

## Chapter 1

# Polynomial Systems

The goal of this book is to describe numerical methods for computing the solutions of systems of polynomial equations. It is appropriate, therefore, to begin by defining “polynomials,” discussing how they may arise in science and engineering, describing in nontechnical terms what the “solutions of polynomial systems” look like and how we might represent these numerically. The last section of this chapter gives an overview of the rest of the book, to help the reader understand it in a larger perspective.

### 1.1 Polynomials in One Variable

As will be our habit throughout the book, we start with simple scenarios before proceeding to more general ones. A polynomial of degree  $d$  in one variable, say  $x$ , is a function of the form

$$f(x) = a_0x^d + a_1x^{d-1} + \cdots + a_{d-1}x + a_d, \quad (1.1.1)$$

where  $a_0, \dots, a_d$  are the *coefficients* and the integer powers of  $x$ , namely  $1, x, x^2, \dots, x^d$ , are *monomials*. In science and engineering, such functions usually have coefficients that are real numbers although sometimes they may be complex. Accordingly, we will consider  $f(x)$  as a function that maps complex numbers to complex numbers,  $f: \mathbb{C} \rightarrow \mathbb{C}$ . The notation  $\mathbb{C}[x]$  is often used to denote the set of all polynomials over the complex numbers in the variable  $x$ , so that we may write  $f(x) \in \mathbb{C}[x]$ . When we say that  $f(x)$  in Equation 1.1.1 is degree  $d$ , this implies that  $a_0 \neq 0$ ; otherwise, we say that  $f$  is at most degree  $d$ .

The “solution set” of the equation  $f(x) = 0$  is the set of all values of  $x \in \mathbb{C}$  such that  $f(x)$  evaluates to zero. We may write this as

$$f^{-1}(0) = \{x \in \mathbb{C} \mid f(x) = 0\}.$$

One of the great advantages of working over complex numbers is that, by the fundamental theorem of algebra (see Theorem 5.1.1), we know that as long as  $a_0 \neq 0$ ,  $f^{-1}(x)$  will consist of exactly  $d$  points, counting multiplicities. Thus, a data struc-

ture convenient for representing the solution of the equation is just a list of  $d$  complex numbers, say  $x_1^*, \dots, x_d^*$ , not all necessarily distinct. These are also called the *roots* of the polynomial. If some of the roots are repeated, then the *reduced solution set* is just the list of distinct roots. We know the solution set is complete and correct if

$$a_0 \prod_{i=1}^d (x - x_i^*) = a_0 x^d + a_1 x^{d-1} + \dots + a_{d-1} x + a_d.$$

That is, we expand the left-hand side and check that all the coefficients match.

It is possible to study polynomials over other rings, for example: the reals,  $\mathbb{R}[x]$ ; rational numbers,  $\mathbb{Q}[x]$ ; the integers,  $\mathbb{Z}[x]$ ; any finite field<sup>1</sup>,  $\mathbb{F}[x]$ ; or sometimes, in statements of theory, an unspecified field, usually denoted  $\mathbb{K}[x]$ . In one sense, there is no loss of generality in restricting our attention to  $\mathbb{C}[x]$ , for if we find all complex solutions of  $f(x) = 0$ , all real solutions will be contained therein, and similarly rational and integer solutions. However, the situation may be turned on its head if we ask other questions. As an example, suppose we seek the conditions for a sixth degree polynomial to be factorable over a field other than  $\mathbb{C}$ . Since the fundamental theorem of algebra tells us that all polynomials of degree greater than one in one variable factor over the complexes, we would have to consider the specific field in question to get an answer. Computer algebra systems deal extensively with polynomials over the rational numbers or over finite fields, since these permit exact calculation. And in the area of encryption, essential to secure digital communications, polynomials on finite fields are crucial. However, in engineering and science, real or complex numbers are of greatest concern, and it is in this arena that we focus our effort.

At this point, it is worth noting that our approach will be *numerical*, so in fact, all of the coefficients and the solutions we compute will be represented in floating point arithmetic. Typically, both will be only approximate, so that in reality we compute approximate solutions to a polynomial that is already an approximation of the original problem. This is the nature of almost all scientific computation. What is critical is that we have some estimate of the sensitivity of the problem so that we have assurance that the solutions are near the correct ones, or, as some would have it, that the problem that our solutions satisfy is near the one we want to solve.

