Application of Kronecker Form to Systems and Control Theory

The *controllable subspace* of $\dot{x}(t) = Ax(t) + Bu(t)$ is the space in which the *system state* $x(t)$ can be "controlled" by choosing the *control input* $u(t)$ starting at $x(0) = 0$. This equation is used to model (feedback) control systems, where the $u(t)$ is chosen by the control system engineer to make $x(t)$ have certain desirable properties, like boundedness. From

$$\begin{aligned} x(t) &= \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau = \int_0^t \sum_{i=0}^{\infty} \frac{(t-\tau)^i}{i!} A^i Bu(\tau) d\tau \\ &= \sum_{i=0}^{\infty} A^i B \int_0^t \frac{(t-\tau)^i}{i!} u(\tau) d\tau \end{aligned}$$

one can prove the controllable space is $\operatorname{span}\{[B, AB, A^2B, \cdots, A^{n-1}B]\}$; any components of $x(t)$ outside this space cannot by controlled by varying $u(t)$. To compute this space in practice, in order to determine whether the physical system being modeled can in fact be controlled by input $u(t)$, one applies a QR-like algorithm to the singular pencil $[B, A - \lambda I]$. For details, see [77, 244, 245].

4.5.3 Nonlinear Eigenvalue Problems

Finally, we consider the *nonlinear eigenvalue problem* or *matrix polynomial*

$$\sum_{i=0}^{d} \lambda^i A_i = \lambda^d A_d + \lambda^{d-1} A_{d-1} + \cdots \lambda A_1 + A_0 \ . \tag{4.7}$$

Suppose for simplicity that the A_i are n-by-n matrices and A_d is nonsingular.

Definition 4.10 *The* characteristic polynomial *of the matrix polynomial (4.7) is $p(\lambda) = \det(\sum_{i=0}^{d} \lambda^i A_i)$. The roots of $p(\lambda) = 0$ are defined to be the* eigenvalues*. One can confirm that $p(\lambda)$ has degree $d \cdot n$, so there are $d \cdot n$ eigenvalues. Suppose γ is an eigenvalue. A nonzero vector x satisfying $\sum_{i=0}^{d} \gamma^i A_i x = 0$ is a* right eigenvector *for γ. A left eigenvector y is defined analogously by $\sum_{i=0}^{d} \gamma^i y^* A_i = 0$.*

Example 4.17 Consider Example 4.1 once again. The ODE arising there in equation (4.3) is $M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = 0$. If we seek solutions of the form $x(t) = e^{\lambda_i t} x_i(0)$, we get $e^{\lambda_i t}(\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i(0) = 0$, or $\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i(0) = 0$. Thus λ_i is an eigenvalue, and $x_i(0)$ is an eigenvector, of the matrix polynomial $\lambda^2 M + \lambda B + K$. $\diamond$

Since we are assuming A_d is nonsingular, we can multiply through by A_d^{-1} to get the equivalent problem $\lambda^d I + A_d^{-1}A_{d-1}\lambda^{d-1} + \cdots + A_d^{-1}A_0$. Therefore, to keep the notation simple, we will assume $A_d = I$ (see section 4.6 for the general case). In the very simplest case where each A_i is 1-by-1, i.e. a scalar, the original matrix polynomial is equal to the characteristic polynomial.

We can turn the problem of finding the eigenvalues of a matrix polynomial into a standard eigenvalue problem by using a trick analogous to the one used to change a high order ODE into a first order ODE. Consider first the simplest case $n = 1$, where each A_i is a scalar. Suppose γ is a root. Then the vector $x' = [\gamma^{d-1}, \gamma^{d-2}, \ldots, \gamma, 1]^T$ satisfies

$$
\begin{aligned}
Cx' &\equiv \begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{bmatrix} x' = \begin{bmatrix} -\sum_{i=0}^{d-1}\gamma^i A_i \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} \\
&= \begin{bmatrix} \gamma^d \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} = \gamma x' \ .
\end{aligned}
$$

Thus x' is an eigenvector and γ is an eigenvalue of the matrix C, which is called the *companion matrix* of the polynomial (4.7).

(The Matlab routine `roots` for finding roots of a polynomial applies the Hessenberg QR iteration of section 4.4.8 to the companion matrix C, since this is currently one of the most reliable, if expensive, methods known [98, 115, 239]. Cheaper alternatives are under development.)

