Results in Applied Mathematics 21 (2024) 100434

Contents lists available at ScienceDirect

Results in Applied Mathematics

journal homepage: www.elsevier.com/locate/results-in-applied-mathematics

An experimental comparison of methods for computing the numerical radius

Tim Mitchell [a], Michael L. Overton [b,*,1]

[a] Department of Computer Science, Queens College/CUNY, 65-30 Kissena Boulevard, Flushing, 11367, NY, USA
[b] Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, 10012, NY, USA

ARTICLE INFO

MSC:
15A60
90C22

Keywords:
Semidefinite programming
Convex optimization
Nonconvex optimization
Field of values
Numerical range

ABSTRACT

We make an experimental comparison of methods for computing the numerical radius of an $n \times n$ complex matrix, based on two well-known characterizations, the first a nonconvex optimization problem in one real variable and the second a convex optimization problem in $n^2 + 1$ real variables. We make comparisons with respect to both accuracy and computation time using publicly available software.

1. Introduction

Let $A \in \mathbf{M}_n$, the space of $n \times n$ complex matrices. The numerical radius $r(A)$ is defined as the maximum, in modulus, of the points in the field of values (numerical range), i.e.,

$$r(A) = \max\left\{ |v^* A v| \,:\, v \in \mathbb{C}^n, \|v\|_2 = 1 \right\}.$$

A well-known result (Johnson [1]; see also Kippenhahn [2]) is

$$r(A) = \max_{\theta \in [0,2\pi)} h(\theta) \tag{1}$$

where

$$h(\theta) = \lambda_{\max}(H(\theta))$$

and

$$H(\theta) = \frac{1}{2}\left(e^{\mathbf{i}\theta} A + e^{-\mathbf{i}\theta} A^* \right),$$

with $\lambda_{\max}(\cdot)$ denoting largest eigenvalue of a Hermitian matrix, $\mathbf{i}$ the imaginary unit and * denoting complex conjugate transpose. Although the optimization problem (1) has only one real variable, it is nontrivial to solve because h is nonconvex. Various algorithms have been introduced to solve (1), including the level-set method of Mengi and Overton [3] (based on earlier work of He and Watson [4] and of Boyd and Balakrishnan [5]), and a cutting plane method of Uhlig [6] (based on earlier work of Johnson [1]).

* Corresponding author.
E-mail addresses: tmitchell@qc.cuny.edu (T. Mitchell), mo1@nyu.edu (M.L. Overton).
1 The work of this author was partially supported by the National Science Foundation, United States under grant DMS-2012250.

https://doi.org/10.1016/j.rinam.2024.100434
Received 25 October 2023; Received in revised form 3 January 2024; Accepted 8 January 2024
Available online 30 January 2024

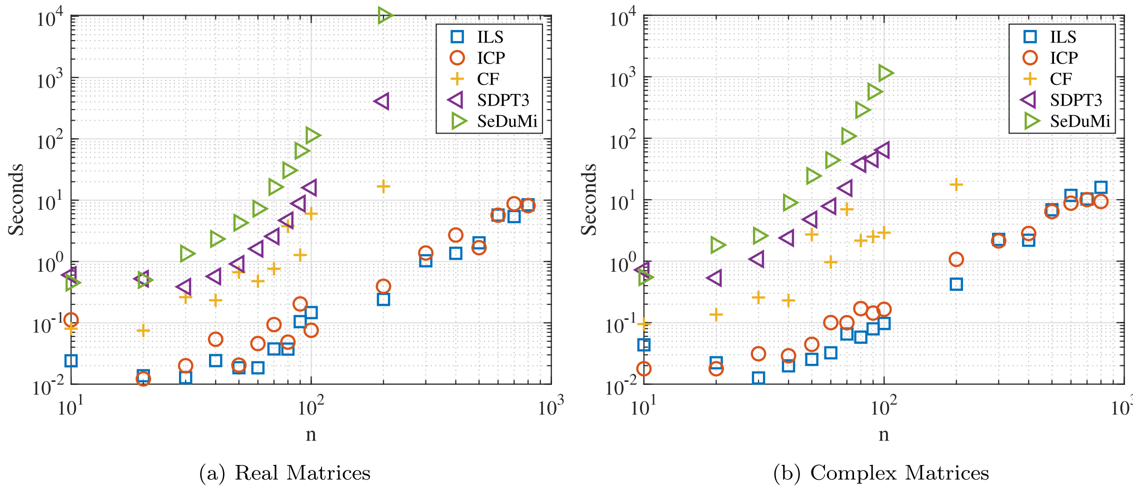

Fig. 1. The elapsed running times of the five methods tested are shown. For problems that took at most 3 min to solve, the reported times are the average of five trials in order to account for timing variability, while all other reported times are from a single run. Due to their high cost, CF, SDPT3, and SeDuMi were only tested up to size $n = 200$, and on the complex matrix with $n = 200$, both SDPT3 and SeDuMi crashed, and thus their times are only reported up to $n = 100$ in the right plot.

