Multihomogeneous Homotopies

We first see how to embed a polynomial system in a multiprojective space and show how this is useful to an application in game theory. The construction of multihomogeneous homotopies leads to a multiprojective version of Bézout's theorem. We extend multihomogenization to general linear-product structures.

1 Multiprojective Space

Consider a 2-by-2 eigenvalue problem $\lambda \mathbf{x} = A\mathbf{x}$ viewed as a polynomial system:

$$f(\lambda, \mathbf{x}) = \left\{ \begin{array}{r} \lambda x_1 - a_{11}x_1 - a_{12}x_2 = 0 \\ \lambda x_2 - a_{21}x_1 - a_{22}x_2 = 0 \\ c_0 + c_1 x_1 + c_2 x_2 = 0. \end{array} \right. \quad (1)$$

The coefficients in the last equation are random (complex) numbers, used to ensure the eigenvectors are uniquely determined and the system has two isolated solutions for general choices of the matrix A.

While we know that any n-dimensional eigenvalue problem has in general n eigenvalues and corresponding eigenvectors, viewed as a polynomial system of n quadratic equations, the product of the degrees equals 2^n – the bound given by Bézout's theorem – and a homotopy based on this product will track 2^n solution paths, with only n of them to converge.

We may correct this problem if we embed the system in a multiprojective space, separating the eigenvalues from the coordinates of the eigenvector. In particular, in the system (1), we replace λ by λ_1/λ_0 and x_1 by z_1/z_0, x_2 by z_2/z_0. After clearing denominators, we then find

$$f(([\lambda_0 : \lambda_1], [z_0 : z_1 : z_2])) = \left\{ \begin{array}{r} \lambda_1 z_1 - a_{11}\lambda_0 z_1 - a_{12}\lambda_0 z_2 = 0 \\ \lambda_1 z_2 - a_{21}\lambda_0 z_1 - a_{22}\lambda_0 z_2 = 0 \\ c_0 z_0 + c_1 z_1 + c_2 z_2 = 0. \end{array} \right. \quad (2)$$

The coordinates of the system $([\lambda_0 : \lambda_1], [z_0 : z_1 : z_2]) \in \mathbb{P}^1 \times \mathbb{P}^2$ are equivalence classes, but now we can scale the coordinates of the eigenvalues independently of the coordinates of the eigenvector. As with projective space, at least one coordinate must be nonzero.

Solutions in $\mathbb{P}^1 \times \mathbb{P}^2$ at infinity are characterized by either $\lambda_0 = 0$ or $z_0 = 0$. We will verify that the system (2) has no solutions at infinity, by examination of the two possibilities. First, if $\lambda_0 = 0$, then both z_1 and z_2 must be zero as well (because $\lambda_1 \neq 0$). But if z_1 and z_2 are both zero, then by the last equation, z_0 must be zero as well, which cannot be. Second, if $z_0 = 0$, observe that then we have a system which is homogeneous in z_1 and z_2, which means that any multiple of a solution for (z_1, z_2) will also be a solution. As we just verified that $\lambda_0 \neq 0$, we may as well take $\lambda_0 = 1$ and consider then the system

$$f(([1 : \lambda_1], [0 : z_1 : z_2])) = \left\{ \begin{array}{r} \lambda_1 z_1 - a_{11} z_1 - a_{12} z_2 = 0 \\ \lambda_1 z_2 - a_{21} z_1 - a_{22} z_2 = 0 \\ c_1 z_1 + c_2 z_2 = 0. \end{array} \right. \quad (3)$$

Using the last equation, we may eliminate z_2, replacing z_2 by $-c_1/c_2 z_1$ (c_2 is random and thus nonzero), which leads to

$$f(([1 : \lambda_1], [0 : z_1])) = \left\{ \begin{array}{r} \lambda_1 z_1 - (a_{11} + a_{12}c_1/c_2) z_1 = 0 \\ -\lambda_1 c_1/c_2 z_1 - (a_{21} z_1 + a_{22} c_1/c_2) z_1 = 0. \end{array} \right. \quad (4)$$

After dividing out z_1 from both equations, we arrive at two equations with only λ_1 left as variable. Since the coefficients c_1 and c_2 were chosen at random, the system (4) can have no solutions.

In many practical applications, the variables have a meaning and occur often in groups, so that a multiprojective embedding is often quite natural. As the number of solutions at infinity decrease, we arrive at a sharper bound on the number of isolated solutions and we can then also construct more efficient homotopies.

2 Nash Equilibria

The application we consider comes from economics, in particular the theory of games. We follow [8], see also [1] and [7].