There is an extensive literature on the numerical solution of polynomials in one variable. We will not delve into it here, as our focus is on multivariate cases. For low-degree polynomials in one variable, one approach is to reformulate the problem as finding the eigenvalues of the *companion matrix*,

$$A = \begin{pmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -\frac{a_d}{a_0} & -\frac{a_{d-1}}{a_0} & \dots & -\frac{a_1}{a_0} \end{pmatrix}, \quad (1.1.2)$$

---

<sup>1</sup>Most commonly, the integers modulo a prime number.

having ones on the superdiagonal and the coefficients of  $f$  in the last row. Since the characteristic polynomial of  $A$  is  $\det(xI - A) = f(x)/a_0$ , its eigenvalues are the roots of  $f$ . This formulation is convenient due to the wide availability of high-quality software for solving eigenvalue problems, and as documented in (Goedecker, 1994), it is a highly effective numerical approach. For polynomials with high degree, divide-and-conquer techniques may be better (Pan, 1997).

## 1.2 Multivariate Polynomial Systems

We may generalize the single-variable case in two ways: we may seek the simultaneous solution of several polynomials, and each of these may involve more than one variable. The formal definition of a polynomial, which includes the single variable case, is as follows.

**Definition 1.2.1 (Polynomial)** *A function  $f(x) : \mathbb{C}^n \rightarrow \mathbb{C}$  in  $n$  variables  $x = (x_1, \dots, x_n)$  is a polynomial if it can be expressed as a sum of terms, where each term is the product of a coefficient and a monomial, each coefficient is a complex number, and each monomial is a product of variables raised to nonnegative integer powers. Restating this in multidegree notation, let  $\alpha = (\alpha_1, \dots, \alpha_n)$  with each  $\alpha_i$  a nonnegative integer, and write monomials in the form  $x^\alpha = \prod_{i=1}^n x_i^{\alpha_i}$ . Then, a polynomial  $f$  is a function that can be written as*

$$f(x) = \sum_{\alpha \in \mathcal{I}} a_\alpha x^\alpha, \quad (1.2.3)$$

where  $\mathcal{I}$  is a finite index set and  $a_\alpha \in \mathbb{C}$ . The notation  $f \in \mathbb{C}[x_1, \dots, x_n] = \mathbb{C}[x]$  means  $f$  is a polynomial in the variables  $x$  with coefficients in  $\mathbb{C}$ . The total degree of a monomial  $x^\alpha$  is  $|\alpha| := \alpha_1 + \dots + \alpha_n$  and of the polynomial  $f(x)$  is  $\max_{\alpha \in \mathcal{I}: a_\alpha \neq 0} |\alpha|$ . When no confusion can result, we abbreviate total degree to degree.

In practice, polynomials often arise in unexpanded form, so that although in principle they can be expanded to the form of Equation 1.2.3, it is neither convenient nor numerically expedient to do so. Consequently, it is useful to make the following simple observations.

**Proposition 1.2.2** *If  $f \in \mathbb{C}[x]$  and  $g \in \mathbb{C}[x]$  are polynomials, then*

- $-f \in \mathbb{C}[x]$ ,
- $f + g \in \mathbb{C}[x]$ ,
- $f - g \in \mathbb{C}[x]$ ,
- $fg \in \mathbb{C}[x]$ , and
- $f^k \in \mathbb{C}[x]$ , for any nonnegative integer  $k$ .

*Proof.* Apply the distributive law to expand each expression into a sum of terms. □

Note that constants are polynomials too, so we may add, subtract or multiply by them as well.

Notice that the operation of division is missing. Although with suitable care, many of the techniques in this book can be extended to algebraic functions, which allow division, we will for the most part concentrate on polynomials. Mainly, this just means that before commencing to solve algebraic equations, we must clear denominators.<sup>2</sup>

The facts listed in Proposition 1.2.2 allow us to consider systems of polynomial functions given in “straight-line” form, convenient for both the analyst and for evaluation by computer.

**Definition 1.2.3 (Straight-Line Function)** *Beginning with a list of known quantities, consisting of internal constants,  $c$ , and a set of variables,  $x$ , a straight-line function specifies a finite sequence of elementary operations whose operands are among the known quantities and whose output is added to the list of known quantities. At termination, a subset of the known quantities are the function values,  $f(x)$ .*

**Definition 1.2.4** *A polynomial straight-line function is one whose elementary operations are limited to those listed in Proposition 1.2.2.*

When coding a function for numerical work, the analyst typically writes the function in a high-level description using the standard rules for precedence of operations and parentheses, as necessary. A compiler program parses this into a low-level sequence of unary and binary operations, producing a succession of intermediate results until the function values are reached. The following is a direct consequence of Proposition 1.2.2 and Definition 1.2.4.

