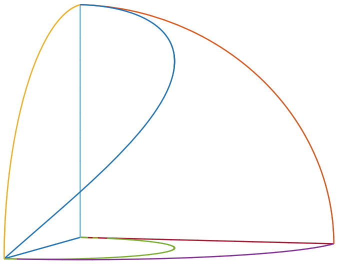
Chapter 2
Vectors and Graphics



We start this chapter by explaining how to use vectors in MATLAB, with an emphasis on practical operations on vectors in the plane and in space. Remember that n-dimensional vectors are simply ordered lists of n real numbers; the set of all such is denoted by $\mathbb{R}^n$. We discuss the standard vector operations, and give several applications to the computations of geometric quantities such as distances, angles, areas, and volumes. The bulk of the chapter is devoted to instructions for graphing curves and surfaces.

2.1 Vectors

In MATLAB, vectors are represented as lists of numbers or variables. You write a list in MATLAB as a sequence of entries encased in square brackets. Thus, you would enter **`v = [3, 2, 1]`** at the prompt to tell MATLAB to treat **`v`** as a vector with x, y, z coordinates equal to 3, 2, and 1, respectively.

You can perform the usual vector operations in MATLAB: vector addition, scalar multiplication, and the dot product (also known as the inner product or scalar product). Here are some examples, in which we use semicolons to suppress output:

```
>> a = [1, 2, 3];
>> b = [-5, -3, -1];
>> c = [3, 0, -2];
>> a + b

ans =
    -4    -1     2

>> 5*c

ans =
    15     0   -10
```


R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_2

```
>> dot(a, b)

ans =
   -14
```

The `dot(a, b)` command computes the dot product of the vectors `a` and `b` (the sum of the products of corresponding entries). As usual, you can use the dot product to compute lengths of vectors (also known as vector norms).

```
>> lengthofa = sqrt(dot(a, a))

ans =
    3.7417
```

Actually, MATLAB has an internal command that automates the numerical computation of vector norms.

```
>> [norm(a), norm(b), norm(c)]

ans =
    3.7417    5.9161    3.6056
```

Our attention throughout this book will be directed to vectors in the plane and vectors in (three-dimensional) space. Vectors in the plane have two components; a typical example in MATLAB is `[x, y]`. Vectors in space have three components, like `[x, y, z]`. The following principle will recur: *Vectors with different numbers of components do not mix.* As you will see, certain MATLAB commands will only work with two-component vectors; others will only work with three-component vectors. To convert a vector in the plane into a vector in space, you can add a zero to the end.

```
>> syms x y; p1 = [x, y]

p1 =
[ x, y]

>> s1 = [p1, 0]

s1 =
[ x, y, 0]
```

To project a vector in space into a vector in the x-y plane, you simply drop the final component.

```
>> syms x y z; s2 = [x, y, z]

s2 =
[ x, y, z]

>> p2 = s2(1:2)

p2 =
[ x, y]
```

The first place we notice a difference in the handling of two- or three-dimensional vectors is with the cross product, which only works on a pair of three-dimensional vectors. To compute the cross product in MATLAB, you can use the **`cross`** command. For example, to compute the cross product of the vectors **`a`** and **`b`** defined above, simply type

```
>> cross(a, b)

ans =
     7   -14     7
```

Note that the cross product is *anti-symmetric*; reversing the order of the inputs changes the sign of the output.

2.1.1 Applications of Vectors

Since MATLAB allows you to perform all the standard operations on vectors, it is a simple matter to compute lengths of vectors, angles between vectors, distances between points and planes or between points and lines, areas of parallelograms, and volumes of parallelepipeds, or any of the other quantities that can be computed using vector operations. Here are some examples.

2.1.1.1 The Angle Between Two Vectors

The fundamental identity involving the dot product is

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos(\varphi),$$

where φ is the angle between the two vectors. So, we can find the angle between **a** and **b** in MATLAB by typing

```
>> phi = acos(dot(a, b)/(norm(a)*norm(b)))

phi =
   2.2555
```

We can convert this value from radians to degrees by typing

```
>> phi*180/pi

ans =
   129.2315
```

2.1.1.2 The Projection Formula

One of the most useful applications of the dot product is for computing the projection of one vector in the direction of another. Given a nonzero vector **b**, we can always write another vector **a** uniquely in the form $\mathrm{proj}_{\mathbf{b}}(\mathbf{a}) + \mathbf{c}$, where $\mathbf{c} \perp \mathbf{b}$

and $\text{proj}_{\mathbf{b}}(\mathbf{a})$ is a scalar multiple of $\mathbf{b}$. To find the formula for the projection, write $\text{proj}_{\mathbf{b}}(\mathbf{a}) = x\mathbf{b}$ and take the dot product with $\mathbf{b}$. We obtain

$$\mathbf{a}\cdot\mathbf{b} = x\,\mathbf{b}\cdot\mathbf{b} + \mathbf{c}\cdot\mathbf{b} = x\,\|\mathbf{b}\|^2 + 0.$$

