ROSTAPACK: RObust STAbality PACKage Version 2.2

Tim Mitchell

September 6, 2019

Abstract

ROSTAPACK: RObust STAbality PACKage is an AGPL-licensed open-source library implemented in MATLAB containing routines for computing or approximating robust stability measures of dynamical systems. In the latter case, the approximation methods are mostly intended for large-scale systems, where it would otherwise be prohibitively expensive to compute a particular stability measure. The purpose of this document is to give a brief overview of the user-facing routines and provide citation information for the underlying algorithms and techniques. Full documentation of each routine is provided in its respective MATLAB `help` documentation.

Pronunciation

ROSTAPACK is pronounced "rost-a-pack", with rost rhyming with cost, frost, lost.

Citing

If you publish work that either refers to or makes use of ROSTAPACK, please cite:

- the software itself with version number and author, e.g.
  ROSTAPACK: RObust STAbility PACKage v2.2, Tim Mitchell
- the relevant paper(s) for the underlying algorithm(s) of your interest.

For your convenience, detailed citation information for each of the main computational routines is provided in this document.

1 Linear time-invariant systems

ROSTAPACK provides routines applicable to both continuous- and discrete-time linear dynamical systems, respectively in the form of

$$E\dot{x} = Ax + Bu, \tag{1a}$$
$$y = Cx + Du \tag{1b}$$

and

$$Ex_{k+1} = Ax_k + Bu_k \tag{2a}$$
$$y_k = Cx_k + Du_k, \tag{2b}$$

where $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{p\times n}$, $D \in \mathbb{C}^{p\times m}$, and $E \in \mathbb{C}^{n\times n}$ is assumed to be invertible.

2 Main methods

Routines located in: `ROSTAPACK_DIR/main/`

The following *exact* methods compute their corresponding robust stability measures. In order to ensure the global convergence facilitating exact computation, computing all eigenvalues of matrix pencils of order $2n$ must be tractable, specifically, $\mathcal{O}(n^3)$ work and $\mathcal{O}(n^2)$ memory, with significant constants, per iteration. However, these exact routines generally converge within just a handful of iterations.

Note that the ROSTAPACK currently uses `eig` for all such eigenvalue problems. However, to mitigate possible numerical issues when attempting to find *purely imaginary* eigenvalues, ideally a structured-preserving eigensolver should be used instead (e.g. SLICOT). Support for SLICOT eigensolvers is coming in a future release.

2.1 Routines

`specValSet` Computes the *ε-spectral value set abscissa or radius*. In the special case that $B = C = E = I$ and $D = 0$, the *ε-pseudospectral abscissa or radius* is computed. This routine implements the newer and extended criss-cross methods of [BM17], which are often significantly faster and more accurate than the earlier criss-cross routines.

2.2 Parameter processing

Without input arguments, these routines return an `options` struct containing the complete set of default values for their relevant parameters. These routines will also check user-supplied parameters, replacing the corresponding default values in `options` or throwing errors for invalid choices.

`specValSetOptions` User-tunable parameters for `specValSet`

3 Approximation methods (sparse eigenvalue based)

Routines located in: `ROSTAPACK_DIR/large-scale/`

The methods, under mild assumptions, return locally optimal (often global) approximations to their corresponding robust stability measures. When such a computed approximate value happens to be globally optimal, these methods return the exact value. The basic unit of computation is a sparse eigenvalue computation, via `eigs`, to obtain either the rightmost or outermost eigenvalue of an evolving $n \times n$ matrix, which is a fixed sparse term plus a changing low-rank modification. In order for `eigs` to be efficient, it is assumed that all system matrices of (1) or (2) permit fast matrix-vector products, particularly A. Note that while dense eigenvalue computations can optionally be enabled (e.g. for testing purpose), the routines in this section still remain approximation methods.

In theory, the algorithms implemented by the following ROSTAPACK routines assume that a globally rightmost/outermost eigenvalue is among the eigenvalues returned by `eigs` but in practice, sporadic failure of `eigs` in this regard may be able to be tolerated. However, if `eigs` does not return any eigenvalues for a given subproblem, then the algorithms have no choice but to halt. For the following ROSTAPACK approximation methods, it is typically the initial eigenvalue subproblem, computing the k rightmost/outermost eigenvalues of A, that may cause difficulty for `eigs`. In other words, if `eigs` is successful on this first eigenvalue subproblem, typically all subsequent calls to `eigs` will succeed. If `eigs` fails on the initial subproblem, one can (a) adjust

its parameters and/or (b) optionally choose to start at a perturbation of A and skip computing eigenvalues of A. For more info, see `eigensolverOptions`, `initialPerturbationOptions`, and `opts.check_stability` from `getStabRadBoundOptions`.

3.1 Routines

`getStabRadBound` Implements the quadratically-convergent Hybrid Expansion-Contraction (HEC) algorithm of [MO16]. HEC returns a value $\hat{\varepsilon}$ that, under mild assumptions, is guaranteed to be a locally optimal upper bound to the *complex stability radius*. As the reciprocal of the complex stability radius is the H_∞ norm, it follows that $\hat{\varepsilon}^{-1}$ is a guaranteed locally optimal lower bound to the H_∞ norm. This routine can also be configured to compute a locally optimal upper bound to the *real stability radius*, using an extended version of the HEC algorithm [GGMO17]. In this case, the uncertainty in (1) or (2) is now restricted to be real valued and measured with respect to the Frobenius norm.