More recently, improvements to both of these approaches have been given by Mitchell [7, Algorithms 3.1 and 5.1], along with a state-of-the-art hybrid method [7, section 6] that combines the virtues of both in order to remain efficient in all cases.

A second well-known characterization of $r(A)$ (Mathias [8]; see also Ando [9]) is

$$r(A) = \min_{c\in\mathbb{R}, Z\in\mathbf{M}_n, Z=Z^*} \left\{ c : \begin{bmatrix} cI_n + Z & A \\ A^* & cI_n - Z \end{bmatrix} \geq 0 \right\}, \tag{2}$$

where $M \geq 0$ means a Hermitian matrix M is positive semidefinite. Thus, $r(A)$ can be computed by solving a convex optimization problem, known as a semidefinite program (SDP), in $n^2 + 1$ real variables. If A is real, then Z can be restricted to be real symmetric, so the optimization problem has $n(n+1)/2 + 1$ variables. It is well known that such convex optimization problems can be solved, up to any given accuracy, in polynomial time using the Turing complexity model; the traditional method guaranteeing such complexity is the ellipsoid method, but more practical interior-point methods have also recently been designed with polynomial-time complexity [10].

However, for all practical purposes, interior-point methods for computing $r(A)$ via (2) are very slow compared to more direct methods based on (1). In this note we illustrate this with some experimental computations carried out using Mitchell's codes available from the supplemental materials published in [7] as well as the latest stable release (version 2.2, build 1148) of the well-known CVX software for "disciplined convex programming" [11]. CVX uses a variety of primal–dual interior-point "solvers" for SDP; we use two of these, SDPT3 and SeDuMi. CVX lacks the polynomial-time guarantees of the methods described by [10], but it is much faster than such methods, partly because it uses floating point arithmetic, while any rigorous polynomial-time method would necessarily have to be implemented using arbitrary precision integer or rational arithmetic, as discussed at the end of [10].

2. The experiments and the results

We experimented with the following methods using MATLAB R2023a Update 5 and a 2023 Mac Mini configured with a 12-core M2 Pro Apple Silicon CPU and 16 GB RAM running macOS 13.6:

- ILS: solve (1) via the Improved Level-Set Algorithm 3.1 in [7]
- ICP: solve (1) via the Improved Cutting-Plane Algorithm 5.1 in [7][2]
- CF: solve (1) using Chebfun via adaptive Chebyshev interpolation[3]
- SDPT3: solve (2) using CVX [11] with the SDPT3 solver

[2] We forgo including Mitchell's hybrid method that uses both Algorithms 3.1 and 5.1 because on the particular problems in our benchmark here, Algorithm 3.1 and Algorithm 5.1 perform quite similarly. However, note that this is not always the case. For an extensive benchmark using a large variety of matrices to illustrate how the relative efficiencies of the level-set, cutting-plane, and hybrid approaches can differ substantially, depending on the specific geometry of the field of values of the matrix, see [7, pp. A773–A778].

[3] Chebfun is a package for globally approximating functions to machine precision accuracy using adaptive Chebyshev interpolation techniques; for more details, see [12]. By method CF, we mean using Chebfun to accurately approximate h on $[0, 2\pi]$ followed by a call to the Chebfun `max` routine to find the maximal value, as proposed in [6].