The same idea works when the A_i are matrices. C becomes an $n \cdot d$-by-$n \cdot d$ *block companion matrix*, where the 1's and 0's below the top row become n-by-n identity and zero matrices, respectively. Also, x' becomes

$$
x' = \begin{bmatrix} \gamma^{d-1}x \\ \gamma^{d-2}x \\ \vdots \\ \gamma x \\ x \end{bmatrix}
$$

where x is a right eigenvector of the matrix polynomial. It again turns out that $Cx' = \gamma x'$.

Example 4.18 Returning once again to $\lambda^2 M + \lambda B + K$, we first convert it to $\lambda^2 + \lambda M^{-1}B + M^{-1}K$ and then to the companion matrix

$$C = \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} .$$

This is the same as the matrix A in equation 4.4 of Example 4.1. $\diamond$

Finally, Question 4.16 shows how to use matrix polynomials to solve a problem in *computational geometry*.

4.6 Summary

The following list summarizes all the canonicals forms, algorithms, their costs, and applications to ODEs described in this chapter. It also includes pointers to algorithms exploiting symmetry, although these are discussed in more detail in the next chapter. Algorithms for sparse matrices are discussed in Chapter 7.

- $A - \lambda I$
  - Jordan Form: For some nonsingular S,

$$A - \lambda I = S \cdot \mathrm{diag}\left(\ldots, \begin{bmatrix} \lambda_i - \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i - \lambda \end{bmatrix}^{n_i \times n_i}, \ldots \right) \cdot S^{-1} .$$

  - Schur form: For some unitary Q, $A - \lambda I = Q(T - \lambda I)Q^*$, where T is triangular.
  - Real Schur form of real A: For some real orthogonal Q, $A - \lambda I = Q(T - \lambda I)Q^T$, where T is real quasi triangular.
  - Application to ODEs: Provides solution of $\dot{x}(t) = Ax(t) + f(t)$.
  - Algorithm: Do Hessenberg reduction (Algorithm 4.6), followed by QR iteration to get Schur form (Algorithm 4.5, implemented as described in section 4.4.8). Eigenvectors can be computed from the Schur form (as described in section 4.2.1).
  - Cost: This costs $10n^3$ flops if eigenvalues only are desired, $25n^3$ is T and Q are also desired, and a little over $27n^3$ if eigenvectors are also desired. Since not all parts of the algorithm can take advantage of the Level 3 BLAS, the cost is

actually higher than a comparison with the $2n^3$ cost of matrix multiply would indicate: Instead of taking $(10n^3)/(2n^3) = 5$ times longer to compute eigenvalues than to multiply matrices, it takes 23 times longer for $n = 100$ and 19 times longer for $n = 1000$ on an IBM RS6000/590 [10, page 62]. Instead of taking $(27n^3)/(2n^3) = 13.5$ times longer to compute eigenvalues and eigenvectors, it takes 41 times longer for $n = 100$ and 60 times longer for $n = 1000$ on the same machine. So computing eigenvalues of nonsymmetric matrices is expensive. (The symmetric case is *much* cheaper; see Chapter 5).

- LAPACK: `sgees` for Schur form or `sgeev` for eigenvalues and eigenvectors; `sgeesx` or `sgeevx` for error bounds too.
- Matlab: `schur` for Schur form or `eig` for eigenvalues and eigenvectors.
- Exploiting symmetry: When $A = A^*$, better algorithms are discussed in Chapter 5, especially section 5.3.

• Regular $A - \lambda B$ $(\det(A - \lambda B) \not\equiv 0)$

- Weierstrass Form: For some nonsingular P_L and P_R,

$$A - \lambda B = P_L \cdot \operatorname{diag}\left(\text{Jordan}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix}^{n_i \times n_i}\right) P_R^{-1} \quad .$$

- Generalized Schur Form: For some unitary Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, where T_A and T_B are triangular.
- Generalized Real Schur Form of real A and B: For some real orthogonal Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^T$, where T_A is real quasi triangular and T_B is real triangular.
- Application to ODEs: Provides solution of $B\dot{x}(t) = Ax(t) + f(t)$, where the solution is uniquely determined, but may depend nonsmoothly on the data (*impulse response*).
- Algorithm: Hessenberg/triangular reduction followed by QZ iteration (QR applied implicitly to AB^{-1}).
- Cost: Computing T_A and T_B costs $30n^3$. Computing Q_L and Q_R in addition costs $66n^3$. Computing eigenvectors as well costs a little less than $69n^3$ in total. As before, Level 3 BLAS cannot be used in all parts of the algorithm.

