

Chapter 2. Vectors and the geographic coordinate system

© 1988 – 2017 by Vincent S. Cronin. Revised 31 January 2017.

2.1 Getting ready to learn

In this and in subsequent chapters, it is assumed that the reader has access to a computer connected to the web that has a functional copy of *Mathematica* by Wolfram Research. The current version of this chapter was developed in *Mathematica* version 8 and was revised in version 10. The Wolfram/*Mathematica* learning center is accessible at <http://www.wolfram.com/support/learn/>, and the “hands-on start to *Mathematica*” is available at <http://www.wolfram.com/broadcast/screencasts/handsonstart/>. Informative web documents accessible from Wolfram *MathWorld* (<http://mathworld.wolfram.com/>) are listed after the references at the end of this chapter. Developing the ability to code in a computer language is, in the author’s opinion, a necessity for any student in science, technology, engineering or mathematics.

2.2 Position vectors and the geographic coordinate system

For any point Q , the geographic-Cartesian coordinates $\{x_Q, y_Q, z_Q\}$ can be determined from the standard geographic coordinates $(\text{lat}_Q, \text{long}_Q)$ as follows:

$$x_Q = \cos[\text{lat}_Q] \cos[\text{long}_Q]$$

$$y_Q = \cos[\text{lat}_Q] \sin[\text{long}_Q]$$

$$z_Q = \sin[\text{lat}_Q]$$

where positive longitude is measured east of the Prime Meridian, west longitudes and south latitudes are negative numbers (Fig. 2.1).

Rather than explicitly refer to the axes in the conventional way, using the letters “x, y and z,” we sometimes use the numbers 1, 2 and 3 to indicate the x, y and z axes, respectively. For example, the coordinates of a location vector to a point Q might be represented as $\{x_Q, y_Q, z_Q\}$ or equivalently as $\{Q_1, Q_2, Q_3\}$.

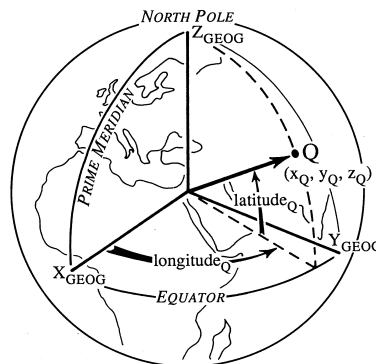


Figure 2.1. The geographic–Cartesian coordinate system. The Z_{GEOG} axis passes through the geographic North Pole, the X_{GEOG} axis contains the intersection of the Prime Meridian and the Equator, and the Y_{GEOG} axis is normal to the Prime Meridian. Positive longitude is measured east of the Prime Meridian; positive latitude is measured north of the Equator. The Cartesian coordinates of a point Q , whose longitude and latitude are known, are $\{x_Q, y_Q, z_Q\}$. From Cronin (1991).

Values for latitude and longitude that we use in our codes are expressed in decimal degrees. Sometimes, geographic coordinates are given in degrees, minutes and seconds, or in degrees and decimal minutes. There are 60 seconds of arc (60”) per minute of arc, and 60 minutes of arc (60’) per degree (°). We can thank the ancient Babylonians for this convoluted method of describing angles. For those of us who are not ancient Babylonians, measurement in decimal degrees is a modest improvement and radian measure is a significant

improvement. If a conversion was necessary between the angle $a^\circ b' c''$ and decimal degrees, the solution would be given by solving the following: $\text{angle in decimal degress} = a + (b/60) + (c/3600)$.

If we would like to determine the unit position vector to a point at latitude 30.42°N and 83.56°W , we begin by using the *Mathematica* input expressions that follow.

```
latQ = 30.42;
longQ = 83.56;
```

The arbitrary variable name **latQ** is used, and is associated with the value 30.42 (as in 30.42°). The definition line is ended with a semicolon. The semicolon has the effect of suppressing the immediate printing of an output line; that is, without the semicolon, an output line would follow the definition, and would reiterate the value assigned to variable **latQ**.

```
latQ = 30.42
30.42
```

When we define a new variable name in *Mathematica*, we must use a small-case letter to begin the variable name. Names that begin with capital letters are reserved for use by functions that are built-in to the *Mathematica* language, such as **Cos**, **Degree**, **Cross** and **VectorAngle**. The *Mathematica* language includes a large number of built-in functions, including all of the standard trigonometric functions. The entities that the function operates on (that is, the *arguments* of the function) are contained within square brackets [].