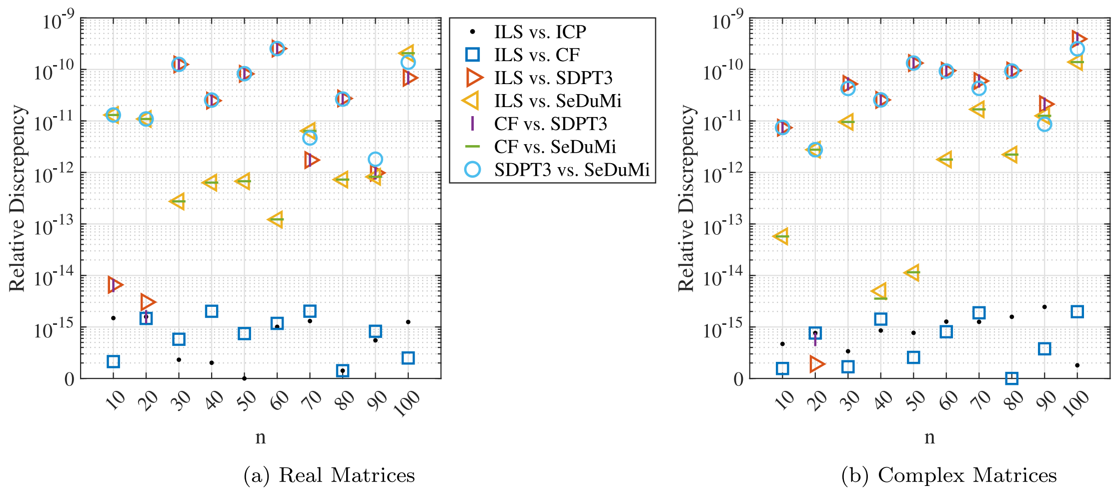

Fig. 2. The plots give the pairwise relative discrepancies of the methods' computed numerical radius estimates, i.e., $|(r_1(A) - r_2(A))| / \max(r_1(A), r_2(A))$, where $r_1(A)$ and $r_2(A)$ are the computed results for the two methods being compared on matrix A. Since the values returned by the ILS and ICP methods agree to about 15 digits, we omit other error comparisons involving the ICP method.

- SeDuMi: solve (2) using CVX [11] with the SeDuMi solver

We conducted our experiments using randomly generated real as well as complex matrices ranging in size from $n = 10$ to $n = 800$; the matrix coefficients (real parts and imaginary parts, as appropriate) were generated using the standard normal distribution.[4] Fig. 1 shows the running times for our test matrices, with the left panel showing the results for real matrices and the right panel the results for complex matrices. We see immediately that ILS and ICP perform similarly and they are far more efficient than the other three methods, and that among those other three, CF is the fastest, then SDPT3, and then SeDuMi.[5] Meanwhile, Fig. 2 shows relative discrepancies between the computed values of the numerical radius r. We see that methods ILS, ICP, and CF have relative discrepancies always near 10^{-15}, indicating that they compute r to about 14 correct digits, quite impressive considering that MATLAB's IEEE double format numbers have only 53 bits of precision, or approximately 16 decimal digits. The other discrepancies show that the interior point methods solving (2) return results which are accurate to only about 9 to 12 digits, although we note that CVX was instructed to compute the results with high precision accuracy.[6]

3. Conclusion

Our experiments show that although the characterization of the numerical radius as the solution of a semidefinite program is a valuable theoretical tool, as a practical matter, Mitchell's improved level-set and cutting-plane algorithms based on the nonconvex characterization are much faster. We note, however, that the SDP formulation has the appealing property that it also allows the convenient solution of linearly parameterized optimization problems involving the numerical radius, such as the one given in [13, Sec. 2.1].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

[4] Code for replicating all of our experiments here has been permanently archived on Zenodo and can be downloaded at https://doi.org/10.5281/zenodo.10449890.

[5] We note that CVX uses precompiled binary files for both SDPT3 and SeDuMi; if these were instead implemented in MATLAB, they would presumably be even slower.

[6] By setting `cvx_precision high` in the calling code.

References

[1] Johnson Charles R. Numerical determination of the field of values of a general complex matrix. SIAM J Numer Anal 1978;15(3):595–602.
[2] Kippenhahn Rudolf. Über den Wertevorrat einer matrix. Math Nachr 1951;6:193–228.
[3] Mengi Emre, Overton Michael L. Algorithms for the computation of the pseudospectral radius and the numerical radius of a matrix. IMA J Numer Anal 2005;25(4):648–69.
[4] He C, Watson GA. An algorithm for computing the numerical radius. IMA J Numer Anal 1997;17(3):329–42.
[5] Boyd S, Balakrishnan V. A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞-norm. Systems Control Lett 1990;15(1):1–7.
[6] Uhlig Frank. Geometric computation of the numerical radius of a matrix. Numer Algorithms 2009;52(3):335–53.
[7] Mitchell Tim. Convergence rate analysis and improved iterations for numerical radius computation. SIAM J Sci Comput 2023;45(2):A753–80.
[8] Mathias Roy. Matrix completions, norms and Hadamard products. Proc Amer Math Soc 1993;117(4):905–18.
[9] Ando T. Structure of operators with numerical radius one. Acta Sci Math (Szeged) 1973;34:11–5.
[10] de Klerk Etienne, Vallentin Frank. On the Turing model complexity of interior point methods for semidefinite programming. SIAM J Optim 2016;26(3):1944–61.
[11] Grant Michael, Boyd Stephen. CVX: Matlab software for disciplined convex programming, version 2.1. 2014, http://cvxr.com/cvx.
[12] Driscoll TA, Hale N, Trefethen LN. Chebfun guide. Oxford, UK: Pafnuty Publications; 2014.
[13] Lewis AS, Overton ML. Partial smoothness of the numerical radius at matrices whose fields of values are disks. SIAM J Matrix Anal Appl 2020;41(3):1004–32.