Thus $x = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|^2}$ and $\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|^2}\mathbf{b}$. By the way, even through the formula for $\text{proj}_{\mathbf{b}}(\mathbf{a})$ is nonlinear in $\mathbf{b}$, it is linear in $\mathbf{a}$. That is because the dot product is linear in each variable when the other variable is held fixed. Computing projections is easy in MATLAB. For example, with our given vectors, we obtain

```
>> (dot(a, b)/dot(b, b))*b

ans =
   2.0000    1.2000    0.4000
```

Exercise 2.1. Use MATLAB to compute the perpendicular component $\mathbf{c}$.

2.1.1.3 The Volume of a Parallelepiped

The volume of the parallelepiped spanned by the vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ is computed using the formula $V = |\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})|$. In MATLAB, it is easy to enter this formula.

```
>> abs(dot(a, cross(b, c)))

ans =
   7
```

2.1.1.4 The Area of a Parallelogram

The volume formula becomes an area formula if you take one of the vectors to be a unit vector perpendicular to the plane spanned by the other two vectors. In particular, given vectors $\mathbf{a}$ and $\mathbf{c}$, $\mathbf{a}\times\mathbf{c}$ is perpendicular to both of them, so a unit vector perpendicular to both $\mathbf{a}$ and $\mathbf{c}$ is $\frac{\mathbf{a}\times\mathbf{c}}{\|\mathbf{a}\times\mathbf{c}\|}$ and the area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{c}$ is just

$$\left|\frac{\mathbf{a}\times\mathbf{c}}{\|\mathbf{a}\times\mathbf{c}\|}\cdot(\mathbf{a}\times\mathbf{c})\right| = \frac{\|\mathbf{a}\times\mathbf{c}\|^2}{\|\mathbf{a}\times\mathbf{c}\|} = \|\mathbf{a}\times\mathbf{c}\|,$$

which you compute in MATLAB by typing

```
>> norm(cross(a, c))

ans =
   13.1529
```

In a similar fashion, you can use the standard mathematical formulas, expressed in MATLAB syntax, to

(i) project a vector onto a line or a plane;

(ii) compute the distance from a point to a plane;
(iii) compute the distance from a point to a line; and
(iv) check that lines and planes are parallel or perpendicular.

2.2 Parametric Curves

We assume that you already know how to use MATLAB's plotting commands to graph plane curves of the form $y = f(x)$. In fact, you can do so using either MATLAB's symbolic plotting command **`fplot`** or its numerical plotting command **`plot`**. The analogs are **`fplot3`** (symbolic) and **`plot3`** (numerical) for curves in space. There are also analogs, as we shall see, for surfaces in space, $z = f(x, y)$, namely **`fsurf`** or **`fmesh`** (symbolic) and **`surf`** or **`mesh`** (numerical). (*Note*: The symbolic **`f`** commands replaced the **`ez`** commands in MATLAB as of version R2016a.)

Now it is very common for both curves (in the plane or in space) and surfaces to be specified by parametric equations, rather than explicitly as just described. As we shall see, the same commands as above can be used to graph them—albeit with slightly different syntax. In this section, we will explain how to graph curves defined by parametric equations, both in the plane and in space, and also how to graph surfaces defined by parametric equations. But let us note: Henceforth in this book, and in the spirit of its subject matter, *we shall use symbolic plotting commands whenever feasible*. We shall resort to numerical plotting routines only when absolutely necessary.

For illustrative purposes, consider the parametrized unit circle

$$x = \cos t,\ y = \sin t,\ 0 \le t \le 2\pi.$$

These parametric equations mean that t is a parameter (ranging through the interval $[0, 2\pi]$), and that associated to each value of t is a pair (x, y) of values defined by the given formulas. As t varies, the points $(x(t), y(t))$ trace out a curve in the plane. Now let us draw the curve, first numerically, then symbolically

```
>> T = 0:0.1:2*pi; plot(cos(T), sin(T)); axis square
>> syms t; figure; fplot(cos(t), sin(t), [0, 2*pi]); axis square
```

Both commands result in the graph in Figure 2.1, although the tick marks differ slightly. Note that we inserted the command **`figure`** in the second instruction. Without it, the second circle would be superimposed on the first instead of it being created in a second graph.

Now let us start in earnest on parametric curves and surfaces by looking first at the spiral plane curve defined parametrically by the equations

$$x = e^{-t/10}(1 + \cos t), \quad y = e^{-t/10} \sin t, \quad t \in \mathbb{R}.$$

```
>> fplot(exp(-t/10)*(1 + cos(t)), exp(-t/10)*(sin(t)), [0 2*pi])
```

Figure 2.2 shows the resulting graph of the portion of the curve that results when the parameter runs from 0 to 2π.