A small example of a non-cooperative game consists of three players, we call them a, b, and c, which can each choose from two different strategies. For each choice of strategy there is a certain payoff. The data for the game is then given by three payoff 2-by-2-by-2 matrices A, B, and C. Let x_1 and x_2, with $x_1 + x_2 = 1$ indicate the allocation of choices for a strategy made by player a. Similarly, for players b and c we have the respective variables (y_1, y_2) and (z_1, z_2), each with $y_1 + y_2 = 1$ and $z_1 + z_2 = 1$. The values for these variables could be real numbers in $[0, 1]$, for example: the percentage of money to allocate to a particular stock; or in a discrete problem setting, e.g.: for fishermen deciding whether to go out and fishing or stay home, the values of a variable could be the probability that a choice for an action is taken.

Then the payoffs α, β, and γ for players a, b, and c are respectively computed as

$$\alpha = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} A_{ijk} x_i y_j z_k, \quad \beta = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} B_{ijk} x_i y_j z_k, \quad \text{and} \quad \gamma = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} C_{ijk} x_i y_j z_k. \tag{5}$$

A vector $(x_1, x_2, y_1, y_2, z_1, z_2)$ is called a Nash equilibrium if no player can increase its payoff by changing its strategy while the other two players keep their strategies fixed. This means that for all (u_1, u_2), with $u_1 + u_2 = 1$ we have that at the equilibrium

$$\alpha \geq \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} A_{ijk} u_i y_j z_k, \quad \beta \geq \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} A_{ijk} x_i u_j z_k, \quad \text{and} \quad \gamma \geq \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} A_{ijk} x_i y_j u_k. \tag{6}$$

Because the expressions at the right hand side of each $\geq$ is linear in the (u_1, u_2), it suffices that the equalities be satisfied for all $(u_1, u_2) \in \{(1, 0), (0, 1)\}$, which reduces the inequalities to

$$\alpha \geq \sum_{j=1}^{2}\sum_{k=1}^{2} A_{1jk} y_j z_k \quad \text{and} \quad \alpha \geq \sum_{j=1}^{2}\sum_{k=1}^{2} A_{2jk} y_j z_k, \tag{7}$$

respectively corresponding to $(u_1, u_2) = (1, 0)$ and $(u_1, u_2) = (0, 1)$.

Using $\alpha = x_1 \sum_{j=1}^{2}\sum_{k=1}^{2} A_{1jk} y_j z_k + x_2 \sum_{j=1}^{2}\sum_{k=1}^{2} A_{2jk} y_j z_k$, $x_1 + x_2 = 1$ and $x_1 \geq 0$, $x_2 \geq 0$, we obtain

$$x_1 \left(\alpha - \sum_{j=1}^{2}\sum_{k=1}^{2} A_{1jk} y_j z_k \right) = x_2 \left(\alpha - \sum_{j=1}^{2}\sum_{k=1}^{2} A_{2jk} y_j z_k \right) = 0. \tag{8}$$

Similar expressions hold for (y_1, y_2) and (z_1, z_2). To reduce the number of variables, we eliminate $x_2 = 1 - x_1$, $y_2 = 1 - y_1$, and $z_2 = 1 - z_1$, and via subtractions we also eliminate α, β, and γ to obtain a polynomial system of three equations in x_1, y_1, and z_1.

A Nash equilibrium is totally mixed if all six probabilities are strictly positive. All totally mixed Nash equilibria can be computed by solving a polynomial system, which – this is the actually the whole point of this example – has a natural multihomogeneous structure, namely, we divide the variables into $\{\{x_1\}, \{y_1\}, \{z_1\}\}$.

It turns out that the corresponding Bézout bound for this type of application is sharp and there are games which have as many equilibria as this bound; see [3], [4], and [5]. The computation of Nash equilibria with Gambit is described in [9].

3 Linear-Product Start Systems

For a particular partition of the set of unknowns, we capture the structure of the polynomial by a table of degrees with respect to each set of variables. When considering the degree of a polynomial with respect to a set of variables, those variables not in the set are considered as constant.