  - LAPACK: `sgges` for Schur form or `sggev` for eigenvalues; `sggesx` or `sggevx` for error bounds too.
  - Matlab: `eig` for eigenvalues and eigenvectors.
  - Exploiting symmetry: When $A = A^*$, $B = B^*$ and B is positive definite, one can convert the problem to finding the eigenvalues of a single symmetric matrix using Theorem 4.12. This is done in LAPACK routines `ssygv`, `sspgv` (for symmetric matrices in "packed storage") and `ssbgv` (for symmetric band matrices).

- Singular $A - \lambda B$

  - Kronecker Form: For some nonsingular P_L and P_R,

$$A - \lambda B = P_L \cdot \mathrm{diag}\left(\text{Weierstrass}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & 1 & \lambda \end{bmatrix}^{n_i \times n_i}, \begin{bmatrix} 1 & & \\ \lambda & \ddots & \\ & \ddots & 1 \\ & & \lambda \end{bmatrix}^{m_i \times m_i}\right) P_R^{-1} .$$

  - Generalized Upper Triangular Form: For some unitary Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, where T_A and T_B are in generalized upper triangular form, with diagonal blocks corresponding to different parts of the Kronecker form. See [78, 244] for details of the form and algorithms.
  - Cost: The most general and reliable version of the algorithm can cost as much as $O(n^4)$, depending on the details of the Kronecker Structure; this is much more than for regular $A - \lambda B$. There is also a slightly less reliable $O(n^3)$ algorithm known [27].
  - Application to ODEs: Provides solution of $B\dot{x}(t) = Ax(t) + f(t)$, where the solution may be overdetermined or underdetermined.
  - Software: NETLIB/linalg/guptri.

- Matrix polynomials $\sum_{i=0}^{d} \lambda^i A_i$ [116]

  - If $A_d = I$ (or A_d is square and well-conditioned enough to replace each A_i by $A_d^{-1} A_i$), then linearize to get the standard problem

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda I \ .$$

– If A_d is ill-conditioned or singular, linearize to get the pencil

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda \begin{bmatrix} A_d & & & & \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix} .$$

4.7 References and Other Topics for Chapter 4

For a general discussion of properties of eigenvalues and eigenvectors, see [137]. For more details about perturbation theory of eigenvalues and eigenvectors, see [159, 235, 51], and chapter 4 of [10]. For a proof of Theorem 4.7, see [68]. For a discussion of Weierstrass and Kronecker canonical forms, see [108, 116]. For their application to systems and control theory, see [244, 245, 77]. For applications to computational geometry, graphics, and mechanical CAD, see [179, 180, 163]. For a discussion of parallel algorithms for the nonsymmetric eigenproblem, see [75].

4.8 Questions for Chapter 4

Question 4.1 *(Easy)* Let A be defined as in equation (4.1). Show that $\det(A) = \prod_{i=1}^{b} \det(A_{ii})$, and then that $\det(A - \lambda I) = \prod_{i=1}^{b} \det(A_{ii} - \lambda I)$. Conclude that the set of eigenvalues of A is the union of the sets of eigenvalues of A_{11} through A_{bb}.

Question 4.2 *(Medium; Z. Bai)* Suppose A is *normal*, i.e. $AA^* = A^*A$. Show that if A is also triangular, it must be diagonal. Use this to show that an n-by-n matrix is normal if and only if it has n orthonormal eigenvectors. Hint: Show that A is normal if and only if its Schur form is normal.

Question 4.3 *(Easy; Z. Bai)* Let λ and μ be distinct eigenvalues of A, let x be a right eigenvector for λ, and let y be a left eigenvecctor for μ. Show that x and y are orthogonal.

Question 4.4 *(Medium)* Suppose A has distinct eigenvalues. Let $f(z) = \sum_{i=-\infty}^{+\infty} a_i z^i$ be a function which is defined at the eigenvalues of A. Let $Q^*AQ = T$ be the Schur form of A (so Q is unitary and T upper triangular).

1. Show that $f(A) = Qf(T)Q^*$. Thus to compute $f(A)$ it suffices to be able to compute $f(T)$. In the rest of the problem you will derive a simple recurrence formula for $f(T)$.

2. Show that $(f(T))_{ii} = f(T_{ii})$, so that the diagonal of $f(T)$ can be computed from the diagonal of T.

3. Show that $Tf(T) = f(T)T$.

4. From the last result, show that the i-th superdiagonal of $f(T)$ can be computed from the $i-1$-st and earlier subdiagonals. Thus, starting at the diagonal of $f(T)$, we can compute the first superdiagonal, second superdiagonal and so on.