Trigonometric functions in *Mathematica* operate on angles that are expressed in radian measure. As the name implies, radian measure is based on the radius of a circle. The circumference of a circle with radius r is equal to $(2 r \pi)$; that is, the circumference is equal to the length of 2π radii. It follows that the full arc of a circle is (2π) radians or 360° . Hence, to convert an angle expressed in radians to degrees, multiply the angle by $(180^\circ/\pi \text{ radians})$, and to convert an angle expressed in degrees to radians, multiply the angle by $(\pi \text{ radians}/180^\circ)$. We can also use the built-in function **Degree** to make the conversion from degrees to radian measure.

Using the input data supplied above, we can define variables **xQ**, **yQ** and **zQ** as the Cartesian coordinates of a unit location vector (also known as a position vector) that corresponds to the input latitude and longitude.

```
xQ = Cos [latQ Degree] Cos [longQ Degree] ;
yQ = Cos [latQ Degree] Sin [longQ Degree] ;
zQ = Sin [latQ Degree] ;
```

Simple multiplication of one number by another number can be achieved in *Mathematica* by using the ***** symbol, as in Excel and some other programming languages, or by simply inserting a space between the two values or variable names to be multiplied.

If we want to output the numerical value of each of the coordinates, we can use the built-in *Mathematica* function **N** as follows, being careful *not* to include a semicolon at the end of each phrase so that the output will be displayed:

```
N[xQ]
0.096722
N[yQ]
0.856896
N[zQ]
0.506335
```

Redefining and clearing variables

Sometimes it seems desirable to change the value or equation associated with a given variable. Imagine we have a variable named **edgar** that we defined early in a program to have the following value.

```
edgar = 47;
```

and so every time we use the variable named **edgar** in an expression, the value 47 takes the place of **edgar**.

```
edgar + 0
47
```

Later in the same notebook, we might need to re-define **edgar** with a different value

```
edgar = 4 π;
```

This will take the place of the earlier definition of **edgar**, so when we ask what value is now associated with the variable name **edgar**, we obtain the new value.

```
edgar
```

```
4 π
```

Mathematica returned the symbolic answer. If you want the numerical value of **edgar**, use the built-in function **N** as follows:

```
N[edgar]
```

```
12.5664
```

If you want to be exceptionally careful, or if you want to delete the values associated with a variable in order to save memory, you can use the built-in *Mathematica* command **Clear**.

```
Clear[edgar];
```

Now, the name **edgar** has no other meaning than a set of letters. It is no longer a variable name.

```
edgar
```

```
edgar
```

```
N[edgar]
```

```
edgar
```

Exercise 2.1 Use Google Earth to find the latitude and longitude of some place that is of interest to you, using decimal degrees. Create a *Mathematica* notebook with headers for its title, introduction, input data, computation, output, and references (if any). Save the notebook using your last name as the first part of the notebook name, then the homework number. For example, “Jackson-HW01”. Add text to the notebook providing your name, the date you began and completed the notebook, a statement of the problem, and relevant explanations of the variables and code. Finally, add input lines of executable code. The purpose of this notebook is to compute the unit location vector to the point that is of interest to you.

2.3 Determining vector length

It is common for the norm or length of a vector **a** to be indicated by single straight brackets, as in $|\mathbf{a}|$. Curly brackets $\{\}$ are used in *Mathematica* to delineate sets of values or variables, as might be used to specify a vector's components. The length of an arbitrary vector **a** with coordinates $\{a_1, a_2, a_3\}$ is given by

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

We can define a vector **a** by equating **a** with a list of three components within curly brackets and separated by commas, as in

```
a = {3, 4, 5};
```

There are several ways to create the *Mathematica* code to find the length of vector **a**. The Basic Math Assistant palette can be used to write the *Mathematica* expression as

```
aLength1 = Sqrt[a[[1]]^2 + a[[2]]^2 + a[[3]]^2];
```

or the expression can be written to make use of the built-in *Mathematica* function **Sqrt** as follows

```
aLength2 = Sqrt[a[[1]]^2 + a[[2]]^2 + a[[3]]^2];
```

in which the caret symbol (^) indicates that the preceding number is to be raised to the power indicated by the number that follows. So (x^2) is $(x \cdot x)$ or, in words, x squared. The equation could have also been written more explicitly using parentheses as

```
aLength3 = Sqrt[(a[[1]]^2) + (a[[2]]^2) + (a[[3]]^2)];
```

with no change in the result. Whether an individual uses extra parentheses or not is a matter of style and does not affect the speed with which the expression is evaluated. Finally, the length of vector **a** can also be determined using the built-in *Mathematica* function **Norm** as follows

```
aLength4 = Norm[a];
```