An initial destabilizing perturbation, a necessary initialization condition for HEC, is computed using Algorithm Fast Upper Bound of [MO16, Section 4.4].

`specValSetBound` Returns a lower bound to the *ε-spectral value set abscissa or radius*, which, under mild assumptions, is locally optimal. Optionally, this routine can instead compute a lower bound to the *real Frobenius-norm bounded spectral value set abscissa or radius*, provided $\min(m, p) > 1$. The underlying algorithm was first introduced for pseudospectra in [GO11], extended to general complex-valued spectral value sets in [GGO13], and then to the real Frobenius-norm case in [GGMO17]. As convergence of the basic algorithm is generally linear, by default this routine uses extrapolation and interpolation techniques to try to accelerate the computation. The rank-1 extrapolation technique for complex-valued spectral value sets (and thus also pseudospectra) is described in [Mit14, Chapter 4.2] and [MO16, Section 6]. The rank-2 extrapolation technique for the real Frobenius-norm case is described in [Mit14, Chapter 6.3.5] and [GGMO17, Section 7.1.2]. The interpolation technique, applicable to pseudospectra and complex- or real-valued spectral sets, is described in [GGMO17, Section 7.1.3]. Improvements to the line search are given in [Mit14, Chapter 4.3] while techniques to efficiently handle $D \neq 0$ when m, p are also large are described in [Mit14, Chapter 4.1] and [MO16, Section 6].

`convertDSS` Both `getStabRadBound` and `specValSetBound` only take (A, B, C, D) systems, where matrix $E = I$. However, nonsingular E matrices can also be supported by first applying E^{-1} to the left side of (1a) or (2a). To do this efficiently for large-scale problems, this utility first precomputes a single LU decomposition of E and then uses it to do the two necessary backsolves for every matrix-vector product requested with $E^{-1}A$, $E^{-1}B$, A^*E^{-*} and B^*E^{-*}. The outputs of this routine can be directly passed to `getStabRadBound` and `specValSetBound` as their `A` and `B` input arguments. In order for the cost of the matrix-vector products to remain low, the LU decomposition of E will need to be quite sparse.

3.2 Parameter processing

Without input arguments, these routines return an `options` struct containing the complete set of default values for their relevant parameters. These routines will also check user-supplied parameters, replacing the corresponding default values in `options` or throwing errors for invalid choices.

3.2.1 Basic

`getStabRadBoundOptions`	High-level user-tunable parameters for `getStabRadBound`
`specValSetBoundOptions`	High-level user-tunable parameters for `specValSetBound`
`initialPerturbationOptions`	Set an initial perturbation and eigenvalue

3.2.2 Advanced (for subroutines of `getStabRadBound` and `specValSetBound`)

`eigensolverOptions`	Parameters for all eigenvalue computations
`epsilonContractOptions`	Relevant to `getStabRadBound` only
`phiSystemsSolverOptions`	Relevant only for $D \neq 0$ with complex-valued perturbations
`upperBoundOptions`	Relevant to `getStabRadBound` only
`uvExpandOptions`	Relevant to `getStabRadBound` and `specValSetBound`

3.3 Utilities

`assertSystem`	Checks input arguments specifying (A, B, C, D) are correct
`frobeniusNormFast`	Returns the Frobenius norm of a low-rank matrix UV^*
`normalizeTwoNormUV`	Scales U and V such that $\|U\|_2 = \|V\|_2 = 1$
`normalizeFroNormUV`	Scales U and V such that $\|UV^*\|_{\mathrm{F}} = 1$

References

[BM17] P. Benner and T. Mitchell. Extended and improved criss-cross algorithms for computing the spectral value set abscissa and radius. e-print arXiv:1712.10067, arXiv, December 2017. math.OC.

[GGMO17] N. Guglielmi, M. Gürbüzbalaban, T. Mitchell, and M. L. Overton. Approximating the real structured stability radius with Frobenius-norm bounded perturbations. *SIAM J. Matrix Anal. Appl.*, 38(4):1323–1353, 2017.

[GGO13] N. Guglielmi, M. Gürbüzbalaban, and M. L. Overton. Fast approximation of the H_∞ norm via optimization over spectral value sets. *SIAM J. Matrix Anal. Appl.*, 34(2):709–737, 2013.

[GO11] N. Guglielmi and M. L. Overton. Fast algorithms for the approximation of the pseudospectral abscissa and pseudospectral radius of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(4):1166–1192, 2011.

[Mit14] T. Mitchell. *Robust and efficient methods for approximation and optimization of stability measures.* PhD thesis, New York University, New York, NY 10003, USA, September 2014.

[MO16] T. Mitchell and M. L. Overton. Hybrid expansion-contraction: a robust scaleable method for approximating the H_∞ norm. *IMA J. Numer. Anal.*, 36(3):985–1014, 2016.