Question 4.5 *(Easy)* Let A be a square matrix. Apply either Question 4.4 to the Schur form of A, or equation (4.6) to the Jordan form of A, to conclude that the eigenvalues of $f(A)$ are $f(\lambda_i)$, where the λ_i are the eigenvalues of A. This result is called the *spectral mapping theorem.*

This question is used in the proof of Theorem 6.5 on page 320, and in section 6.5.6.

Question 4.6 *(Medium)* In this problem we will show how to solve the *Sylvester* or *Lyapunov* equation $AX - XB = C$, where X and C are m-by-n, A is m-by-m and B is n-by-n. This is a system of mn linear equations for the entries of X.

1. Given the Schur decompositions of A and B, show how $AX-XB = C$ can be transformed into a similar system $A'Y - YB' = C'$ where A' and B' are upper triangular.

2. Show how to solve for the entries of Y one at a time by a process analogous to back substitution. What condition on the eigenvalues of A and B guarantees that the system of equations is nonsingular?

3. Show how to transform Y to get the solution X.

Question 4.7 *(Medium)* Suppose $T = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$ is in Schur form. We want to find a matrix S so that $S^{-1}TS = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$. It turns out we can choose S of the form $\begin{bmatrix} I & R \\ 0 & I \end{bmatrix}$. Show how to solve for R.

Question 4.8 *(Medium; Z. Bai)* Let A be m-by-n and B be n-by-m. Show that the matrices

$$\begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}$$

are similar. Conclude that the nonzero eigenvalues of AB are the same as those of BA.

Question 4.9 *(Medium; Z. Bai)* Let A be n-by-n with eigenvalues $\lambda_1, \ldots, \lambda_n$. Show that

$$\sum_{i=1}^{n} |\lambda_i|^2 = \min_{\det(S)\neq 0} \|S^{-1}AS\|_F^2$$

Question 4.10 *(Medium; Z. Bai)* Let A be an n-by-n matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$.

1. Show that A can be written $A = H + S$ where $H = H^*$ is Hermitian and $S = -S^*$ is skew-Hermitian. Give explicit formulas for H and S in terms of A.
2. Show that $\sum_{i=1}^{n} |\Re\lambda_i|^2 \leq \|H\|_F^2$.
3. Show that $\sum_{i=1}^{n} |\Im\lambda_i|^2 \leq \|S\|_F^2$.
4. Show that A is normal ($AA^* = A^*A$) if and only if $\sum_{i=1}^{n} |\lambda_i|^2 = \|A\|_F^2$.

Question 4.11 *(Easy)* Let λ be a simple eigenvalue, and let x and y be right and left eigenvectors. We define the *spectral projection* P corresponding to λ as $P = xy^*/(y^*x)$. Prove that P has the following properties.

1. P is uniquely defined, even though we could use any nonzero scalar multiples of x and y in its definition.
2. $P^2 = P$. (Any matrix satisfying $P^2 = P$ is called a *projection matrix.*)

3. $AP = PA = \lambda P$. (These properties motivate the name *spectral* projection, since P "contains" the left and right invariant subspaces of λ.)

4. $\|P\|_2$ is the condition number of λ.

Question 4.12 *(Easy; Z. Bai)* Let $A = \begin{bmatrix} a & c \\ 0 & b \end{bmatrix}$. Show that the condition numbers of the eigenvalues of A are both equal to $\left(1 + \left(\frac{c}{a-b}\right)^2\right)^{1/2}$. Thus, the condition number is large if the difference $a - b$ between the eigenvalues is small compared to c, the offdiagonal part of the matrix.

Question 4.13 *(Medium, Z. Bai)* Let A be a matrix, x a unit vector ($\|x\|_2 = 1$), μ a scalar, and $r = Ax - \mu x$. Show that there is a matrix E with $\|E\|_F = \|r\|_2$ such that $A + E$ has eigenvalue μ and eigenvector x.

Question 4.14 *(Medium; Programming)* In this question we will use a Matlab program to plot eigenvalues of a perturbed matrix and their condition numbers . (It is available at HOMEPAGE/Matlab/eigscat.m.) The input is

a = input matrix
err = size of perturbation
m = number of perturbed matrices to compute

The output consists of three plots where each symbol is the location of an eigenvalue of a perturbed matrix:

'o' marks the location of each unperturbed eigenvalue.
'x' marks the location of each perturbed eigenvalue, where a real perturbation matrix of norm err is added to a.
'.' marks the location of each perturbed eigenvalue, where a complex perturbation matrix of norm err is added to a.