Lets compare the results of the four expressions for determining the length of vector **a**:

```
N[aLength1]
```

```
7.07107
```

```
N[aLength2]
```

```
7.07107
```

```
N[aLength3]
```

```
7.07107
```

```
N[aLength4]
```

```
7.07107
```

As demonstrated above, the four expressions for the length of vector **a** are equivalent. We will generally use the built-in function **Norm** to define the length of a vector.

2.4 Defining unit vectors

A unit vector is a vector that has a length of 1. The unit vector associated with some arbitrary vector **a** has the same orientation as the original vector **a**, but the unit vector has a length of 1. We will sometimes represent a unit vector using the “hat” on top of the vector name (**$\hat{\mathbf{a}}$**). Each component of the unit vector is determined by dividing the original vector’s component by the length of the original vector (**|a|**).

$$\mathbf{a} = \{a_1, a_2, a_3\}$$

$$\hat{\mathbf{a}} = \left\{ \frac{a_1}{|\mathbf{a}|}, \frac{a_2}{|\mathbf{a}|}, \frac{a_3}{|\mathbf{a}|} \right\}$$

We can write a *Mathematica* expression to define the unit vector that coincides with vector **a**

```
unitVectorA = {a[[1]] / Norm[a], a[[2]] / Norm[a], a[[3]] / Norm[a]};
```

where **a[[1]]** is the first element of the vector **a** (that is, coordinate a_1), **a[[2]]** is the second element of **a**, and **a[[3]]** is the third element of **a**.

The numerical result is

```
N[unitVectorA]
```

```
{0.424264, 0.565685, 0.707107}
```

We can check to be sure that the length of **unitVectorA** is equal to 1

```
Norm[unitVectorA]
```

```
1
```

A user-defined function

If the computation of a unit vector is a task that is repeated several times in a notebook, it might be easier to define a function or subroutine like **unitVector3D** that will act on any arbitrary vector and find the corresponding coordinates to the unit vector.

```
unitVector3D[vect_] :=  
{vect[[1]] / Norm[vect], vect[[2]] / Norm[vect], vect[[3]] / Norm[vect]};
```

The text **unitVector3D** is the name of the user-defined function that determines the unit vector of any vector of arbitrary length in a 3-dimensional reference frame. The text **vect_** is enclosed in square brackets and is the argument of the user-defined function -- the name of the vector that the function operates on. The variable name **vect_** is a place-holder that is automatically replaced by the variable name that is supplied by the user when the function **unitVector3D** is used. The underscore line at the end of **vect_** is an essential part of the variable name that identifies it, in context, as a place-holder. The symbol **:=** begins the definition of the function, and the semicolon (**;**) ends that definition. This particular function generates three numbers enclosed in curly brackets and separated from one another using commas. We interpret this output to be the three components of the unit vector associated with the arbitrary vector that the user has supplied as the argument of the function.

Testing this user-defined function using the vector **a**,

```
unitVector3D[a]
```

$$\left\{ \frac{3}{5\sqrt{2}}, \frac{2\sqrt{2}}{5}, \frac{1}{\sqrt{2}} \right\}$$

The numerical components of vector **a** are

```
N[unitVector3D[a]]
```

```
{0.424264, 0.565685, 0.707107}
```

The length or norm of vector **a** is

```
N[Norm[unitVector3D[a]]]
```

```
1.
```

2.5 Some details about inputting and displaying vector coordinates

In the geographic–Cartesian system, the positive Z_{GEOG} axis passes through Earth's north pole, and the intersection of the Prime Meridian and the Equator marks the positive X_{GEOG} axis (Figure 2.1). The Cartesian coordinates we have defined can be represented in a 3x1 matrix that defines a unit position vector **Q** to the point *Q*

$$\mathbf{Q} = \begin{pmatrix} x_Q \\ y_Q \\ z_Q \end{pmatrix}$$

We will use the component values of position vector **Q** that we already computed in section 2.2. Coding this in *Mathematica* is easy, and can be done in several ways. We will refer to the vector using variable name **vectorQ**, and show that the different ways of defining the vector give equivalent results. The first way uses a built-in template for creating matrices, available from the Basic Math Assistant palette in *Mathematica*. Look for the **Typesetting** part of the **Basic Math Assistant** palette, and find the symbol that looks like this:

