APMA920 Project
Developing a method of computing the boundary of the polynomial numerical hull

Colin Macdonald

2002/12/18

Introduction

The behavior of matrices is often analyzed and understood by investigating the eigenvalues and understanding *their* behavior. This approach, although often applied, is ill-suited to the study of non-normal matrices.

We will begin by demonstrating, by way of an example, what can go wrong if we rely solely on eigenvalues during our analysis of matrices. We then introduce the polynomial numerical hull and present a method of computing its boundary based on a bisection method combined with an explicitly tracked front.

The Polynomial Numerical Hull of degree k

The polynomial numerical hull allows us to study polynomial functions of A by applying those polynomials to a set in the complex plane. The polynomial numerical null of degree k is defined as the set of complex points

$$\mathcal{G}_k(A) = \{z \in \mathbb{C} : ||p(A)|| \geq |p(z)| \quad \forall p \in \mathcal{P}_k\}, \tag{1}$$

where $\mathcal{P}_k$ is the set of all polynomials of degree k or less and $||\cdot||$ is, for the purposes of this project, the 2-norm. For example, a Toeplitz matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix}, \tag{2}$$

has the polynomial numerical hulls shown in Figure 1. Polynomials of degree 1 and 2 applied to this matrix will behave like the same polynomials applied to complex values from within the outer-most curve in Figure 1.

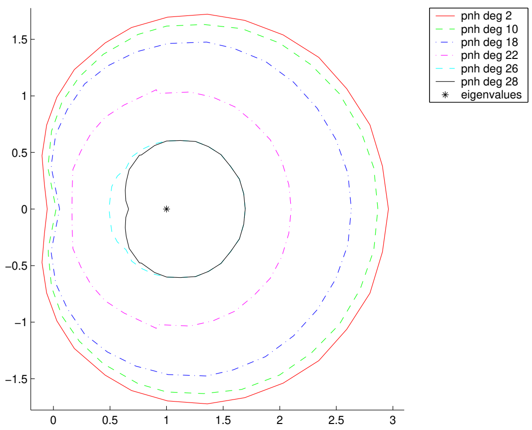

Figure 1: Polynomial numerical hulls of various degrees for the upper-tridiagonal matrix A. The (high-multiplity) eigenvalue is shown as a star.

Some Properties of Polynomial Numerical Hulls

- $\mathcal{G}_1$ is the field of values of the matrix.
- $\mathcal{G}_k$ always contains $\mathcal{G}_{k+1}$. That, is the $\mathcal{G}$ are nested.
- $\mathcal{G}_k$ always contains the spectrum.
- $\mathcal{G}_k$ is the spectrum if $k \geq n$ (specifally the degree of the minimum polynomial of A).
- $\mathcal{G}_k$ is invariant under unitary transforms.

An Investigation: Gauss-Seidel

Consider the iterative Gauss-Seidel method for solving a linear system $Cx = b$. That is, iterating the sequence

$$x_{k+1} := Ax_k + c \tag{3}$$

for some random x_0 and $A = I - L^{-1}C$, $c = L^{-1}b$ where L is the lower triangular part of C. The convergence and error bound analysis is typically done in terms of the spectral radius $\rho(A)$ and include the following statements (see, for example, [BF]):

1. *The sequence defined by (3) converges to the solution as $k \to \infty$ if and only if $\rho(A) < 1$.*
2. *The error decreases like $||A^k||$ as $k \to \infty$ (and thus like $\rho(A)$).*

It is important to note that these statements apply *asymptotically* for large k. *But what happens for small* k? Consider[1] the 30×30 square matrix

$$C = \begin{pmatrix} 1 & -1.16 & & \\ .16 & 1 & \ddots & \\ & \ddots & \ddots & -1.16 \\ & & .16 & 1 \end{pmatrix}, \qquad (4)$$

which has spectral radius $\rho(A) = 0.73$. However, as seen in Figure 2, there is initially a huge growth in error before things settle down to the convergence we expect.