Next we draw a larger portion of the curve.

```
>> fplot(exp(-t/10)*(1 + cos(t)), exp(-t/10)*(sin(t)), [0 20*pi])
```

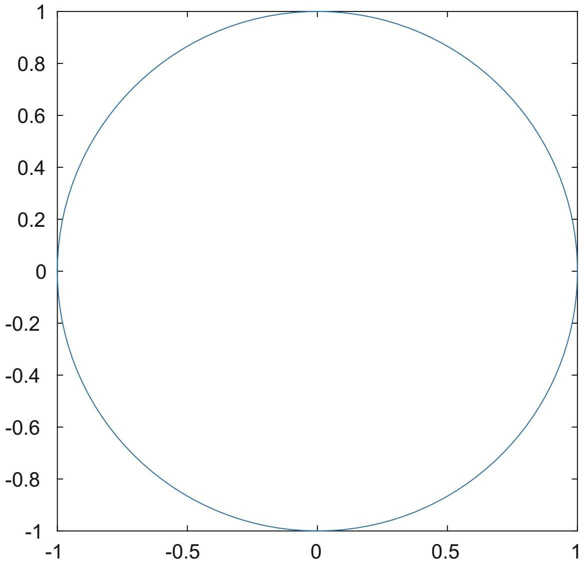


Fig. 2.1 The Unit Circle

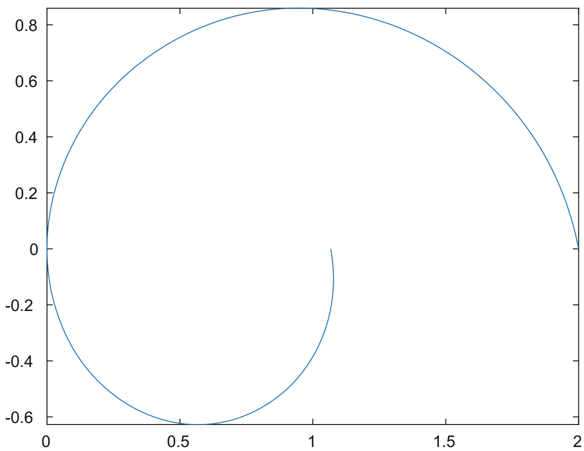


Fig. 2.2 Spiral Curve—One Rotation

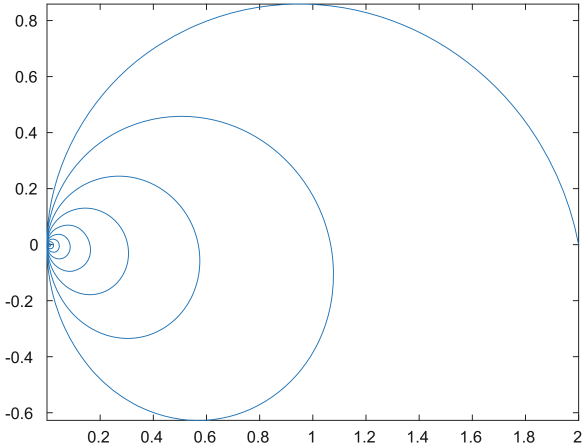


Fig. 2.3 Spiral Curve—Ten Rotations

Figure 2.3 shows the resulting graph of the portion of the curve that results when the parameter runs from 0 to 20π.

You can adjust this graph in various ways by using the **axis** command with various options. (See MATLAB online help for suggestions.)

Next let us look at a space curve defined by a set of parametric equations. As our example, we will take Viviani's curve, which is defined as the intersection of the sphere $x^2 + y^2 + z^2 = 4$ with the cylinder $(x-1)^2 + y^2 = 1$. The projection of Viviani's curve into the x-y plane is defined by the same equation that defines the cylinder; thus, it is a circle that has been shifted away from the origin. We can parametrize this circle in the plane by taking

$$x = 1 + \cos t, \quad y = \sin t, \quad -\pi \le t \le \pi.$$

Using this parametrization to solve the sphere's equation for z, we find that

$$\begin{aligned} z &= \pm\sqrt{4 - x^2 - y^2} \\ &= \pm\sqrt{4 - (1+\cos t)^2 - \sin^2 t} \\ &= \pm\sqrt{2 - 2\cos t} \\ &= \pm 2\sin\left(\frac{t}{2}\right). \end{aligned}$$

Now we will define Viviani's curve in MATLAB. After that, we will use **fplot3** to graph the part of the curve in the first octant.

```
>> syms t; x = 1 + cos(t); y = sin(t), z = 2*sin(t/2);
>> fplot3(x, y, z, [0 pi])
```