For the eigenvalue problem, we have the natural partition $\{\{\lambda\},\{\mathbf{x}\}\}$ and we record the degrees of the equations as follows:

$$f(\lambda,\mathbf{x}) = \left\{ \begin{array}{lcr} f_1 & : & \lambda x_1 - a_{11}x_1 - a_{12}x_2 = 0 \\ f_2 & : & \lambda x_2 - a_{21}x_1 - a_{22}x_2 = 0 \\ f_3 & : & c_0 + c_1 x_1 + c_2 x_2 = 0 \end{array} \right. \qquad \begin{array}{cc c|c} \text{eq.} & : & \lambda & \mathbf{x} \\ \hline f_1 & : & 1 & 1 \\ f_2 & : & 1 & 1 \\ f_3 & : & 0 & 1 \end{array} \tag{9}$$

All polynomial systems with the same multihomogeneous structure as $f(\lambda,\mathbf{x}) = \mathbf{0}$ will have the same degree table as at the of (9). If we want to solve $f(\lambda,\mathbf{x}) = \mathbf{0}$ by homotopies, we look for a system with the same structure that is easier to solve. For this problem, we use systems whose equations are products of linear equations. The variables in each equation correspond to one set of variables. We take as many factors with variables of one set as its corresponding degree.

$$\begin{array}{ccc|c} \text{eq.} & : & \lambda & \mathbf{x} \\ \hline f_1 & : & 1 & 1 \\ f_2 & : & 1 & 1 \\ f_3 & : & 0 & 1 \end{array} \quad \Leftrightarrow \quad \begin{array}{ccc|c} \text{eq.} & : & \lambda & \mathbf{x} \\ \hline g_1 & : & \alpha_{10} + \alpha_{11}\lambda & \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 \\ g_2 & : & \alpha_{20} + \alpha_{21}\lambda & \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 \\ g_3 & : & 1 & \beta_{30} + \beta_{31}x_1 + \beta_{32}x_2 \end{array} \tag{10}$$

All coefficients for the linear equations are randomly chosen. Multiplying the equations at the right of (10) defines a linear-product start system $g(\lambda,\mathbf{x}) = \mathbf{0}$ and gives rise to the following homotopy:

$$h(\lambda,\mathbf{x},t) = (1-t) \left(\begin{array}{r} (\alpha_{10} + \alpha_{11}\lambda)(\beta_{10} + \beta_{11}x_1 + \beta_{12}x_2) \\ (\alpha_{20} + \alpha_{21}\lambda)(\beta_{20} + \beta_{21}x_1 + \beta_{22}x_2) \\ \beta_{30} + \beta_{31}x_1 + \beta_{32}x_2 \end{array} \right) + t \left(\begin{array}{r} \lambda x_1 - a_{11}x_1 - a_{12}x_2 \\ \lambda x_2 - a_{21}x_1 - a_{22}x_2 \\ c_0 + c_1x_1 + c_2x_2 \end{array} \right) = \mathbf{0}. \tag{11}$$

Just as the system $f(\lambda,\mathbf{x}) = \mathbf{0}$ has no solution at infinity when viewed in the multiprojective space $\mathbb{P}^1 \times \mathbb{P}^n$, also the start system $h(\lambda,\mathbf{x},0) = g(\lambda,\mathbf{x}) = \mathbf{0}$ has in this same multiprojective space no solutions at infinity.

As we solve a linear-product start system, we choose from every equation exactly one factor and for every set of variables we choose exactly as many factors as the number of elements in the set. Formally, we may execute on the degree table the same moves as when we are solving the linear-product start system. The formal root count for this example is

$$B_{\{\{\lambda\},\{\mathbf{x}\}\}} = 1_{(f_1,\lambda)} \times 1_{(f_2,\mathbf{x})} \times 1_{(f_3,\mathbf{x})} + 1_{(f_2,\lambda)} \times 1_{(f_1,\mathbf{x})} \times 1_{(f_3,\mathbf{x})} = 2. \tag{12}$$

The subscripts with each number indicate the coordinates of the degree in the table.

In general, for any partition Z of the set of unknowns, we compute the degrees of the polynomials in the system $f(\mathbf{x}) = \mathbf{0}$ with respect to every set of variables in Z and record these degrees in a table. This table defines then a linear-product start system $g(\mathbf{x}) = \mathbf{0}$ with the same multihomogeneous structure as $f(\mathbf{x}) = \mathbf{0}$. Formally, we may compute the number B_Z before we solve $g(\mathbf{x}) = \mathbf{0}$.

The number B_Z is called a multihomogenous Bézout bound. The computation of B_Z from the degree table resemble the same computations as when one computes a determinant, only without sign alternations. Therefore, B_Z is also a permanent of the degree table.

Although in many applications the variables already naturally occur in groups, finding the best partition Z (i.e.: leading to the lowest B_Z) is a hard computational problem. For a modest number of variables, a program could enumerate all possible partitions.

4 Bézout's Theorem in Multiprojective Space

Calling B_Z a Bézout bound is justified by the following theorem.