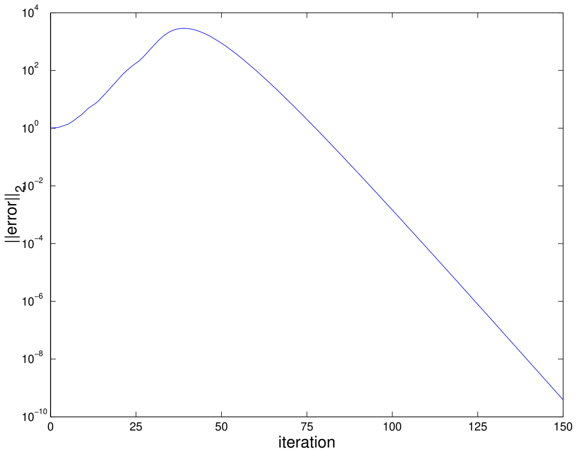

Figure 2: Error of solution produced by kth iteration of Gauss-Seidel

Now consider the polynomial numerical hull for A. An estimate of the boundary of $\mathcal{G}_{29}(A)$ is shown in Figure 3 and we can estimate the largest modulus for a point in $\mathcal{G}_{29}(A)$ to be 1.26. Since $||A^{29}|| \geq |z^{29}|$ for all z in $\mathcal{G}_{29}(A)$ (from the definition (1)), we have that error on x_{29} must be *at least* $1.26^29 \approx 800$ times the initial error. The polynomial numerical hull has thus provided a lower bound for the errors produced by the Gauss-Seidel method.

1 Computing the Polynomial Numerical Hull

An equivalent form of the polynomial numerical hull is given by

$$\mathcal{G}_k(A) = \left\{ z \in \mathbb{C} : \min_{p \in \mathcal{P}_k(0)} ||p(A - zI)|| \geq 1 \right\} \qquad (5)$$

where $\mathcal{P}_k(0)$ is the set of polynomials with $p(0) = 1$ of degree $\leq k$. Using this definition, we can check if any particular $z \in \mathbb{C}$ is in $\mathcal{G}_k(A)$ by defining $B = A - zI$ and computing

$$\min_{p \in \mathcal{P}_k(0)} ||p(B)||, \qquad (6)$$

[1] This example is used in [Gr1].

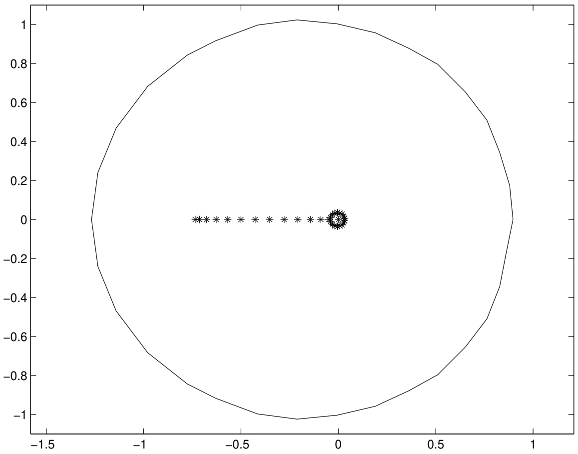

Figure 3: Boundary (black solid curve) of the polynomial numerical hull of degree 29 for matrix A with $n = 30$. (Eigenvalues shown as stars).

and then comparing the result to 1. A Matlab library for semidefinite programming, SDPT3 (see [TTT]), will perform this calculation. We use this as a black box code.

A simple way to estimate the polynomial numerical hull is to start with a region known to contain $\mathcal{G}_k(A)$ (such as the field of values) and then create a grid over that region. Then simply test each grid point by checking if equation (6) is larger than 1.

Unfortunately, equation (6) is difficult to minimize and SDPT3 takes a (relatively) long time to do so. Thus, computing over a grid is not a very practical way to compute accuate polynomial numerical hull for large matrices.

Instead we begin with a boundary that is known to contain $\mathcal{G}_k$. We then track that front explicitly and move it inwards toward $\mathcal{G}_k$ using a bisection method.