$$\begin{pmatrix} \square \\ \square \end{pmatrix}$$

Extra rows are created by inserting the cursor inside the parentheses in the default 2x1 matrix, the pressing **[CTRL] [RET]** (that is, press the return key while depressing the control key). When you do that once, the matrix template will look like this

$$\begin{pmatrix} \square \\ \square \\ \square \end{pmatrix}$$

and you can click on each individual square to insert a value or variable name in that space.

$$\mathbf{vectorQ1} = \begin{pmatrix} \mathbf{xQ} \\ \mathbf{yQ} \\ \mathbf{zQ} \end{pmatrix};$$

```
N[vectorQ1]
```

```
{0.096722, 0.856896, 0.506335}
```

The second way uses curly brackets to define a matrix that has 3 rows (each of which is defined by one set of curly brackets) and 1 column (implied by the fact that there is only one variable or value enclosed by each of the three sets of curly brackets), all of which are enclosed in a set of curly brackets as follows:

```
vectorQ2 = {{xQ}, {yQ}, {zQ}};
```

```
N[vectorQ2]
```

```
{0.096722, 0.856896, 0.506335}
```

We can use the built-in function **MatrixForm** to output the matrix in a more familiar manner

```
MatrixForm[N[vectorQ2]]
```

$$\begin{pmatrix} 0.096722 \\ 0.856896 \\ 0.506335 \end{pmatrix}$$

We can also use the custom function **squareMatrixForm** that is defined below to output the matrix using square brackets, as it might appear in a textbook on matrix mathematics. This function was written by programmers at Wolfram Research. Notice that this user-defined function makes use of several functions that are built-in to the *Mathematica* language. The built-in functions all begin with a capital letter, and include **CellPrint**, **Cell**, **ToBoxes**, **StyleBox**, **SpanMaxSize**, and **Infinity**. You can investigate what these functions do by choosing Wolfram Documentation from the *Mathematica* Help menu.

```
squareMatrixForm[mat_] := CellPrint[
  Cell[ToBoxes[MatrixForm@mat] /. {"(" -> StyleBox["[", SpanMaxSize -> Infinity],
    ")" -> StyleBox["]", SpanMaxSize -> Infinity]}], "Output"]]
```

```
squareMatrixForm[N[vectorQ2]]
```

$$\begin{bmatrix} 0.096722 \\ 0.856896 \\ 0.506335 \end{bmatrix}$$

We will leave the details of the user-defined function **squareMatrixForm** for another time; however, that function has to be defined in any *Mathematica* notebook in which it is used. Built-in functions do not need to be defined separately in every notebook they are used in, because they are (as the name implies) built-in to the *Mathematica* application.

2.6 Determining the angle between two vectors

There are several ways of computing the angle between two vectors **a** and **b**. First, we define the coordinates of the two vectors.

```
a = {3, 4, 5};
```

```
b = {2, 5, 8};
```

We can use the function **unitVector3D** that we defined in section 3.4 to compute the unit vectors of **a** and **b**. The corresponding unit vectors are given by

```
aUnit = unitVector3D[a];
```

```
bUnit = unitVector3D[b];
```

The simplest way is to use the built-in *Mathematica* function **VectorAngle**, which outputs the angle in radian measure.

```
angleAB1 = VectorAngle[a, b];
```

The answer, in radian measure, is

```
N[angleAB1]
```

```
0.254175
```

We could have generated the answer in degrees by using the conversion factor ($180^\circ/\pi$ radians)

```
N[angleAB1 (180 / \pi)]
```

```
14.5632
```

Vector dot product or scalar product

The more conventional manner of determining the angle between two vectors is to compute the dot product (also known as the scalar product) of the two vectors. The dot product of two non-collinear vectors of arbitrary length (**a** = { a_1, a_2, a_3 } and **b** = { b_1, b_2, b_3 }) is given by

$$\mathbf{a} \cdot \mathbf{b} = (a_1 b_1) + (a_2 b_2) + (a_3 b_3) = |\mathbf{a}| |\mathbf{b}| \cos[\theta]$$

where θ is the angle between vectors **a** and **b**. This can be rearranged to isolate θ as follows

$$\theta = \cos^{-1} \left[\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} \right]$$