A table of the eigenvalues of A and their condition numbers is also printed.

Here are some interesting examples to try (for as large an m as you want to wait; the larger the m the better; m equal to a few hundred is good.

```
(1) a = randn(5) (if a does not have complex eigenvalues,
                  try again)
       err=1e-5, 1e-4, 1e-3, 1e-2, .1, .2

(2) a = diag(ones(4,1),1);  err=1e-12, 1e-10, 1e-8
```

```
(3) a=[[1 1e6 0     0]; ...
      [0 2   1e-3  0]; ...
      [0 0   3    10]; ...
      [0 0  -1     4]]
       err=1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3

(4) [q,r]=qr(randn(4,4));a=q*diag(ones(3,1),1)*q'
       err=1e-16, 1e-14, 1e-12, 1e-10, 1e-8

(5) a = [[1 1e3 1e6];[0 1 1e3];[0 0 1]],
       err=1e-7, 1e-6, 5e-6, 8e-6, 1e-5, 1.5e-5, 2e-5

(6) a = [[1   0   0   0   0   0]; ...
         [0   2   1   0   0   0]; ...
         [0   0   2   0   0   0]; ...
         [0   0   0   3 1e2 1e4]; ...
         [0   0   0   0   3 1e2]; ...
         [0   0   0   0   0   3]]
       err= 1e-10, 1e-8, 1e-6, 1e-4, 1e-3
```

Your assignment is to try these examples, and compare the regions occupied by the eigenvalues (the so-called *pseudospectrum*) with the bounds described in section 4.3. What is the difference between real perturbations and complex perturbations? What happens to the regions occupied by the eigenvalues as the perturbation err goes to zero? What is limiting size of the regions as err goes to zero (i.e. how many digits of the computed eigenvalues are correct)?

Question 4.15 *(Medium; Programming)* In this question we use a Matlab program to plot the diagonal entries of a matrix undergoing unshifted QR iteration. The values of each diagonal are plotted after each QR iteration, each diagonal corresponding to one of the plotted curves. (The program is available at HOMEPAGE/Matlab/qrplt.m, and also shown below.) The inputs are

a = input matrix
m = number of QR iterations

and the output is a plot of the diagonals.

Examples to try this code on are as follows (choose m large enough so that the curves either converge or go into cycles):

```
a = randn(6);
b = randn(6); a = b*diag([1,2,3,4,5,6])*inv(b);
```

```
  a = [[1 10];[-1 1]]; m = 300
  a = diag((1.5*ones(1,5)).\verb+^+(0:4)) +
      .01*(diag(ones(4,1),1)+diag(ones(4,1),-1)); m=30
```

What happens if there are complex eigenvalues?

In what order do the eigenvalues appear in the matrix after many iterations?

Perform the following experiment: Suppose a is n-by-n and symmetric. In Matlab, let perm=(n:-1:1). This produces a list of the integers from n down to 1. Run the iteration for m iterations. Let a=a(perm,perm); we call this "flipping" a, because it reverses the order of the rows and columns of a. Run the iteration again for m iterations, and again form a=a(perm,perm). How does this value of a compare with the original value of a? You should not let m be too large (try $m = 5$) or else round-off will obscure the relationship you should see. (See also Corollary 5.4 and Question 5.25.)

Change the code to compute the error in each diagonal from its final value (do this just for matrices with all real eigenvalues). Plot the log of this error versus the iteration number. What do you get asymptotically?

```
hold off
e=diag(a);
for i=1:m,
   [q,r]=qr(a);dd=diag(sign(diag(r)));r=dd*r;q=q*dd;a=r*q; ...
   e=[e,diag(a)];
end
clg
plot(e','w'),grid
```

Question 4.16 *(Hard; Programming)* This problem describes an application of the nonlinear eigenproblem to computer graphics, computational geometry and mechanical CAD; see also [179, 180, 163].

Let $F = [f_{ij}(x_1, x_2, x_3)]$ be a matrix whose entries are polynomials in the three variables x_i. Then $\det(F) = 0$ will (generally) define a two-dimensional surface S in 3-space. Let $x_1 = g_1(t)$, $x_2 = g_2(t)$, and $x_3 = g_3(t)$ define a (one-dimensional) curve C parameterized by t, where the g_i are also polynomials. We want to find the intersection $S \cap C$. Show how to express this as an eigenvalue problem (which can then be solved numerically). More generally, explain how to find the intersection of a surface $\det(F(x_1, \ldots, x_n)) = 0$ and curve $\{x_i = g_i(t),\ \ 1 \le i \le n\}$. At most how many discrete solutions can there be, as a function of n, the dimension d of F, and the maximum of the degrees of the polynomials f_{ij} and g_k?