Outline of Boundary Finding Algorithm

Start with `bdry` (an outer estimate of the boundary), `steps` (initial stepsizes for each node in `bdry`), `converged` (a boolean vector specifying whether each node satisfies a given tolerance `TOL`).

While $\sum$ `converged` $\neq 0$ do

> For each node `bdry(i)` with `converged(i)==0` do
>
> > Estimate the inward normal at `bdry(i)`.
> >
> > Beginning with a stepsize `steps(i)`, perform several steps of a bisection search in the direction of the inward normal to. Store the final stepsize in `steps(i)`. Check if the node has converged to within `TOL` of the boundary and set `converged(i)` accordingly.
>
> Check for nodes which are to close/to far from their neighbours and add/remove nodes as necessary.

Results

Figure 4 shows the result of running our algorithm on the Grcar matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & & \\ -1 & 1 & 1 & 1 & \ddots & \\ & -1 & 1 & 1 & \ddots & 1 \\ & & -1 & 1 & \ddots & 1 \\ & & & \ddots & \ddots & 1 \\ & & & & -1 & 1 \end{pmatrix}, \tag{7}$$

with $n = 32$ and using a a circle as the initial boundary. This agrees with Grcar polynomial numerical hulls in [Gr1].

In Figure 5, our algorithm has been applied to Matlab's 5×5 `gallery(5)` matrix

$$\begin{pmatrix} -9 & 11 & -21 & 63 & -252 \\ 70 & -69 & 141 & -421 & 1684 \\ -575 & 575 & -1149 & 3451 & -13801 \\ 3891 & -3891 & 7782 & -23345 & 93365 \\ 1024 & -1024 & 2048 & -6144 & 24572 \end{pmatrix}. \tag{8}$$

which is said to be a difficult eigenvalue roblem. Although the results look promising in the first plot, things break down in the second plot (which has a tighter tolerance and a denser initial boundary) and results seem to not converge.

Conclusions and Future Work

Figure 5 suggests that the polynomial numerical hull for the `gallery(5)` matrix might not be connected. Trying to deal with separations or other topological changes using explicit front tracking methods is very difficult, so perhaps implicit tracking, based on Level-Set methods, would be appropriate for this problem.

Recall that since $\mathcal{G}_{k+1}$ is nested inside $\mathcal{G}_k$, the output of the computation for a lower degree PNH can be used as the initial guess for higher degrees. This could potentially save a lot of calculation time if more than one degree of polynomial numerical hull was required (as in Figure 1, for example).

In conclusion, the polynomial numerical hull is a relatively new idea, and thus although our approach is not particularly novel, it seems to improve upon existing codes and will hopefully aid others in discovering more about polynomial numerical hulls.

Acknowledgments

Thanks to Dr. Anne Greenbaum for supplying some very useful source code on short notice and for directing me to her paper [Gr1].