**Corollary 1.2.5** *A polynomial straight-line function is a polynomial in  $x$  with coefficients that are polynomial in the internal constants  $c$ .*

One of our goals will be to solve such polynomial systems with a minimum of symbolic processing. Particularly, we do not wish to expand the polynomials into the form of Equation 1.2.3. For example, if  $f = 1 + x_1 + x_2 + \cdots + x_k$ , then the efficient way to evaluate  $f^n$  given values of  $x_1, \dots, x_k$  is to evaluate  $f$  and raise it to the  $n$ th power, as we would do in a straight-line program. Fully expanded,  $f^n$  has  $\binom{n+k}{k}$  terms, which can become rather large even for moderate  $n$  and  $k$ .

**Example 1.2.6** Consider the polynomial  $f(x, y) = (1 + 2.2x - 0.3y)^3$ . In fully expanded form, this has ten terms, namely

$$f(x, y) = 1 + 6.6 * x + 14.52 * x^2 + 10.648 * x^3 - 0.9 * y - 3.96 * x * y - 4.356 * x^2 * y + 0.27 * y^2 + 0.594 * x * y^2 - 0.027 * y^3.$$

---

<sup>2</sup> *Laurent polynomials*, which allow negative exponents, are treated briefly in § 8.5

An un-optimized evaluation of the function proceeding left to right and accumulating results term-by-term will not be very efficient. Compiling this into a sequence of elementary operations would give 27 operations in total. A computer code that only accepts fully expanded polynomials may optimize the evaluation procedure in some fashion. For example, the function is easily rearranged into nested Horner form as

$$f(x, y) = 1 + x * (6.6 + x * (14.52 + 10.648 * x)) \\ + y * (-0.9 + x * (-3.96 - 4.356 * x) + y * (0.27 + 0.594 * x - 0.027 * y)),$$

which reduces the operation count to 18. This is still far from the most efficient straight-line form, in which we first evaluate the quantity inside the parentheses and then cube the result. Compiled into a sequence of elementary operations, the evaluation proceeds as follows, using two temporary variables  $a$  and  $b$  and only five operations

$$\begin{aligned} a &\leftarrow (2.2 * x) \\ b &\leftarrow (1 + a) \\ a &\leftarrow (-0.3 * y) \\ b &\leftarrow (b + a) \\ f &\leftarrow (b^3) \end{aligned}$$

Here, the “ $\leftarrow$ ” symbol indicates that the right-hand expression should be evaluated and loaded into the variable at the left.

### 1.3 Trigonometric Equations as Polynomials

Problems in geometry and kinematics are often formulated using trigonometric functions. Very often these can be converted to polynomials. For example, equations involving  $\sin \theta$  and  $\cos \theta$  can be treated by replacing these with new indeterminates, say  $s_\theta$  and  $c_\theta$ , respectively, and then adding the polynomial relation  $s_\theta^2 + c_\theta^2 = 1$ . Once solution values for  $s_\theta$  and  $c_\theta$  have been found, the value of  $\theta$  is easily determined.<sup>3</sup> Sine or cosine of a multiple angle can always be reduced to a polynomial in sine and cosine of the angle, e.g.,  $\sin 2\theta = 2 \sin \theta \cos \theta$ , and the sine or cosine of sum and differences of angles can also be expanded into polynomials in the sines and cosines of the angles.

There are limits, of course: Not all trigonometric expressions can be converted to polynomials. Examples include  $x + \sin x$  and  $\sin x + \sin xy$ .

The reason that trigonometric expressions arising in practice are so often convertible to polynomials is that they usually have to do with angular rotations,

---

<sup>3</sup>A different maneuver is to use a new variable  $t$  and the substitutions  $\sin \theta = 2t/(1 + t^2)$  and  $\cos \theta = (1 - t^2)/(1 + t^2)$ . This avoids introducing a new equation at the cost of making the substitution quadratic.