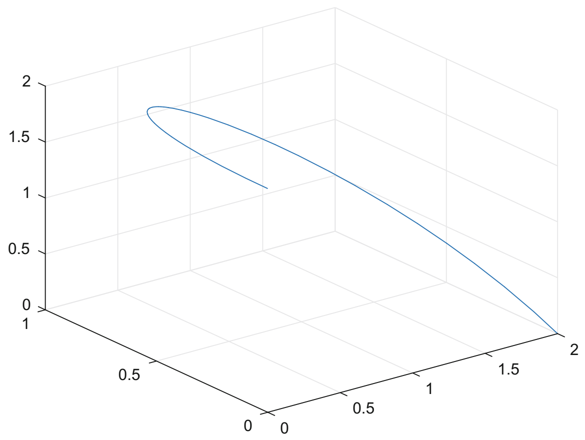


Fig. 2.4 Viviani's Curve—First Drawing

Figure 2.4 is not terribly illuminating; let us see if we can improve it. We shall do so by getting rid of the grid that, by default, surrounds all three-dimensional graphics in MATLAB. We shall also dispense with the labels and tick marks on the axes. It would also be nice to show the curve inside the sphere and cylinder that are used to define it. Therefore, we will show the arcs obtained by intersecting the sphere with the coordinate planes. Finally, we will include the projected circle in the *x*-*y* plane. The reader is encouraged to examine the code carefully to see how we achieved these objectives.

```
>> hold on
>> fplot3(sym(0), 2*cos(t/2), 2*sin(t/2), [0, pi])
>> fplot3(2*cos(t/2), sym(0), 2*sin(t/2), [0, pi])
>> fplot3(2*cos(t/2), 2*sin(t/2), sym(0), [0 pi])
>> fplot3(1+cos(t), sin(t), sym(0), [0, pi])
>> fplot3(sym(0), sym(0), t, [0 2]);
>> fplot3(sym(0), t, sym(0), [0 2]);
>> fplot3(t, sym(0), sym(0), [0 2])
>> title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis off;
```

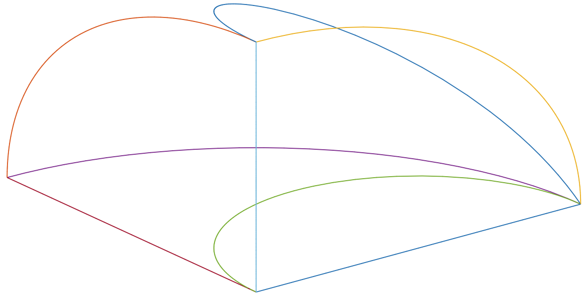


Fig. 2.5 Viviani's Curve—Second Drawing

Note that, since **fplot3** requires symbolic input, we had to specify the numerical input "0" as **sym(0)**. (Alternatively, we could have used function handles format for the input.) The picture still isn't very good; however, a minor change will improve it significantly. The main problem is the viewpoint (Figure 2.5) from which MATLAB has chosen to show us the graph. The default viewpoint is not in the first octant. This viewpoint is chosen generically, to make it unlikely that significant features of a random graph will be obscured. It has the definite disadvantage, however, that it changes the apparent directions of the x- and y-axes when compared with most mathematical textbooks. Choosing all positive values for the viewpoint will put your viewpoint into the first octant with the axes proceeding in the usual directions. In this case, we will make the change **view([10, 3, 1])** to get a better look at the graph. The result is shown in Figure 2.6. (In many cases, **[1, 1, 1]** is a good choice; you may need to experiment to find the best viewpoint.)

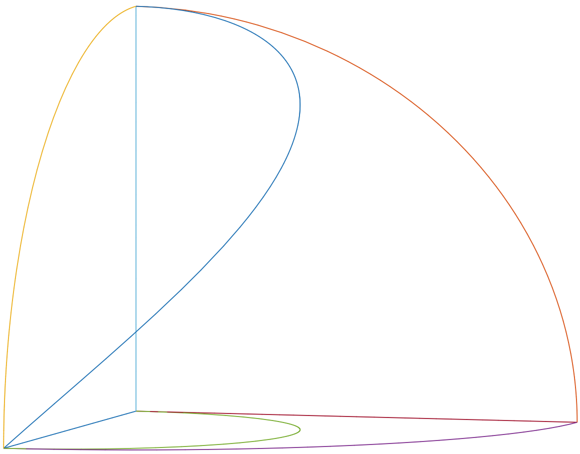


Fig. 2.6 Viviani's Curve—Third Drawing

2.3 Graphing Surfaces

Our next goal is to learn how to graph surfaces that are defined by a single equation $z = f(x, y)$. There are two (symbolic) graphing commands we can use to do that, namely **fmesh** and **fsurf**. The first produces a transparent mesh surface, the latter an opaque shaded one. We will illustrate them both. There are of course numerical analogs **mesh** and **surf**. We'll leave it to the reader to explore those if so desired.