Theorem 4.1 (multihomogeneous theorem of Bézout) *Let Z be any partition of the set of unknowns $\mathbf{x}$ of the system $f(\mathbf{x}) = \mathbf{0}$ and let B_Z be its corresponding Bézout bound. Then the number of isolated solutions of $f(\mathbf{x}) = \mathbf{0}$ in $\mathbb{C}^n$ is bounded by B_Z.*

When using a linear-product start system in a homotopy, the geometric interpretation of the homotopy is that we deform the every polynomial in the system into a product of hyperplanes. By the proper choice of the multiprojective space, no solution is lost to infinity. The regularity of the solution paths is again proven by the application of the main theorem of elimination theory.

For more on multihomogeneous homotopies, we refer to [7].

5 Row Expansion for Permanents

For our application, we need a more generalized definition of the permanent. The set of variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is partitioned into m sets. The ith set has k_i variables. Below we sketch a recursive algorithm to compute the permanent.

Algorithm 5.1 Row Expansion for Computing the Permanent

Input: $A \in \mathbb{N}^{n\times m}$, $\mathbf{k} = (k_1, k_2, \ldots, k_m)$, $k_1 + k_2 + \cdots + k_m = n$;
i controls level of recursion, call with $i = 1$.
Output: $\mathrm{per}(A, \mathbf{k})$ permanent of A with respect to $\mathbf{k}$.

```
s := 0; k' := k;
for j from 1 to m do
  if k'_j ≠ 0 and a_ij ≠ 0 then
    k'_j := k'_j − 1;
    if i = n then
      s := s + a_ij;
    else
      s := s + a_ij × per(A, k', i + 1);
    end if;
    k'_j := k'_j + 1;
  end if;
end for;
return s.
```

The recursion level i selects one a_{ij} and because A has n rows, the maximal depth of recursion of this algorithm is n. The row number j corresponds to the group of variables. The algorithm tries all combinations of choices. Although computing the permanent has an exponential complexity — for a square n-by-n matrix all $n!$ permutations may contribute — the algorithm above takes into account the zeroes and will make far fewer combinations for sparse matrices.

6 Symmetric Polynomial Systems

Many polynomial systems arising in practical applications have an obvious permutation symmetry. For such systems, it suffices to compute only the generators of the solution set.

To describe symmetry mathematically, we commonly use groups, where each element of the group corresponds to a matrix. As we restrict ourselves to permutation symmetry, we use permutation matrices in $\mathbb{N}^{n\times n}$. Formally, a matrix representation of a finite group is given by

$$V : G \rightarrow \mathbb{N}^{n\times n} : e \mapsto V(e) = V_e \qquad (13)$$

such that the action of an element $e \in G$ on a vector $\mathbf{x}$ is calculated as the matrix-vector product $V_e\mathbf{x}$.

Let $f(\mathbf{x}) = \mathbf{0}$ be a polynomial system and G a finite group. The solution set $f^{-1}(\mathbf{0})$ is invariant under G if $\forall e \in G$ and $\forall \mathbf{z} \in f^{-1}(\mathbf{0})$: $V_e\mathbf{z} \in f^{-1}(\mathbf{0})$.

The set $\{ V_e\mathbf{z} \mid e \in G \}$ is called the orbit generated by $\mathbf{z}$. When the solution set is invariant under actions of a group G, it suffices to compute one solution per orbit.

Let V and W be matrix representations of permutation groups. The system $f(\mathbf{x}) = \mathbf{0}$ is (G, V, W)-symmetric if there holds:

$$\forall e \in G : \quad W_e f(\mathbf{x}) = f(V_e\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{C}^n. \qquad (14)$$

We also say that f has a (G, V, W)-symmetric structure. We call symmetric systems with $V = W$ equivariant.

As #G may be as large as $n!$ for the full permutation group, it suffices to verify the symmetry for the generators of the group G.

The (G, V, W) description may seem cumbersome, but it is necessary, as systems may have G-invariant solution sets, with different (G, V, W)-symmetric structures, as illustrated by the following example. Consider the following polynomial systems f_A and f_B:

$$f_A(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \\ x_3^2 - 1 = 0 \end{array} \right. \quad \text{and} \quad f_B(\mathbf{x}) = \left\{ \begin{array}{l} x_3^2 - 2 = 0 \\ x_1^2 - 2 = 0 \\ x_2^2 - 2 = 0. \end{array} \right. \qquad (15)$$

Let S_3 denote the group of all permutations of a set of 3 elements. The generators of S_3 are represented by the matrices

$$V_1 = \left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{array} \right] \quad \text{and} \quad V_2 = \left[\begin{array}{ccc} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{array} \right]. \qquad (16)$$