Table 1.1 Solution Sets of Polynomial Systems

<b>Univariate</b> 1 Equation, 1 Variable solution points double roots, etc. Factorization, $\prod_i (x - a_i)^{\mu_i}$	<b>Multivariate System</b> $n$ Equations, $N$ Variables sol'n points, curves, surfaces, etc. sets with multiplicity Irreducible decomposition
<b>Numerical Representation</b>	
list of points	list of witness sets

whose main property is the preservation of length. Length relations are inherently polynomial, due to the Pythagorean Theorem.

#### 1.4 Solution Sets

We have already described above the nature of the solution set of a single polynomial in one variable, which can be represented numerically by a list of approximate solution points. As summarized in Table 1.1, the situation is more complicated for multivariate systems of polynomials. Such a system may have solution sets of several different dimensions: that is, a system could have isolated solution points (dimension 0), curves (dimension 1), surfaces (dimension 2), etc., all simultaneously. Moreover, just as a univariate polynomial may have repeated roots, a multivariate system may have solution sets that appear with multiplicity greater than one. Corresponding to the factorization of a univariate polynomial into linear factors, the solution set of a multivariate system can be broken down into its *irreducible components*. Isolated points are always irreducible, but higher dimensional sets may factor. For example, the quadratic  $x^2 + y^2$  factors into two lines  $(x + iy)(x - iy)$ , whereas the quadratic  $x^2 + y^2 - 1$  is an irreducible circle. The computation of a numerical representation of the *irreducible decomposition* of a multivariate polynomial system is the major topic in Part III of this book. This requires *witness sets*, a special numerical data structure. We postpone any further discussion of this until that point.

If  $f(x) : \mathbb{C}^N \rightarrow \mathbb{C}^n$  is a system of multivariate polynomials, we use the notations  $f^{-1}(0)$  and  $V(f)$  interchangeably to mean the solution set of  $f(x) = 0$ , i.e.,

$$V(f) = f^{-1}(0) = \{x \in \mathbb{C}^N \mid f(x) = 0\}.$$

The set  $V(f)$  contains no multiplicity information. When multiplicity is at issue, we will explicitly say so.

$V(f)$  is read as *the algebraic set associated to  $f$*  or the *algebraic set of  $f$* . The letter  $V$  in  $V(f)$  stands for variety, and indeed  $V(f)$  is sometimes referred to as the variety associated to  $f$ . As we will see at the start of § 12.2, the word variety often stands irreducible algebraic set. Because of the possible confusion that results, we have avoided using the word variety in this book.

Let us state now one caveat regarding real solutions. Higher dimensional solution

sets retain the property that the complex solution sets must contain the real solution sets. However, the containment can now be looser, because the real solution set may be of lower dimension than the complex component that contains it. For example, the complex line  $x + iy = 0$  on  $\mathbb{C}^2$  only contains one real point  $(x, y) = (0, 0)$ . Also, an irreducible complex component can contain more than one real component, as for example, the solution of  $y^2 - x(x^2 - 1) = 0$  is one complex curve that has two disconnected real components, one in the range  $x \geq 1$  and one in  $-1 \leq x \leq 0$ . Regrettably, the extraction of real components from complex ones is not developed enough for treatment in this book. We refer the reader to (Lu, Sommese, & Wampler, 2005).

This caveat notwithstanding, the complex solutions often give all the information that an analyst desires. In fact, although systems can, and often do, have solution sets at several dimensions, a scientist or engineer may often only care about isolated solution points. When circumstances dictate this, higher dimensional solutions may be justifiably labeled “degenerate” or “mathematical figments of the formulation.” Consequently, methods that are guaranteed to find the isolated solutions, without systematically finding the higher dimensional solution sets, are of significant value, and we will spend a large portion of this book discussing how to do this efficiently. Moreover, the numerical treatment of higher dimensional solutions will rest upon the ability to reformulate the problem so that at each dimension we are seeking a set of isolated solution points.

## 1.5 Solution by Continuation

The earliest forms of continuation tracked just one root as parameters of a problem were moved from a solved problem to a new problem. A notable example is the “bootstrap method” of (Roth, 1962; Freudenstein & Roth, 1963), which happened to be applied to problems involving polynomials but made no essential use of their properties. Beginning in the 1970’s, an approach to solving multivariate polynomial systems, called “polynomial continuation,” was developed. To just list a few of the early articles, there are (Drexler, 1977, 1978; Chow, Mallet-Paret, & Yorke, 1979; García & Zangwill, 1979, 1980; Keller, 1981; Li, 1983; Morgan, 1983). A more detailed history of the first period of the subject may be found in (Morgan, 1987). That period had relatively sparse use of algebraic geometry and centered on numerically computing all isolated solutions by means of total degree homotopies. A more recent survey of developments in finding all isolated solutions, taking into account which monomials appear in the equations, may be found in (Li, 2003). Methods for finding higher-dimensional solution sets are new; for these, we refer you to Part III of this book. In (Allgower & Georg, 2003, 1993, 1997), a broader perspective on continuation, including non-polynomial systems, is available.