The dot product of two vectors **a** and **b** is implemented in *Mathematica* as **Dot[a,b]**, or simply by using a period between the **a** and **b**: **a.b**. Does the order in which the variables are listed between the square brackets matter? That is, does **a.b = b.a**? Let's check.

```
result1 = a.b;
```

```
result2 = b.a;
```

```
N[result1]
```

```
66.
```

```
N[result2]
```

```
66.
```

So the dot product is a commutative function for which the order of the arguments does not matter. If the vectors are not necessarily unit vectors, the angle between them is

```
angleAB2 = ArcCos[Dot[a, b] / (Norm[a] Norm[b])];
```

If the two vectors are unit vectors, the angle between the two vectors is given by

```
angleAB3 = ArcCos[Dot[aUnit, bUnit]];
```

or

```
angleAB4 = ArcCos[aUnit.bUnit];
```

Comparing the four results,

```
N[angleAB1]
```

```
0.254175
```

```
N[angleAB2]
```

```
0.254175
```

```
N[angleAB3]
```

```
0.254175
```

```
N[angleAB4]
```

```
0.254175
```

2.7 Vector cross product

We will work with two vectors, **a** and **b**, that we redefine as follows.

```
a = {0.6, 0, 0};
```

```
b = {0, 1.4, 0};
```

Vector **a** is coincident with the positive “x” axis of a Cartesian coordinate system, and **b** coincides with the positive “y” axis. The result of the cross product of vectors **a** and **b** is a “pseudovector” we shall call **c** that is oriented perpendicular to the plane that contains vectors **a** and **b**. That is, **c** is perpendicular to **a**, and **c** is also perpendicular to **b**. The components of a vector **c** defined by cross product **a × b = c** is given by

$$\mathbf{a} \times \mathbf{b} = \mathbf{c} = \{(a_2 b_3) - (a_3 b_2), (a_3 b_1) - (a_1 b_3), (a_1 b_2) - (a_2 b_1)\}$$

or

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} (a_2 b_3) - (a_3 b_2) \\ (a_3 b_1) - (a_1 b_3) \\ (a_1 b_2) - (a_2 b_1) \end{pmatrix}$$

If you are curious about what a *pseudovector* is, I suggest that you consult Wikipedia or <http://mathworld.wolfram.com/Pseudovector.html>. With apologies to any physicists or mathematicians who might be reading this, we will refer to pseudovectors as simply *vectors* unless there is a buring reason to do otherwise.

Example. Manually compute the vector cross product $\mathbf{a} \times \mathbf{b} = \mathbf{c}$ given the vector components specified above.

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1.4 \\ 0 \end{pmatrix} = \begin{pmatrix} (0 \cdot 0) - (0 \cdot 1.4) \\ (0 \cdot 0) - (0.6 \cdot 0) \\ (0.6 \cdot 1.4) - (0 \cdot 0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0.84 \end{pmatrix}$$

So vector \mathbf{c} , expressed to the appropriate number of decimal places, has components $\{0, 0, 0.8\}$.

This vector cross product can also be written as a determinant, in which $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are unit vectors along the coordinate axes:

$$\mathbf{a} \times \mathbf{b} = \det \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

One way to compute the cross product between two vectors is to adapt the formulae from the previous discussion.

```
result1 = {(a[[2]] b[[3]]) - (a[[3]] b[[2]]),
           (a[[3]] b[[1]]) - (a[[1]] b[[3]]), (a[[1]] b[[2]]) - (a[[2]] b[[1]])};
```

Another way is to use the \times symbol that is available in the Typesetting part of the Basic Math Assistant palette in *Mathematica*.

```
result2 = a × b;
```

Perhaps the simplest and least ambiguous way to compute the cross product between two vectors is to use the built-in *Mathematica* function **Cross**.

```
result3 = Cross[a, b];
```

Let's check to see if these three approaches give the same results.

```
N[result1]
{0., 0., 0.84}

N[result2]
{0., 0., 0.84}

N[result3]
{0., 0., 0.84}
```

Because these approaches yield the same results, we will usually choose to use one of the easier methods, either **Cross[a, b]** or the \times symbol as in $\mathbf{a} \times \mathbf{b}$.