Both systems have an S_3-invariant solution set. Denote $W_1^{f_A}$ for $W_1^{f_A} f_A(\mathbf{x}) = f_A(V_1\mathbf{x})$ and $W_1^{f_B}$ for $W_1^{f_B} F_B(\mathbf{x}) = f_B(V_1\mathbf{x})$, then

$$W_1^{f_A} = \left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{array} \right] \quad \text{and} \quad W_1^{f_B} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{array} \right]. \qquad (17)$$

Because $(V_1, W_1^{f_A}) \neq (V_1, W_1^{f_B})$, the systems f_A and f_B do not have the same (S_3, V, W)-symmetric structure.

If $f(\mathbf{x}) = \mathbf{0}$ and $g(\mathbf{x}) = \mathbf{0}$ share the same (G, V, W)-symmetric structure, then all systems in the homotopy $h(\mathbf{x}, t) = \gamma(1 - t)g(\mathbf{x}) + tf(\mathbf{x}) = \mathbf{0}$, for $\gamma \in \mathbb{C}$ and $t \in [0, 1]$ are also (G, V, W)-symmetric. Then, to solve $f(\mathbf{x}) = \mathbf{0}$, we must track only one path per orbit.

7 Neural Networks

Our application of today comes from [6], studying the asymptotic behavior of an n-dimensional Lotka-Volterra system that models the dynamics of an adaptive cellular network.

The model consists of n interconnected cells. Each cell is characterized by a positive continuous function $X_i(t)$ representing the activity level of cell i at time t. The rate of change of activity level at the ith cell is described by Lotka-Volterra equations:

$$\frac{\partial X_i(t)}{\partial t} = X_i(t) \left[1 - cX_i(t) + \sum_{j=1}^{n} \delta_{ij} A_{ij}(t) X_j(t) \right], \quad i = 1, 2, \ldots, n, \tag{18}$$

where the connection matrix $\Delta = (\delta_{ij})$ is defined as

$$\delta_{ij} = \left\{ \begin{array}{rl} 0 & \mbox{if } i = j, \mbox{ or if cell } j \mbox{ is not connected to cell } i, \\ 1 & \mbox{if cell } j \mbox{ has an excitatory connection to cell } i, \\ -1 & \mbox{if cell } j \mbox{ has an inhibitory connection to cell } i. \end{array} \right. \tag{19}$$

The weight of the connection from cell j to cell i is given by the function $A_{ij}(t)$. The functions $A_{ij}(t)$ are defined by n^2 autonomous ordinary differential equations:

$$\frac{\partial A_{ij}(t)}{\partial t} = \left[X_i(t) X_j(t) - A_{ij}(t) \right] / T, \tag{20}$$

for some positive parameter T.

An interior critical point of the system (18) is any positive vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ satisfying

$$1 - cx_i + \sum_{j=1}^{n} \delta_{ij} x_i x_j^2 = 0, \quad i = 1, 2, \ldots, n. \tag{21}$$

To find all asymptotically stable critical points of the system, it is important to solve the polynomial system (21).

For certain specific choices of the connection matrix Δ, the solutions to (21) can all be found analytically, but for an arbitrary connection matrix, there is no known method for finding the solutions in closed form. The paper [6] contains theorems on the number of interior critical points, depending on values for Δ and choices for the parameter c.

For $n = 3$ and $\delta_{ii} = 0$, $\delta_{ij} = 1$, for $i \neq j$, we obtain the following system:

$$\left\{ \begin{array}{l} 1 - cx_1 + x_1 x_2^2 + x_1 x_3^2 = 0 \\ 1 - cx_2 + x_2 x_1^2 + x_2 x_3^2 = 0 \\ 1 - cx_3 + x_3 x_1^2 + x_3 x_2^2 = 0. \end{array} \right. \tag{22}$$

We see that the solution set does not change under all permutations of the three variables. The symmetry group of 6 permutations is generated by two elements represented by the matrices V_1 and V_2 in (16).

As this application generates polynomial systems for any n, typically, the number of solutions will increase exponentially in n. Exploitation of symmetry will reduce the complexity of this problem.

8 Linear-Product Start Systems

We encountered linear-product start systems in homotopies which respect the multihomogeneous structure of a polynomial system. Recall that such start systems are based on a partition of the set of unknowns. To exploit the symmetry, it is often necessary to take different partitions for different equations. To increase the flexibility even more, we work with set structures.