Now, let's look at a simple example. We'll plot the function $f(x, y) = 1 - (x^2 + y^2)$ on the square $-1 \le x \le 1$, $-1 \le y \le 1$ in Figure 2.7.

```
>> syms x y; figure; fmesh(1 - x^2 - y^2, [-1, 1, -1, 1])
```

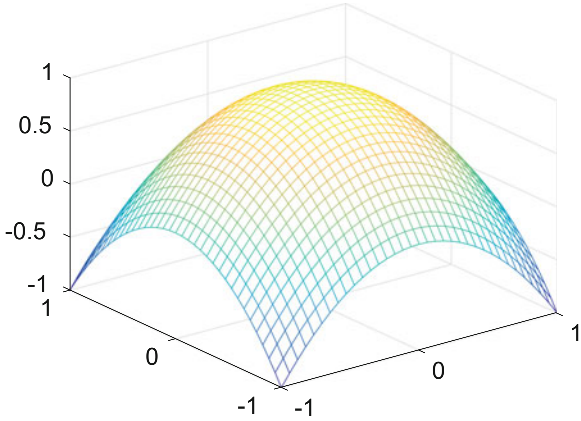


Fig. 2.7 A Paraboloid Above a Square

As in two dimensions, we can combine three-dimensional graphs via the **`hold on`** command. For example, let's draw a second surface over (see Figure 2.8) the same square.

```
>> hold on; fsurf(sin(6*x*y), [-1, 1, -1, 1])
>> title('Interwoven Surfaces')
```

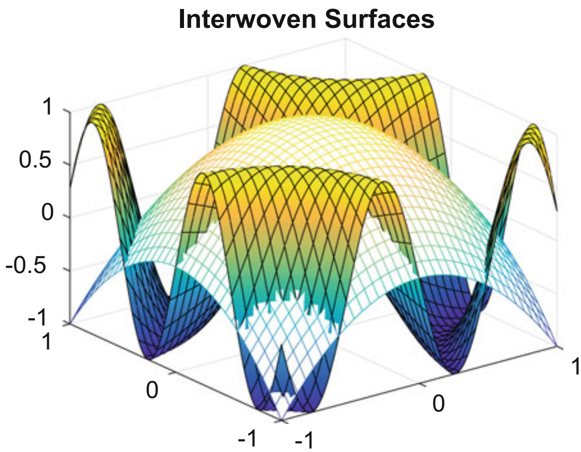


Fig. 2.8 A Paraboloid Interwoven with a Sinusoidal Surface

The **`fsurf`** command requires that the region in the *x*-*y* plane, over which the surface is plotted, must be a rectangle. Later on, we will see that it is helpful to be able to visualize portions of surfaces that lie over curved regions in the *x*-*y* plane.

This will be especially important when we study multiple integrals. In the meantime, here is a simple scheme to implement such a drawing, shown in Figure 2.9.

```
>> fsurf(sqrt(1-x^2-y^2)*heaviside(1-x^2-y^2), [-1, 1, -1, 1])
>> title('Hemisphere with Smooth Edges'), axis equal
```

You can consult the online help for more information on the **heaviside** function, but, essentially, **heaviside**(t) is 1 when $t > 0$ and vanishes otherwise.

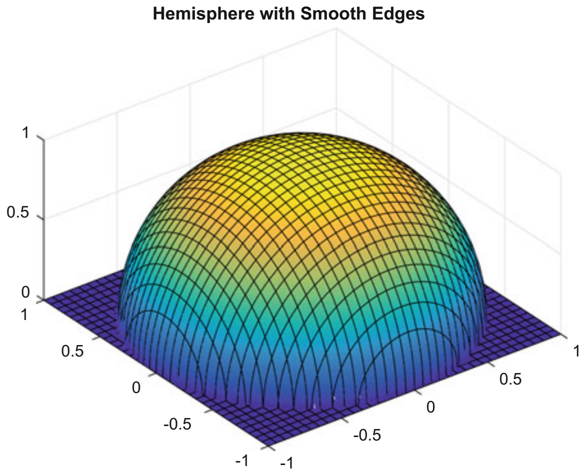


Fig. 2.9 The Hemisphere with no Rough Edges

2.4 Parametric Surfaces

The final topic is the plotting of surfaces that are defined by parametric equations. A parametric curve, being one-dimensional, depends on a single parameter. A parametric surface, being two-dimensional, requires us to use two parameters. Let's denote the parameters by u and v. For each parameter pair (u, v), we need to specify an associated point $(x(u, v), y(u, v), z(u, v))$ in space. As the parameter point (u, v) varies over its domain (which is some region in the plane), the associated point will trace out a surface in three-dimensional space. As an example, we will consider the surface whose parametric equations are

$$x = u^3, \quad y = v^3, \quad z = uv.$$

We often express this notion mathematically by writing

$$(x, y, z) = (u^3, v^3, uv).$$

In MATLAB, we can easily graph this surface. We will use some of the same "tricks" as above to render an uncluttered image, shown in Figure 2.10.

```
>> syms u v, fsurf(u^3, v^3, u*v, [-1, 1, -1, 1])
>> view([1, 1, 1]), title(''); xlabel(''); ylabel(''); zlabel('')
>> grid off; axis off;
```