Does the order in which the arguments are provided matter? Is a vector cross product a commutative function? Does $\mathbf{a} \times \mathbf{b} = \mathbf{b} \times \mathbf{a}$? Let's check.

```
firstResult = Cross[a, b];
secondResult = Cross[b, a];

N[firstResult]
{0., 0., 0.84}

N[secondResult]
{0., 0., -0.84}
```

We have shown that the vector cross product is not a commutative function, so the order of the arguments determines what the answer will be. That is, $\mathbf{a} \times \mathbf{b} \neq \mathbf{b} \times \mathbf{a}$. However, $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$, and hence the result of $\mathbf{a} \times \mathbf{b}$ is a vector that has the same length but the opposite direction as the result of $\mathbf{b} \times \mathbf{a}$.

Understanding the direction of the vector result of a cross product is important. We start our visualization with non-colinear vectors \mathbf{a} and \mathbf{b} , which have a common origin and are part of the \mathbf{a} - \mathbf{b} plane (Fig. 2.2). Given the equation $\mathbf{a} \times \mathbf{b} = \mathbf{c}$, we want to understand the orientation of resultant vector \mathbf{c} . Imagine pointing the index finger of your right hand along vector \mathbf{a} , then relaxing your index finger so that it curls toward \mathbf{b} , taking the most direct route to \mathbf{b} in the \mathbf{a} - \mathbf{b} plane. (That is, in the direction that the angle between \mathbf{a} and \mathbf{b} is less than 180° .) Then your outstretched right thumb will point (approximately) perpendicular to the \mathbf{a} - \mathbf{b} plane, coincident with vector \mathbf{c} . That is, if you start at vector \mathbf{a} , vector \mathbf{b} is a positive (counter-clockwise) rotation of less than 180° around vector \mathbf{c} .

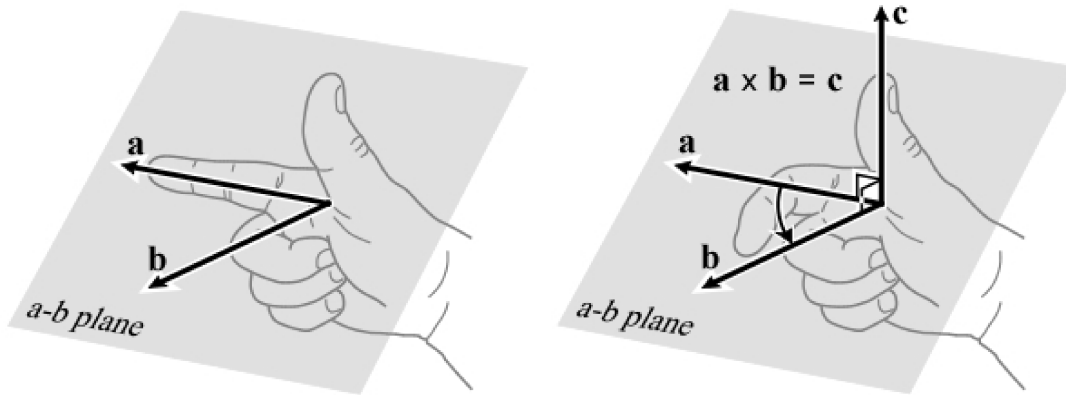


Figure 2.2. Visualization of the geometry of a vector cross product. The result of the vector cross product $\mathbf{a} \times \mathbf{b}$ is pseudovector \mathbf{c} , which is oriented perpendicular to the \mathbf{a} - \mathbf{b} plane. The direction of \mathbf{c} is given by the right-hand rule: the thumb of the right hand points in the direction of \mathbf{c} , where vector \mathbf{b} is an anticlockwise rotation of $<180^\circ$ from \mathbf{a} around their common origin.

Recall that, as we initially defined them at the start of this section, vector $\mathbf{a} = \{0.6, 0, 0\}$ is oriented along the positive x axis (which we know because the components have the form $\{n, 0, 0\}$, where n is any non-zero real number), vector $\mathbf{b} = \{0, 1.4, 0\}$ was along the positive y axis (because the components have the form $\{0, n, 0\}$), and the result of $\mathbf{a} \times \mathbf{b}$ is a vector $\mathbf{c} = \{0, 0, 0.8\}$ that is along the positive z axis. Those three vectors are mutually perpendicular. Let's change the components of \mathbf{a} and \mathbf{b} a little bit so they are not perpendicular to each other, keeping them on the x - y plane (that is, giving both a z coordinate of zero), and see if the result of a vector cross product is still a vector that coincides with the z axis.

```
a = {0.9, 0.1, 0};
b = {0.2, 0.7, 0};
c = Cross[a, b];
N[c]
{0., 0., 0.61}
```

Indeed, \mathbf{c} is a vector that is parallel to the z axis, which is clear because any vector that coincides with the z axis will have components $\{0, 0, n\}$.