By using algebraic geometry and specializing “homotopy continuation” to take advantage of the properties of polynomials, the algorithms can be designed to be

theoretically complete and practically very robust. Besides being general, polynomial continuation has the advantage that very little symbolic information needs to be extracted from a polynomial system to proceed. It often suffices, for example, just to know the degree of each polynomial, which is easily obtained without a full expansion into terms. For small systems, other approaches may be faster, and we will mention some of these. But these alternatives are quickly overwhelmed by systems of even moderate size, whereas continuation pushes out the boundary to include a much larger set of practical applications. For this reason, we highly recommend continuation and we devote nearly all of this book to that approach.

## 1.6 Overview

The main text of this book is divided into three main parts:

- Part I** an introduction to polynomial systems and continuation, along with material to familiarize the reader with one-variable polynomials and a chapter summarizing alternatives to continuation,
- Part II** a detailed study of continuation methods for finding the isolated solutions of multivariate polynomials systems, and
- Part III** in which continuation methods dealing with higher dimensional solution sets are presented.

As such, Part I is a combination of classical material and warm-ups for a serious look at the continuation method. Although we give brief looks at some alternative solution methods, beyond Part I, we concentrate exclusively on polynomial continuation. Part II is our attempt to put a common perspective on the major developments in that method from the 1980's and 1990's. Part III brings the reader to the cutting edge of developments.

The book also contains two substantial appendices. The first, Appendix A, provides extra material on some of the results we use from algebraic geometry. The style of the main text is intended to be understood without these extra details, but some readers will wish to dig deeper. Unfortunately, most of the existing mathematical texts take a more abstract point of view, necessitated by the mathematicians' drive to be general by encompassing polynomials over number fields other than the complexes. By collecting the basics of algebraic geometry over complex numbers, we hope to make this theory more accessible. Even mathematicians from outside the specialty of algebraic geometry might find the material useful in developing a better intuition for the field.

Appendix C is important for the serious student who wishes to work the exercises in the book. We give a user's guide to HomLab, a collection of Matlab routines for polynomial continuation. In addition to the basic HomLab distribution, there is a collection of routines associated with individual examples and exercises. These are documented in the exercises themselves.

## 1.7 Exercises

As the focus of this book is on numerical work, most of the exercises will involve the use of a computer and a software package with numerical facilities, such as Matlab. A free package called SciLab is also available. While most exercises require a modicum of programming in the way of writing scripts or at least interactive sessions with the packages, there are a few that require extensive programming.

Unless stated otherwise, statements such as `>>x=eig(A)` refer to Matlab commands, where “>>” is the Matlab prompt. Similar commands are available in the other packages mentioned above.

**Exercise 1.1 (Companion Matrices)** See Equation 1.1.2 for the definition of a companion matrix. In the following, `poly()` is a function that returns the coefficients of a polynomial given its roots, whereas `roots()` returns the roots given the coefficients.

- (1) Form the companion matrix for

$$f(x) = x^5 - 1.500x^4 - 0.320x^3 - 0.096x^2 + 0.760x + 0.156$$

and find its roots using an eigenvalue solver (in Matlab: `eig`).

- (2) Repeat the example using `>>f=poly([1, 1.5, -.4+.6i, -.4-.6i, -.2])` to form the polynomial and `>>roots(f)` to find its roots. (Note that in Matlab, `roots()` works by forming the companion matrix and finding its eigenvalues.)
- (3) **Wilkinson polynomials.** Use `>>roots(poly(1:n))` to solve the Wilkinson polynomial (Wilkinson, 1984) of order  $n$

$$\prod_{i=1}^n (x - i).$$

Explore how the accuracy behaves as  $n$  increases from 1 to 20. Why does it degrade? (Examine the coefficients of the polynomials.)