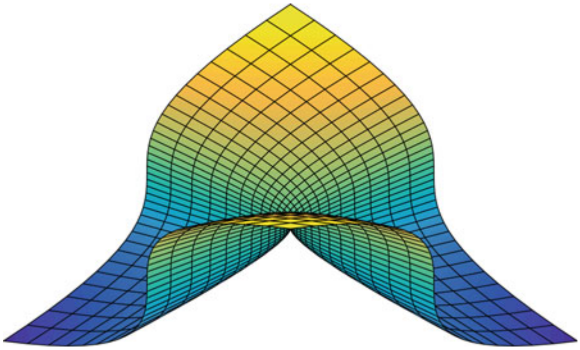


Fig. 2.10 A Cubic Surface

Standard methods for improving two-dimensional plots can also be employed for three-dimensional plots. For example, if your surfaces or curves look jagged or ill-defined, you can open the **Tools** tab in the menu bar of your plot window where you will find a host of tools for adjusting, enhancing and improving your graphs. Depending on your first picture, you may be able to estimate that the range you supplied (for the parameter or the base rectangle) should be adjusted. It is a simple matter to edit the input cell and then reinvoke the command. Your first picture will be superseded with a second picture (provided you do not have **`hold on`**). Labeling the axes, labeling the graph, changing the **`axis`** or **`view`**—these, and other techniques that you will learn to use as you produce more graphs, can greatly improve the quality of your plots.

Problem Set B. Vectors and Graphics

Problem 2.1. Find the distance between the two points $P = (0, 4.516, -5.298)$ and $Q = (-3.33, 0.234, 7.8)$.

Problem 2.2. Show that the point $P = (2, 0, 3)$ is equidistant from the two points $Q1 = (0.12, -1, 5.55)$ and $Q2 = (3.88, 1, 0.45)$.

Problem 2.3. Suppose that the points $P1 = (-2, -3, 5)$ and $P2 = (-6, 3, 1)$ are the endpoints of a diameter in a sphere. Find the equation of the sphere. Then compute the coordinates of any points on the line $z = 10y = -x$ which also lie on the surface of the sphere.

Problem 2.4. Let $\mathbf{a} = (2.9999, 400001, -6)$ and $\mathbf{b} = (0, -3.8765, 592320)$. Find $\mathbf{a} + \mathbf{b}$, $\|\mathbf{b}\|$ and $7\mathbf{a}$. Explain why $\|\mathbf{b}\|$ apparently equals the z-coordinate of $\mathbf{b}$, even though the y-coordinate is not zero.

Problem 2.5. Compute the angle (in degrees) that the vector $\mathbf{a} = -24.56\mathbf{i} + 44.689\mathbf{j}$ makes with the x-axis, measured counterclockwise from the x-axis.

Problem 2.6. Two tugboats are pulling a cruise ship. Tugboat 1 exerts a force of 1000 pounds on the ship and pulls in the direction 30 degrees north of due east. Tugboat 2 pulls in the direction 45 degrees south of due east. What force must Tugboat 2 exert to keep the ship moving due east?

Problem 2.7. Consider the vectors

$$\begin{aligned} \mathbf{a} &= (9, -3, 0.25), \\ \mathbf{b} &= (-3, -4, 60), \\ \mathbf{c} &= (-20.4, -6.2, 155.65). \end{aligned}$$

(a) Verify that $\mathbf{a}$ and $\mathbf{b}$ are perpendicular.

(b) The vector $\mathbf{c}$ lies in the same plane as $\mathbf{a}$ and $\mathbf{b}$. Resolve the vector $\mathbf{c}$ into its $\mathbf{a}$ and $\mathbf{b}$ components. Check your answer.

Problem 2.8. Find the angle (in degrees) between each of the following pairs of vectors:

(a) $\mathbf{a} = (2.467, -4.196, 0.433)$ and $\mathbf{b} = (-10.43, 9.344, 0)$.

(b) $\mathbf{a} = (-3.54, -10.79, 0.991)$ and $\mathbf{b} = (-1.398, 0, 6.443)$.