A set structure is a sequence of arrays of subsets of the set of unknowns. We say that an array of subsets S supports the polynomial $p \in \mathbb{C}[\mathbf{x}]$ if for each monomial $\mathbf{x}^{\mathbf{a}}$ which occurs in p with a nonzero coefficient there at least a_k subsets in S that contain x_k, for $k = 1, 2, \ldots, n$.

For example, consider the following polynomial system:

$$f(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 x_2 + x_2^2 + x_1 + 1 = 0 \\ x_2^2 x_1 + x_1^2 + x_2 + 1 = 0 \end{array} \right. \quad \text{where} \quad \left(\begin{array}{c} \{\{x_1\},\{x_1,x_2\},\{x_2\}\} \\ \{\{x_2\},\{x_2,x_1\},\{x_1\}\} \end{array} \right) \tag{23}$$

is a supporting set structure for f. Note how $x_1^2 x_2 = x_1 x_1 x_2$ is distributed among the sets. This observation leads to an algorithm to construct supporting set structures for polynomials, distributing the variables in each monomial among the sets. For symmetric polynomial systems, it may suffice to create a supporting set structure for one polynomial, and then apply the group actions to generate the whole set structure. The resulting set structure will then share the same symmetry as the given polynomial system.

A set structure defines the structure of a linear-product start system. Every set leads to a linear equation, only those variables in the set appear with nonzero coefficient in the linear equation. For example:

$$\left(\begin{array}{c} \{\{x_1\},\{x_1,x_2\},\{x_2\}\} \\ \{\{x_2\},\{x_2,x_1\},\{x_1\}\} \end{array} \right) \Leftrightarrow \quad g(\mathbf{x}) = \left\{ \begin{array}{l} (c_{11}x_1 + c_{12})(c_{13}x_1 + c_{14}x_2 + c_{15})(c_{16}x_2 + c_{17}) = 0 \\ (c_{21}x_2 + c_{22})(c_{23}x_2 + c_{24}x_1 + c_{25})(c_{26}x_1 + c_{27}) = 0 \end{array} \right. \tag{24}$$

The coefficients $c_{ij} \in \mathbb{C}$ are chosen at random.

Solving the system $g(\mathbf{x}) = \mathbf{0}$ can be formalized in the calculation of a Bézout number B_S, based directly on the supporting set structure S:

$$\begin{array}{rcccccccc} B_S & = & \underset{\{x_1\}\{x_2\}}{1} & + & \underset{\{x_1\}\{x_2,x_1\}}{1} & + & \underset{\{x_1,x_2\}\{x_2\}}{1} & + & \underset{\{x_1,x_2\}\{x_2,x_1\}}{1} \\ & + & \underset{\{x_1,x_2\}\{x_1\}}{1} & + & \underset{\{x_2\}\{x_2,x_1\}}{1} & + & \underset{\{x_2\}\{x_1\}}{1} & = & 7. \end{array} \tag{25}$$

The sets underneath the formula (25) indicate the sets associated with the linear systems in $g(\mathbf{x}) = \mathbf{0}$ that lead to a solution for all random choices of the coefficients. Compared to the total degree, the gain of using $B_S = 7$ looks insignificant, except that we may exploit the symmetry more easily. Consider the following choice of the coefficients of $g(\mathbf{x}) = \mathbf{0}$:

$$g(\mathbf{x}) = \left\{ \begin{array}{l} (c_1 x_1 + c_2)(c_3 x_1 + c_4 x_2 + c_5)(c_6 x_2 + c_7) = 0 \\ (c_1 x_2 + c_2)(c_3 x_2 + c_4 x_1 + c_5)(c_6 x_1 + c_7) = 0. \end{array} \right. \tag{26}$$

Setting $x_2 = x_1$ collapses the system into one single cubic equation. Computing these fixed points may be done directly on $f(x_1, x_1) = \mathbf{0}$, so we avoid tracking three paths. Observe the other symmetries:

$$\left(\begin{array}{c} c_1 x_1 + c_2 \\ c_3 x_2 + c_4 x_1 + c_5 \end{array} \right) \Leftrightarrow \left(\begin{array}{c} c_3 x_1 + c_4 x_2 + c_5 \\ c_1 x_2 + c_2 \end{array} \right) \quad \left(\begin{array}{c} c_6 x_2 + c_7 \\ c_3 x_2 + c_4 x_1 + c_5 \end{array} \right) \Leftrightarrow \left(\begin{array}{c} c_3 x_1 + c_4 x_2 + c_5 \\ c_6 x_1 + c_7 \end{array} \right) \tag{27}$$

which reduces the remaining four paths to two.

