, $z_1 = a$ or $z_2 = b$,
- Gauss–Lobatto: $M = 2$, $z_1 = a$ and $z_2 = b$.

The evaluation of these quadrature rules is reduced to

- computation of orthogonal polynomials via three-term recurrence,
- or, equivalently, computation of entries and spectral information of the corresponding tridiagonal matrix (Lanczos).

Gauss quadrature (cont.)

For instance, we have for the Gauss rule:

$$\int_a^b f(\lambda)d\mu(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + R[f]$$

with

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \Big[\prod_{j=1}^{N}(\lambda - t_j)\Big]^2 d\mu(\lambda),$$

for some $a < \eta < b$.

Theorem (Golub-Meurant)

$$\sum_{j=1}^{N} w_j f(t_j) = \mathbf{e}_1^T f(J_N)\mathbf{e}_1 = [f(J_N)]_{11}$$

Gauss quadrature (cont.)

The tridiagonal matrix J_N corresponds to the three-term recurrence relationship satisfied by the set of polynomials orthonormal with respect to $d\mu$:

$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \gamma_{N-1} & \omega_N \end{pmatrix}$$

The eigenvalues of J_N are the Gauss nodes, whereas the Gauss weights are given by the squares of the first entries of the normalized eigenvectors of J_N.

The quadrature rule is computed with the Golub–Welsch QR algorithm. Alternatively, the (1,1) entry of $f(J_N)$ can be computed via explicit diagonalization of J_N.

Gauss quadrature (cont.)

Consider the case $\mathbf{u} = \mathbf{v} = \mathbf{e}_i$ (corresp. to the (i, i) entry of $f(A)$).

The entries of J_N are computed using the symmetric Lanczos algorithm:

$$\gamma_j \mathbf{x}_j = \mathbf{r}_j = (A - \omega_j I)\mathbf{x}_{j-1} - \gamma_{j-1}\mathbf{x}_{j-2}, \qquad j = 1, 2, \ldots$$
$$\omega_j = \mathbf{x}_{j-1}^T A \mathbf{x}_{j-1},$$
$$\gamma_j = \|\mathbf{r}_j\|_2$$

with initial vectors $\mathbf{x}_{-1} = \mathbf{0}$ and $\mathbf{x}_0 = \mathbf{e}_i$.

Two approaches for computing bounds:

- Explicit, *a priori* bounds are obtained by taking a single Lanczos step.
- Alternatively, one may explicitly carry out a certain number of Lanczos iterations (MMQ Matlab toolbox by G. Meurant). Each additional Lanczos step amounts to adding another node to the Gauss-type quadrature rule, resulting in tighter and tighter bounds.

Gauss quadrature (cont.)

For $f(A) = \mathrm{e}^A$ and $f(A) = (I - \alpha A)^{-1}$ we obtain:

- bounds on $[f(A)]_{ii}$ from symmetric Lanczos,
- bounds on $[f(A)]_{ij}$ using the identity

$$\mathbf{e}_i^T f(A)\mathbf{e}_j = \frac{1}{4}[(\mathbf{e}_i + \mathbf{e}_j)^T f(A)(\mathbf{e}_i + \mathbf{e}_j) - (\mathbf{e}_i - \mathbf{e}_j)^T f(A)(\mathbf{e}_i - \mathbf{e}_j)],$$

- or, bounds on $[f(A)]_{ii} + [f(A)]_{ij}$ using nonsymmetric Lanczos,
- lower bounds from the Gauss and the Gauss-Radau rules,
- upper bounds from the Gauss-Radau and Gauss-Lobatto rules.

In computations one can use simple Geršgorin estimates instead of the exact values of $\lambda_{\mathsf{min}}(A)$, $\lambda_{\mathsf{max}}(A)$; however, convergence may be slowed down. Using more accurate estimates of the extreme eigenvalues of A generally leads to improved results.

Explicit bounds on the entries of matrix functions

Carrying out explicitly ("by hand") a single Lanczos step, we obtain explicit bounds on the entries of $f(A)$.

These bounds can be fairly tight for matrices that have strong diagonal dominance, and can be used, for instance, to compute simple preconditioners in the case of $f(A) = A^{-1}$ or $f(A) = A^{-1/2}$.

They can also be used to give lower and upper bounds on the entries of functions of adjacency matrices of graphs.

See

M. Benzi and G. H. Golub, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417–418.

M. Benzi & P. Boito, *Quadrature rule-based bounds for functions of adjacency matrices*, Linear Algebra Appl., 433: 637–652 (2010).