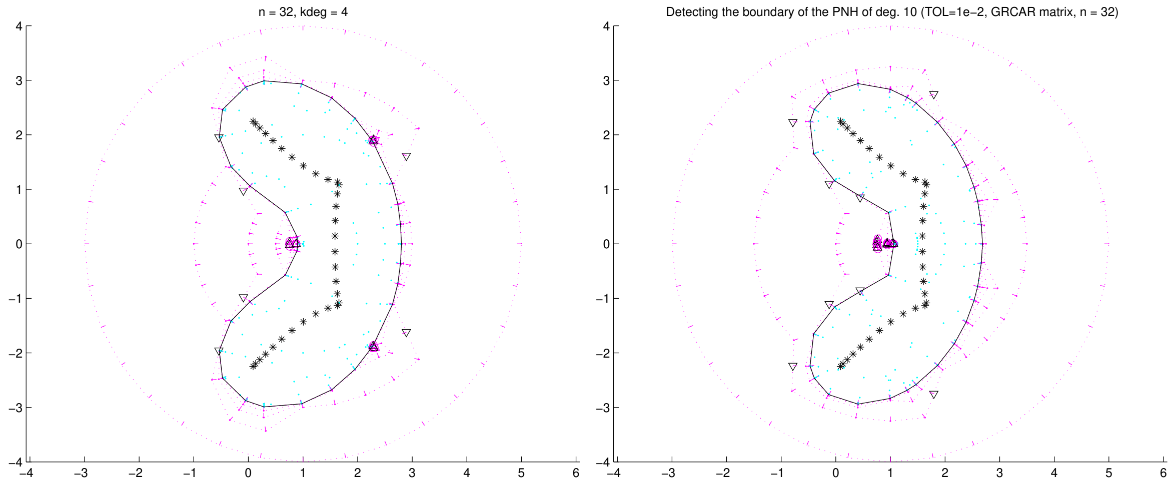

Figure 4: Estimate of $\mathcal{G}_4$ (left) and $\mathcal{G}_{10}$ (right) for the GRCAR matrix with $n = 30$. Eigenvalues are shown as black stars; the polynomial numerical hull as a solid black line; previous iterations are shown in dotted magenta with black triangles were nodes were added or removed; points tested inside the polynomial numerical hull are shown as blue dots.

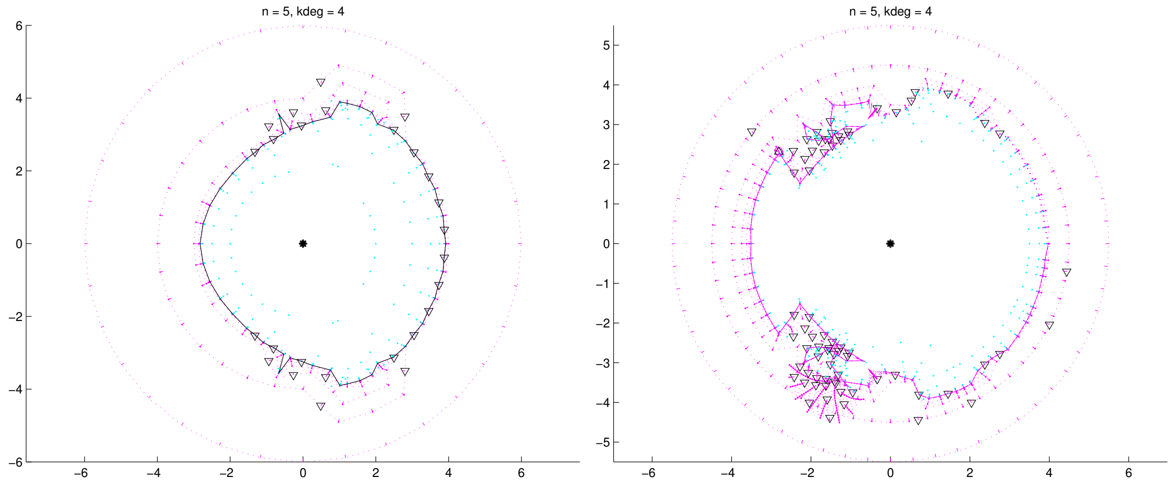

Figure 5: Polynomial numerical hull of degree 4 for the 5×5 matrix returned by Matlab (legend same as Figure 4). The left plot shows a tolerance of 2×10^{-2}, the right plot did not converge and was computed with a tolerance of 1×10^{-2}.

References

[BF] R. Burden, J. Faires, *Numerical Analysis*, 7th edition, Wadsworth Group, (2001).

[Gr1] A. Greenbaum, *Generalizations of the field of values useful in the study of polynomial functions of a matrix*, Lin. Alg. Appl. 347 (2002), 233–249.

[Gr2] A. Greenbaum, *Theory and applications of the polynomial numerical hull*, outline of presentation, private communication, (2002).

[TTT] K. Toh, M. Todd, R. Tütüncü, *SDPT3—a Matlab software package for semidefinite programming*, Optim. Meth. Softw. (1999).