- (4) **Roots of unity.** Use `roots()` to solve  $x^n - 1 = 0$  for  $n = 1, \dots, 20$ . Compare answers to the roots of unity,  $e^{2\pi i/n}$ , where  $i = \sqrt{-1}$ .
- (5) **Repeated roots.** Solve  $x^6 - 12x^5 + 56x^4 - 130x^3 + 159x^2 - 98x + 24$  using `>>roots(poly([1, 1, 1, 2, 3, 4]))`. What is the accuracy of the triple root? What is the centroid (average) of the roots clustered around  $x = 1$ ?

**Exercise 1.2 (Straight-Line Polynomials: Efficiency)** Consider the determinantal polynomials  $p_n(x_1, \dots, x_{n^2})$ , where  $p_n$  is the determinant of the  $n \times n$  matrix having elements  $x_1, \dots, x_{n^2}$  listed row-wise. For example,

$$p_2(x_1, x_2, x_3, x_4) = \begin{vmatrix} x_1 & x_2 \\ x_3 & x_4 \end{vmatrix} = x_1x_4 - x_2x_3.$$

- (1) What is the degree of  $p_n$ ?

- (2) How many terms are there in the fully expanded form of  $p_n$ ? Using the sequence of operations implied by the fully expanded expression, how many arithmetic operations are required to numerically evaluate  $p_n$ , given numeric values of  $x_1, \dots, x_n$ ?
- (3) Using expansion by minors, how many operations are required to numerically evaluate  $p_n$ ?
- (4) What method does Matlab use to efficiently evaluate the determinant of an  $n \times n$  matrix? How many operations are required?

**Exercise 1.3 (Straight-Line Polynomials: Degree)** As mentioned in Definition 1.2.1, the *degree* of a monomial is the sum of the exponents appearing in it, e.g., the degree of  $xy^2z = x^1y^2z^1$  is 4 and the degree of a polynomial is the maximal degree of any of its terms. The purpose of this exercise is to find the degree of a straight-line polynomial without expanding it.

- (1) Given the degrees of  $f$  and  $g$ , what can you say about the degree of the result for each of the operations listed in Proposition 1.2.2?
- (2) Suppose each step of a straight-line program is given as an operator followed by a list of the addresses of one or two operands (as appropriate) and an address for the result. Design an algorithm to compute an upper bound on the degree of a straight-line polynomial. The complexity of the algorithm should be linear in the number of steps in the straight-line program.
- (3) Implement your algorithm in a language of your choice.
- (4) Can you think of a polynomial for which your algorithm computes a degree that is too high?

**Exercise 1.4 (A Trigonometric Problem)** Figure 1.1 shows a planar two-link robot arm, with upper arm length  $a$  and forearm length  $b$ . The end of the arm is at point  $(x, y)$  in the plane. Simple trigonometry gives the relations

$$x = a \cos \theta + b \cos \phi, \quad y = a \sin \theta + b \sin \phi. \quad (1.7.4)$$

- (1) Given  $a, b, x, y$ , use trigonometry to find  $\theta$  and  $\phi$ .
- (2) Reformulate Equations 1.7.4 as polynomials using the method suggested in § 1.3.
- (3) An alternative formulation is to let the coordinates of the “elbow” point be  $(u, v)$  and write equations for the squared distance from  $(u, v)$  to  $(0, 0)$  and from  $(u, v)$  to  $(x, y)$ . Do so.
- (4) Reduce the pair of equations in  $(u, v)$  to a single quadratic in  $u$ . What does this tell you about the number of solutions of the two-link arm?
- (5) What region of the plane can the endpoint of the arm reach? What happens to the solutions of the polynomial outside this range?

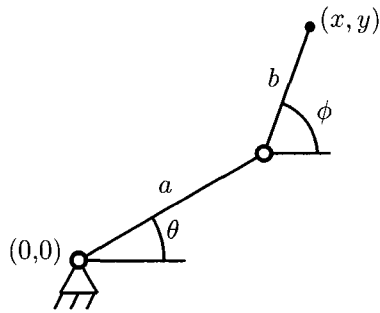


Fig. 1.1 A planar two-link robot arm. The triangle with hash marks indicates a grounded link, meaning that it cannot move. Open circles indicate hinge joints that allow relative rotation of the adjacent links.

**Exercise 1.5 (Solution Sets)** Create a system of three polynomials in three variables such that the solution set includes a surface, a curve, and several isolated points? (Hint: it is easier to do if the equations are written as products of factors, some of which appear in more than one equation.)