Write a Matlab program to solve this problem, for $n = 3$ variables, by converting it to an eigenvalue problem. It should take as input a compact description of the entries of each $f_{ij}(x_k)$, and $g_i(t)$, and produce a list of the intersection points. For instance, it could take the following inputs:

- Array NumTerms(1:d,1:d), where NumTerms(i,j) is the number of terms in the polynomial $f_{ij}(x_1, x_2, x_3)$.
- Array Sterms(1:4, 1:TotalTerms), where TotalTerms is the sum of all the entries in NumTerms(.,.). Each column of Sterms represents one term in one polynomial: The first NumTerms(1,1) columns of Sterms represent the terms in f_{11}, the second Numterm(2,1) columns of Sterms represent the terms in f_{21}, and so on. The term represented by Sterms(1:4,k) is Sterm$(4,k)\cdot x_1^{\mathrm{Sterm}(1,k)}\cdot x_2^{\mathrm{Sterm}(2,k)}\cdot x_3^{\mathrm{Sterm}(3,k)}$.
- Array tC(1:3) contains the degrees of polynomials g_1, g_2 and g_3, in that order.
- Array Curve(1: tC(1)+tC(2)+tC(3)+3) contains the coefficients of the polynomials g_1, g_2 and g_3, one polynomial after the other, from the constant term to the highest order coefficient of each.

Your program should also compute error bounds for the computed answers. This will only be possible when the eigenproblem can be reduced to one for which the error bounds in Theorems 4.4 or 4.5 apply. You do not have to provide error bounds when the eigenproblem is a more general one. (For a description of error bounds for more general eigenproblems, see [10, 235].

Write a second Matlab program, which plots S and C for the case $n = 3$, and marks the intersection points.

Are there any limitations on the input data for your codes to work? What happens if S and C do not intersect? What happens if S lies in C?

Run your codes on at least the following examples. You should be able to solve the first 5 by hand, to check your code.

1. $g_1 = t$, $g_2 = 1+t$, $g_3 = 2+t$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.
2. $g_1 = t^3$, $g_2 = 1+t^3$, $g_3 = 2+t^3$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.
3. $g_1 = t^2$, $g_2 = 1+t^2$, $g_3 = 2+t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

4. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} 1 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 9 \end{bmatrix}$.

5. $g_1 = t^2$, $g_2 = 1+t^2$, $g_3 = 2+t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 8 \end{bmatrix}$.

6. $g_1 = t^2$, $g_2 = 1+t^2$, $g_3 = 2+t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & x_1 \\ x_3 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

7. $g_1 = 7 - 3t + t^5$, $g_2 = 1 + t^2 + t^5$, $g_3 = 2 + t^2 - t^5$, $F =$

$$\begin{bmatrix} x_1x_2 + x_3^5 & 3 - x_2^2 & 5 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 \\ x_2 - 7x_3^5 & 1 - x_1^2 + x_1x_2x_3^3 & 3 + x_1 + 3x_3 - 9x_2x_3 \\ 2 & 3x_1 + 5x_2 - 7x_3 + 8 & x_1^3 - x_2^4 + 4x_3^5 \end{bmatrix}$$

You should turn in

Mathematical formulation of the solution in terms of an eigenproblem.

The algorithm in at most 2 pages, including a road map to your code (subroutine names for each high level operation). It should be easy to see how the mathematical formulation leads to the algorithm, and how the algorithm matches the code.

At most how many discrete solutions can there be?

Do all compute eigenvalues represent actual intersections? Which ones do?

What limits does your code place on the input for it to work correctly?

What happens if S and C do not intersect?

What happens if S contains C?

Mathematical formulation of the error bounds.

The algorithm for computing the error bounds in at most 2 pages, including a road map to your code (subroutine names for each high level operation). It should be easy to see how the mathematical formulation leads to the algorithm, and how the algorithm matches the code.

Program listing.

For each of the seven examples, you should turn in

The original statement of the problem.

The resulting eigenproblem.

The numerical solutions.

Plots of S and C; do your numerical solutions match the plots?

The result of substituting the computed answers in the equations defining S and C: are they satisfied (to within roundoff)?