MMQ (iteratively computed) bounds

- This refers to the (increasingly tight) bounds obtained by adding quadrature nodes (= performing additional Lanczos steps) to estimate entries of $f(A)$;
- Computations are performed using G. Meurant's MMQ package, suitably adapted to handle sparse matrices;
- Fast and accurate computation of upper and lower bounds for entries of $f(A)$;
- Complexity is at most $\mathcal{O}(n)$ per iteration (much less for the first few iterations) ;
- For certain matrices, the number of iterations is (nearly) independent of n (see below);
- Parallelization is possible on shared memory machines.

Low-rank approximations

A drawback of these techniques in the case of large matrices is that the cost remains high.

The approximate $\mathcal{O}(n^2)$ scaling to evaluate the diagonal entries of $f(A)$ is unacceptably high for matrices of large size (n in the millions or more).

For some matrix functions, the cost can be reduced by working with low-rank approximations of A. For example, estimating a small number $k \ll n$ of the largest eigenpairs $(\lambda_i, \mathbf{x}_i)$ and using the approximation

$$\mathrm{e}^A \approx \sum_{i=1}^{k} \mathrm{e}^{\lambda_i} \mathbf{x}_i \mathbf{x}_i^T$$

can lead to significant savings. The accuracy is typically good in network analysis problems, due to rapid decay in the eigenvalues of A for many networks.

Low-rank approximations (cont.)

For details, we refer to the following works:

C. Fenu, D. Martin, L. Reichel, & G. Rodriguez, *Network analysis via partial spectral factorization and Gauss quadrature*, SIAM J. Sci. Comput., 35 (2013), pp. A2046–2068.

C. Fenu, D. Martin, L. Reichel, & G. Rodriguez, *Block Gauss and anti-Gauss quadrature with application to networks*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1655–1684.

J. Baglama, C. Fenu, L. Reichel, & G. Rodriguez, *Analysis of directed networks via partial singular value decomposition and Gauss quadrature*, Linear Algebra Appl., 456 (2014), pp. 93–121.

Computing row and column sums

Many important problems in computational science require computing the action of a matrix function on a given vector: $f(A)\mathbf{v}$.

The problem arises, for instance, in the context of exponential integrators for systems of ODEs. It also arises in computing the total communicability vector for a graph G with adjacency matrix A:

$$\mathbf{TC}(G) = \mathrm{e}^{A}\mathbf{1},$$

where $\mathbf{1}$ is the vector of all ones.

Of course, approximating the solution $\mathbf{x} = A^{-1}\mathbf{b}$ of a large linear system is also a problem of this type.

Such computations can be done very fast using Krylov subspace methods for evaluating the action of the matrix function on a vector.

A very efficient Matlab tolbox has been developed by Stefan Güttel:

`http://www.guettel.com`

Krylov subspace methods

Krylov subspace methods are the algorithms of choice for solving many linear algebra problems, including:

- Large-scale linear systems of equations $A\mathbf{x} = \mathbf{b}$
- Large-scale (generalized) eigenvalue problems $A\mathbf{x} = \lambda B\,\mathbf{x}$
- Computing $f(A)\mathbf{v}$ where A is a large matrix, $\mathbf{v}$ is a vector and f a given function (e.g., $f(A) = \mathrm{e}^{-tA}$, $t > 0$)
- Solving matrix equations (e.g., $AX + XA^* + B = 0$)

An attractive feature of Krylov methods in some applications is that the matrix A is not needed explicitly: only the ability to multiply A times a given vector is required (*action*, or *operator principle*).

Krylov subspace methods (cont.)

The main idea behind Krylov methods is the following:

- A nested sequence of suitable low-dimensional subspaces (the Krylov subspaces) is generated;
- The original problem is projected onto these subspaces;
- The (small) projected problems are solved “exactly";
- The approximate solution is expanded back to the original N-dimensional space, once sufficient accuracy has been attained.

Krylov subspace methods are example of polynomial approximation methods, where $f(A)\mathbf{v}$ is approximated by $p(A)\mathbf{v}$ where p is a (low-degree) polynomial. Since every matrix function $f(A)$ is a polynomial in A, this is appropriate.

Krylov subspace methods (cont.)