9 Solving Linear-Product Systems

Formally, we could still define the number of solutions of a linear-product system via a generalized permanent, but the formal evaluation tends to have a combinatorial complexity. Instead we calculate the generalized Bézout bound via the solution of a generic linear-product system, as is done in [11].

Algorithm 9.1 Solving a linear-product system

Input: $S = (S_1, S_2, \ldots, S_n)$, n sets of subsets of $\mathbf{x}$, $S_i = (S_{i1}, S_{i2}, \ldots, S_{id_i})$, S supports g;

$$g = (g_1, g_2, \ldots, g_n),\ g_i = \prod_{j=1}^{d_i} h_{ij}(\mathbf{x}),\ h_{ij} = c_{ij0} + \sum_{x_k \in S_{ij}} c_{ijk} x_k,\ c_{ij0}, c_{ijk} \in \mathbb{C} \setminus \{0\}.$$

Output: $g^{-1}(\mathbf{0})$, the solutions of $g(\mathbf{x}) = \mathbf{0}$.

Let Solve(g, i, A, b) be a recursive function
  where i controls the depth of the recursion (initial call: $i = 1$),
  and work space $A \in \mathbb{C}^{n \times n}$, $b \in \mathbb{C}^{n \times 1}$.
Definition of Solve:
  if $i > n$ then
    $g^{-1}(\mathbf{0}) := g^{-1}(\mathbf{0}) \cup A^{-1}b$;
  else
    for$j$ from 1 to d_i do
      update A and b with the coefficients of h_{ij};
      if Rank(A) $= i$ then
        $g^{-1}(\mathbf{0}) := g^{-1}(\mathbf{0}) \cup$ Solve($g, i+1, A, b$);
      end if;
    end for;
  end if;
  return $g^{-1}(\mathbf{0})$.

The coefficient matrix A is stored in upper triangulated form so the cost of the rank computations is shared between various solutions.

10 A Generalized Bézout Theorem

The regularity of a homotopy which uses a linear-product start system provides a constructive proof (see e.g. [10] or [7]) for the following theorem:

Theorem 10.1 *Let S be a supporting set structure for the system $f(\mathbf{x}) = \mathbf{0}$. Let B_S be the number of solutions of a linear-product system $g(\mathbf{x}) = \mathbf{0}$ supported by S. Then the number of solutions in $\mathbb{C}^n$ of $f(\mathbf{x}) = \mathbf{0}$ is bounded by B_S.*

One can prove this theorem again with a homotopy, showing that by a generic choice of the coefficients of g, all solution paths will be regular and isolated. The multiprojective space to compactify $\mathbb{C}^n$ is more complicated to describe.

In the light of this theorem, we may interpret the homotopy $h(\mathbf{x}, t) = (1-t)g(\mathbf{x}) + tf(\mathbf{x}) = \mathbf{0}$, with $t \in [0, 1]$ as a symbolic deformation of the system f. As we let t go from 1 to 0, the polynomials in the system deform into products of hyperplanes.

11 The Cost of Polynomial Evaluations

Below is computation with Maple 12:

```
[> p := product(c[i,0] + sum(c[i,j]*x[j],j=1..7),i=1..3):
[> codegen[cost](p);
                      23 multiplications + 21 additions + 45 subscripts
[> q := expand(p):
[> codegen[cost](q);
                     511 additions + 2368 multiplications + 2719 subscripts
[> h := convert(q,horner,[seq(x[i],i=1..7)]):
[> codegen[cost](h);
                     511 additions + 1143 multiplications + 1655 subscripts
[> dp1 := diff(p,x[1]):
[> codegen[cost](dp1);
                      44 additions + 48 multiplications + 93 subscripts
[> dq1 := diff(q,x[1]):
[> codegen[cost](dp2);
                      168 additions + 653 multiplications + 781 subscripts
```

Above we defined a cubic polynomial as a product of three linear factors in seven variables. The evaluation cost of this product is directly proportional to the degree times the dimension. After expanding this polynomial, we obtain 512 terms. In addition to this more than tenfold increase in the space complexity, evaluating this polynomial costs more than one hundred times more. Although the conversion to a nested Horner scheme cuts this high cost almost in half, it still remains more than fifty times higher than the original cost. This effect is amplified if we expand also the derivatives in the calculation of the Jacobian as needed for Newton's method.

Three factors influence the cost of tracking solutions paths: (1) the number of corrector steps; (2) the cost of solving a linear system; and (3) the cost of evaluating the polynomials. Even already with a modest number of variables, the cost to evaluate polynomials could be the dominating factor.