Problem 2.9. The following four vectors lie in a plane in 3-space. Do their endpoints determine a parallelogram? a rhombus? a square?

$$\mathbf{a} = (1, 1, 1), \qquad \mathbf{b} = (2, 3, 3), \qquad \mathbf{c} = (4, 2, 5), \qquad \mathbf{d} = (3, 0, 3).$$

Problem 2.10. In each of the following cases, find the cross product of the vectors $\mathbf{a}$ and $\mathbf{b}$, and then use it to find the angle between the two vectors.

(a) $\mathbf{a} = (-4.275, -2.549, 9.333)$, $\mathbf{b} = (6.302, -2.043, 0.444)$.

(b) $\mathbf{a} = (77, 88, 99)$, $\mathbf{b} = (22, 44, 66)$.

Problem 2.11. Prove the identity

$$\|\mathbf{a} \times \mathbf{b}\|^2 = \|\mathbf{a}\|^2 \|\mathbf{b}\|^2 - (\mathbf{a} \cdot \mathbf{b})^2$$

by assigning letter (i.e., variable) coordinates to **a** and **b** and evaluating both sides of the identity using MATLAB.

Problem 2.12. In this problem, we study the volumes of parallelepipeds.

(a) Find the volume of the parallelepiped determined by the three vectors:

$$\mathbf{a} = (8324, 5789, 2098),$$
$$\mathbf{b} = (9265, -246, 8034),$$
$$\mathbf{c} = (4321, -765, 7903).$$

(b) Now consider all parallelepipeds whose base is determined by the vectors $\mathbf{a} = (2, 0, -1)$ and $\mathbf{b} = (0, 2, -1)$, and whose height is variable $\mathbf{c} = (x, y, z)$. Assume that x, y, and z are positive and $\|\mathbf{c}\| = 1$. Use the triple product to compute a formula for the volume of the parallelepiped involving x, y, and z. Compute the maximum value of that volume in terms of x and y as follows. It is clear from the following formula:

$$\mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}) = \|\mathbf{c}\| \, \|\mathbf{a} \times \mathbf{b}\| \cos\theta,$$

where θ is the angle between **c** and the line perpendicular to the plane determined by **a** and **b**, that the maximum occurs when **c** is perpendicular to both **a** and **b**. Use the dot product to determine the vector **c** yielding the maximum value. (We will see in Chapter 7, *Optimization in Several Variables*, how to solve multivariable max-min problems.)

Problem 2.13. Find parametric equations for each of the following lines, then graph the line using **`fplot3`**.

(a) The line containing the points $(5.2, -4.11, 9)$ and $(0.3, 6.33, -2.34)$.

(b) The line passing through the point $(4, 0.35, -3.72)$ and parallel to the vector $\mathbf{v} = (4.66, -2.1, -3.51)$.

Problem 2.14. Find the distance from the point $(4.3, 5.4, 6.5)$ to the line whose parametric equations are $x = -1 + t$, $y = -2 + 2t$, $z = -3 + 3t$.

Problem 2.15. Draw the cylinder whose points lie at a distance 1 from the line $x = t$, $y = 10t$, $z = 0$. (Hint: To use **`fsurf`**, choose two unit vectors perpendicular to the line and use them and a vector along the line to parametrize the cylinder.)

Problem 2.16. For each of the following, find the equation of the plane and graph it:

(a) The plane containing the point $P_0 = (3.4, -2.6, 5)$ and having normal vector $\mathbf{n} = (-3.22, 1.2, 0.3)$.

(b) The plane containing the two lines

$$\begin{aligned} x &= 1 + t, & y &= 2 + t, & z &= 1 + 2t, \\ x &= 2t, & y &= 1 + t, & z &= -1 - t. \end{aligned}$$

Problem 2.17. Find the distance from the point $P = (100, 201, 349)$ to the plane $-213x - 438y + 301z = 500$.

Problem 2.18. Find parametric equations for the line formed by the intersection of the following two planes:

$$\begin{aligned} 2x - 3y + z &= 10, \\ -5x - 2y + 3z &= 15. \end{aligned}$$

Graph the two planes and the line of intersection on the same plot.

Problem 2.19. Consider the vector-valued functions

$$\begin{aligned} \mathbf{F}(t) &= (e^t, \sqrt{1+t}, \ln(1+t^2)), \\ \mathbf{G}(t) &= (\sin(t), \sec(t+1), (t-1)/(t+1)). \end{aligned}$$

Compute the functions $\mathbf{F} + \mathbf{G}$, $\mathbf{F} \cdot \mathbf{G}$, and $\mathbf{F} \times \mathbf{G}$.

Problem 2.20. Plot the following curves. In each case indicate the direction of motion. You will have to be careful when selecting the time interval on which to display the curve in order to get a meaningful picture. You may find **`axis`** useful in improving your plots.