The kth Krylov subspace of $A \in \mathbb{C}^{n\times n}$ and a nonzero vector $\mathbf{v} \in \mathbb{C}^n$ is defined by

$$\mathcal{K}_k(A, \mathbf{v}) = \text{span}\,\{\mathbf{v}, A\mathbf{v}, \ldots, A^{k-1}\mathbf{v}\},$$

and it can be written as

$$\mathcal{K}_k(A, \mathbf{v}) = \{q(A)\mathbf{v} \,|\, q \text{ is a polynomial of degree } \leq k-1\}.$$

Obviously,

$$\mathcal{K}_1(A, \mathbf{v}) \subset \mathcal{K}_2(A, \mathbf{v}) \subset \cdots \subset \mathcal{K}_d(A, \mathbf{v}) = \cdots = \mathcal{K}_n(A, \mathbf{v}).$$

Here d is the degree of the minimum polynomial of A with respect to $\mathbf{v}$.

Krylov subspace methods (cont.)

As is well known, computing projections onto a subspace is greatly facilitated if an orthonormal basis for the subspace is known. This is also desirable for numerical stability reasons.

An orthonormal basis for a Krylov subspace can be efficiently constructed using the Arnoldi process; in the Hermitian case, this is known as the Lanczos process (Arnoldi, 1951; Lanczos, 1952). Both of these are efficient implementations of the classical Gram–Schmidt process.

In Arnoldi's method, the projected matrix has upper Hessenberg structure, which can be exploited in the computation. In the Hermitian case it is tridiagonal.

Arnoldi's process

For arbitrary $A \in \mathbb{C}^{n\times n}$ and $\mathbf{v} \in \mathbb{C}^n$, $\mathbf{v} \neq \mathbf{0}$, the Arnoldi process is:

- Set $\mathbf{q}_1 = \mathbf{v}/\|\mathbf{v}\|_2$;
- For $j = 1, \ldots, m$ do:
  - $h_{i,j} = \langle A\mathbf{q}_i, \mathbf{q}_j \rangle$ for $i = 1, 2, \ldots, j$
  - $\mathbf{u}_j = A\mathbf{q}_j - \sum_{i=1}^{j} h_{i,j}\mathbf{q}_i$
  - $h_{j+1,j} = \|\mathbf{u}_j\|_2$
  - If $h_{j+1,j} = 0$ then STOP;
  - $\mathbf{q}_{j+1} = \mathbf{u}_j/\|\mathbf{u}_j\|_2$

Remarks:

(i) If the algorithm does not stop before the mth step, the Arnoldi vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ form an ONB for the Krylov subspace $\mathcal{K}_m(A, \mathbf{v})$.

(ii) At each step j, the Arnoldi process requires one matrix-vector product, $j+1$ inner products, and j linked triads.

(iii) At each step the algorithm computes $A\mathbf{q}_j$ and then orthonormalizes it against all previously computed $\mathbf{q}_j$'s.

Arnoldi's process (cont.)

Define $Q_m = [\mathbf{q}_1, \ldots, \mathbf{q}_m] \in \mathbb{C}^{n \times m}$. Introducing the $(m+1) \times m$ matrix $\widehat{H}_m = [h_{ij}]$ and the $m \times m$ upper Hessenberg matrix H_m obtained by deleting the last row of $\widehat{H}_m$, the following Arnoldi relations hold:

$$AQ_m = Q_m H_m + \mathbf{u}_m \mathbf{e}_m^T = Q_{m+1} \widehat{H}_m$$

$$Q_m^* A Q_m = H_m$$

Hence, the $m \times m$ matrix H_m is precisely the projected matrix $Q_m^* A Q_m$.

If $A = A^*$, then $H_m = H_m^* = T_m$ is a tridiagonal matrix, and the Arnoldi process reduces to the Lanczos process, which is much cheaper in terms of both operations and storage.

Indeed, the Lanczos process consists of a three-term recurrence, with constant operation count and storage costs per step, whereas the Arnoldi process has increasing costs for increasing j.

Krylov subspace approximation to $f(A)\mathbf{v}$

To approximate $f(A)\mathbf{v}$ we take $\mathbf{q}_1 = \mathbf{v}/\|\mathbf{v}\|_2$ and for an appropriate k we compute

$$\mathbf{f}_k := \|\mathbf{v}\|_2 Q_k f(H_k)\mathbf{e}_1 = Q_k f(H_k) Q_k^* \mathbf{v}.$$

The vector $\mathbf{f}_k$ is the kth approximation to $f(A)\mathbf{v}$. Typically, $k \ll n$ and computing $f(H_k)$ is inexpensive, and can be carried out in a number of ways. For instance, when $H_k = H_k^* = T_k$, it can be computed via explicit diagonalization of T_k.

Deciding when this approximation $\mathbf{f}_k$ is sufficiently close to $f(A)\mathbf{v}$ to stop the process is a non-trivial problem, and an area of active research.