Linear-product polynomials are more general cases of monomial products. The expression swell arising from the symbolic execution of the product rule for derivatives motivated the reverse mode of algorithmic differentiation [2].

12 Exercises

1. Embed the system (1) in ordinary projective space and compute the two solutions at infinity.

2. For the 2-homogeneous 2-dimensional eigenvalue problem (2) there are no solutions at infinity. Show that the same holds for a 2-homogeneous 3-dimensional eigenvalue problem. Can you generalize the arguments to an n-dimensional eigenvalue problem?

3. The generalized eigenvalue problem considers $\beta A\mathbf{x} = \alpha B\mathbf{x}$, for matrices A, B and corresponding pairs (α, β) of generalized eigenvalues. Given two matrices A and B, we look for all pairs (α, β) for which a nonzero $\mathbf{x}$ satisfies $\beta A\mathbf{x} = \alpha B\mathbf{x}$. Compute a multihomogeneous Bézout bound for this problem. Is this bound sharp?

4. Write a MATLAB (or Octave) function to compute the permanent of a matrix. You may also of course use Maple or Sage. The input to this function consists of an n-by-m matrix (with $m \leq n$) and a vector $\mathbf{k}$ of m positive integers $\mathbf{k} = (k_1, k_2, \ldots, k_m)$, with $k_1 + k_2 + \cdots + k_m = n$. The number k_i is the cardinality of the ith set of variables.

5. Use a computer algebra system to generate a polynomial system to compute all Nash equilibria for three players with each two strategies. Use the following data for the payoff matrices:

	111	112	121	122	211	212	221	222
$A =$	0	6	11	1	6	4	6	8
$B =$	12	7	6	8	1	12	8	1
$C =$	11	11	3	3	0	14	2	7

   (28)

   Compute the 3-homogeneous Bézout bound and solve the system, using `phc`, available for download at `http://www.math.uic.edu/~jan/download.html`.

6. Describe the symmetric structure of the system (22) in detail, listing the matrices V and W for the generators of the group.

7. For the system (23), compute a 2-homogeneous Bézout number B_Z, for the partition $Z = \{\{x_1\}, \{x_2\}\}$.

8. Compute a supporting set structure for the system (22) and a corresponding linear-product start system. Count the number of solutions and compare this number to the total degree of the system.

9. For the system (22), create a linear-product start system $g(\mathbf{x}) = \mathbf{0}$ that respects the same permutation symmetry as in (22). Solve $g(\mathbf{x}) = \mathbf{0}$ to compute the number of solution orbits. Taking into account the fixed points, how many solution paths need to be tracked when a symmetric homotopy is used?

10. Describe how to modify Algorithm 9.1 for symmetric linear-product systems. When a representation of a symmetry group is given on input, the algorithm computes only the generators of the solution set.

References

[1] R.S. Datta. Using computer algebra to find Nash equilibria. In J.R. Sendra, editor, *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC 2003)*, pages 74–79. ACM, 2003.

[2] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, second edition, 2008.

[3] R.D. McKelvey and A. McLennan. The maximal number of mixed equilibria. *Journal of Economic Theory*, 72(2):411–425, 1997.

[4] A. McLennan. The maximal number of real roots of a multihomogeneous system of polynomial equations. *Beiträge zur Algebra und Geometry/Contributions to Algebra and Geometry*, 40(2):343–350, 1999.

[5] A. McLennan. The expected number of real roots of a multihomogeneous system of polynomial equations. *American Journal of Mathematics*, 124(1):49–73, 2003.

[6] V.W. Noonburg. A neural network modeled by an adaptive Lotka-Volterra system. *SIAM J. Appl. Math.*, 49(6):1779–1792, 1989.

[7] A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific Press, Singapore, 2005.

[8] B. Sturmfels. *Solving Systems of Polynomial Equations.* Number 97 in CBMS Regional Conference Series in Mathematics. AMS, 2002.

[9] T.L. Turocy. Towards a black-box solver for finite games: Computing all equilibria with Gambit and PHCpack. In *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and Its Applications*, pages 133–147. Springer-Verlag, 2008.

[10] J. Verschelde. *Homotopy Continuation Methods for Solving Polynomial Systems.* PhD thesis, K.U.Leuven, Dept. of Computer Science, 1996.

[11] S.M. Wise, A.J. Sommese, and L.T. Watson. Algorithm 801: POLSYS_PLP: a partitioned linear product homotopy code for solving polynomial systems of equations. *ACM Trans. Math. Softw.*, 26(1):176–200, 2000.