(a) $\mathbf{F}(t) = (\cos t, \sin t, t/2)$.

(b) $\mathbf{F}(t) = (e^{-t} \sin t, e^{-t} \cos t, 1)$.

(c) $\mathbf{F}(t) = (t, t^2, t^3)$.

Problem 2.21. Graph the cycloid

$$\mathbf{r}(t) = (2(t - \sin t), 2(1 - \cos t))$$

and the trochoid

$$\mathbf{s}(t) = (2t - \sin t, 2 - \cos t)$$

together on the interval $[0, 4\pi]$. Find the coordinates of the four points of intersection. (Hint: Solve the equation $\mathbf{r}(t) = \mathbf{s}(u)$. Note the different independent variables for **r** and **s**—the points of intersection need not correspond to the same "time" on each curve. Also, since the coordinate functions are transcendental, you may need to use **`vpasolve`** rather than **`solve`**.) Use MATLAB to mark the four points on your graph.

Problem 2.22. Here's a problem to practice simultaneous plotting of curves and surfaces, as well as finding intersection points.

(a) Plot the two curves $2x^2 + 20y = -1$ and $y = x^4 - x^2$ on the same graph. Find the coordinates of all points of intersection.

(b) Plot the two surfaces $x^2 + y^2 + z^2 = 16$ and $z = 4x^2 + y^2$ and superimpose the plots. (The first surface is a sphere, the second is an elliptic paraboloid. The top half of the sphere will suffice here.) Use **`fcontour`** to plot the projection into the x-y plane of the curve of intersection of the two surfaces. You may find the option **`LevelList`** useful in identifying the precise curve.

Problem 2.23. This problem is about intersecting surfaces and curves. For helpful models, see the discussions of Viviani's curve, of "Graphing Surfaces," and of "Parametric Surfaces" in Chapter 2, *Vectors and Graphics*. You might wish to adjust the **`view`** in each of your three-dimensional plots.

(a) Draw three-dimensional plots of the paraboloid $z = x^2 + y^2$ and of the cylinder $(x-1)^2 + y^2 = 1$. Since the cylinder is not given by an equation of the form $z = f(x, y)$, you will need to plot it parametrically. You can use the parametrization

$$(1 + \cos t,\ \sin t,\ z)$$

with parameters t and z. Superimpose the two three-dimensional plots to see the curve where the surfaces intersect. Find a parametrization of the curve of intersection, and then draw an informative three-dimensional plot of this curve.

(b) Do the same for the paraboloid $z = x^2 + y^2$ and the upper hemisphere $z = \sqrt{1 - (x-1)^2 - y^2}$. This time the equation of the curve of intersection is a bit complicated in rectangular coordinates. It becomes simpler if you project the curve into the x-y plane and convert to polar coordinates. Apply **`solve`** to the polar equation of the projection to find $r(\theta)$, the formula for r in terms of θ. You can then parametrize the curve by

$$(r(\theta)\cos\theta,\ r(\theta)\sin\theta,\ r(\theta)^2),$$

with θ varying. You might find the option **`MaxDegree`** useful.

Glossary of MATLAB Commands

`abs` The *absolute value* function

`acos` The *arc cosine* function

`axis` Selects the ranges of x and y to show in a 2D-plot; or x, y and z in a 3D-plot

`clear` Clears values and definitions for variables and functions. If you specify one or more variables, then only those variables are cleared.

`cross` The cross product of two vectors

`dot` The dot product of two vectors

`double` Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

`fcontour` Plots the contour curves of a symbolic expression $f(x, y)$

`figure` Start a new graphic

`fminbnd` Find minimum of single-variable function on a fixed interval

`fplot` Easy function plotter

`fplot3` Easy 3D function plotter

`fsurf` Easy 3D surface plotter

`fzero` Finds (numerically) a zero of a function near a given starting value

`linspace` Generates a linearly spaced vector

`norm` Norm of a vector or matrix

`polarplot` Plots a curve in polar coordinates

`real` Follows **`syms`** to insure variables are real

`solve` Symbolic equation solver

`subs` Substitute for a variable in an expression

`syms` Set up one or more symbolic variables

`view` Specifies a point from which to view a 3D graph

`vpasolve` Finds numerical solutions to symbolic equations

Options to MATLAB Commands

`LevelList` Specifies the contour levels for **`fcontour`**

`MaxDegree` An option to **`solve`**, giving the maximum degree of polynomials for which MATLAB will try to find explicit solution formulas

Springer

http://www.springer.com/978-3-319-65069-2

Multivariable Calculus with MATLAB®
With Applications to Geometry and Physics
Lipsman, R.L.; Rosenberg, J.M.
2017, XII, 276 p. 87 illus., 86 illus. in color., Hardcover
ISBN: 978-3-319-65069-2