For the case of the matrix exponential, Saad (1992) suggested the error estimate

$$\|\mathrm{e}^A\mathbf{v} - Q_k \mathrm{e}^{H_k} Q_k^* \mathbf{v}\|_2 \approx \|\mathbf{v}\|_2 \mathbf{e}_k^T \mathrm{e}^{H_k} \mathbf{e}_1 \,.$$

Note that $\mathbf{e}_k^T \mathrm{e}^{H_k} \mathbf{e}_1$ is the $(k,1)$ entry of e^{H_k}.

A convergence result for functions of adjacency matrices

Theorem

Let $\{A_n\}$ be the adjacency matrices associated with a sequence of graphs $\{G_n\}$ of size $n \to \infty$. Assume that the degree d_i of any node in G_n satisfies $d_i \leq D$ for all n, with D constant. Let f be an analytic function defined on a region containing the interval $[-D, D]$. Then, for any given $\varepsilon > 0$, the number of Lanczos steps (or, equivalently, of quadrature nodes) needed to approximate any entry $[f(A)]_{ij}$ with an error $< \varepsilon$ is bounded independently of n.

The assumption of bounded degree may be restrictive for certain graphs. More generally, it can be shown that the number of Lanczos steps grows at worse like $\mathcal{O}(D_n)$, where D_n is the max degree of any node in G_n.

The result is a consequence of the fact that the coefficients of an analytic function in the Chebyshev expansion decay super-exponentially to 0.

Some results for real world networks

Network	n	NNZ	λ_1	λ_2
Zachary Karate Club	34	156	6.726	4.977
Drug Users	616	4024	18.010	14.234
Yeast PPI	2224	13218	19.486	16.134
Pajek/Erdos971	472	2628	16.710	10.199
Pajek/Erdos972	5488	14170	14.448	11.886
Pajek/Erdos982	5822	14750	14.819	12.005
Pajek/Erdos992	6100	15030	15.131	12.092
SNAP/ca-GrQc	5242	28980	45.617	38.122
SNAP/ca-HepTh	9877	51971	31.035	23.004
SNAP/as-735	7716	26467	46.893	27.823
Gleich/Minnesota	2642	6606	3.2324	3.2319

Characteristics of selected real world networks. All networks are undirected.

Some results for real world networks (cont.)

Network	`expm`	`mmq`	`funm_kryl`
Zachary Karate Club	0.062	0.138	0.120
Drug Users	0.746	2.416	0.363
Yeast PPI	47.794	9.341	0.402
Pajek/Erdos971	0.542	2.447	0.317
Pajek/Erdos972	579.214	35.674	0.410
Pajek/Erdos982	612.920	39.242	0.393
Pajek/Erdos992	656.270	53.019	0.325
SNAP/ca-GrQc	281.814	23.603	0.465
SNAP/ca-HepTh	2710.802	58.377	0.435
SNAP/as-735	2041.439	75.619	0.498
Gleich/Minnesota	1.956	10.955	0.329

Timings (in seconds) to compute all the diagonal entries and row sums of e^A for various test problems using different codes. `expm`: Matlab's built-in matrix exponential; `mmq`: Meurant's code, modified; `funm_kryl`: Güttel's code.

A large example: the Wikipedia graph

The Wikipedia graph is a directed graph with $n = 4,189,503$ nodes and $67,197,636$ edges (as of June 6, 2011).

The row/columns of the exponential of the adjacency matrix can be used to rank the “hubs" and “authorities" in Wikipedia in order of importance.

Using the Arnoldi-based code `funm_kryl` we can compute the hub and authority rankings for all the nodes in about 216s to high accuracy on a parallel system comprising 24 Intel(R) Xeon(R) E5-2630 2.30GHz CPUs.

Outline

Concluding remarks

- Problems involving matrix functions arise in many areas of computational science and engineering
- Algorithms based on the Lanczos and Arnoldi processes enable the efficient solution of various computational problems involving functions of large matrices
- The performance of these methods depends strongly on structural properties of these matrices, such as their eigenvalue distribution, graph structure, and the presence of localization/delocalization in $f(A)$

Outline

Monographs

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

- J. Liesen and Z. Strakos, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, 2013.

Software

- Complex Networks Package for Matlab:

  `http://www.levmuchnik.net/Content/Networks/ComplexNetworksPackage.html`

- CONTEST Matlab Toolbox:

  `http://www.mathstat.strath.ac.uk/research/groups/numerical_analysis/contest/toolbox`

- G. Meurant's MMQ package: `http://pagesperso-orange.fr/gerard.meurant/`.

- Stefan Güttel package `funm_kryl`: `http://www.guettel.com`