Exercise 2.2 You are given a vector \mathbf{a} with components $\{4, 2, 7\}$ and a vector $\mathbf{b} = \{-2, 5, -4\}$. Write a *Mathematica* notebook that analyzes the input data to complete the following tasks.

- Compute the vector cross product $\mathbf{a} \times \mathbf{b}$ to yield the vector \mathbf{c} , and list the components of vector \mathbf{c}
- Determine the length of vector \mathbf{c}
- Determine the angle between vectors \mathbf{a} and \mathbf{c}
- Determine the angle between vectors \mathbf{b} and \mathbf{c}
- What can you say about the orientation of vector \mathbf{c} relative to vectors \mathbf{a} and \mathbf{b} that will be generally true given any such non-zero, non-colinear vectors \mathbf{a} and \mathbf{b} ?

We know how to determine the length of a vector (see section 2.3). The length or norm of the vector result of a vector cross product $\mathbf{a} \times \mathbf{b}$ is given by

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}| |\mathbf{b}| \sin[\theta]$$

where $|\mathbf{a}|$ is the norm or length of vector \mathbf{a} , $|\mathbf{b}|$ is the norm of \mathbf{b} , and θ is the angle between vectors \mathbf{a} and \mathbf{b} given by the dot product

$$\cos[\theta] = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}}$$

$$\theta = \cos^{-1}[\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}]$$

in which $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ are unit vectors that coincide with arbitrary vectors \mathbf{a} and \mathbf{b} , respectively. So let's test these mathematical expressions, using some simple vectors \mathbf{a} and \mathbf{b} that we redefine as follows.

```
a = {0.9, 0.1, 0};
```

```

b = {0.2, 0.7, 0};
testLength1 = Norm[Cross[a, b]];
testLength2 = Norm[a] Norm[b] Sin[ArcCos[Dot[unitVector3D[a], unitVector3D[b]]]];
N[testLength1]
0.61
N[testLength2]
0.61

```

Note that vectors **a** and **b** are both on the x-y plane of our Cartesian coordinate system, so we already know that the (pseudo)vector result of $\mathbf{a} \times \mathbf{b}$ will be along the z axis.

```

c = Cross[a, b];
N[c]
{0., 0., 0.61}

```

The value of the z component of vector **c** in this example is the same as the length of the vector, which is consistent with the results obtained above.

2.8 Determining the azimuth from one location vector to another

Another common task is to determine the compass direction from one location to another, as well as the distance along the most direct path across Earth's surface between the two points. Azimuth is an angle that is usually expressed in degrees relative to north (0°) and that increases in a clockwise direction so that east is 90° , south is 180° , and west is 270° . The most direct path between two points on the surface of a sphere is called the great-circle distance, because the path is along part of a great-circle or circumferential arc.

For any two non-zero vectors that are not colinear, a unique plane can be identified that contains those two vectors. We say that the two vectors *define* the plane. The location vectors to two points *N* and *Q* define a plane through their common origin at Earth's center, and that plane traces a circumferential or great-circle arc through *N* and *Q* on Earth's surface (Fig. 2.3). If we identify the location vectors through points *N* and *Q* as vectors **n** and **q**, respectively, the vectors normal to the *N-Q* plane are the results of the vector cross products

$$\mathbf{normalNQ} = \mathbf{n} \times \mathbf{q}$$

$$\mathbf{normalQN} = \mathbf{q} \times \mathbf{n}$$

where

$$\mathbf{normalNQ} = -\mathbf{normalQN}$$

That is, these two normal vectors are colinear, but point in opposite directions.

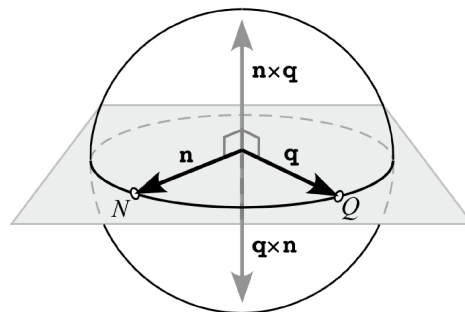


Figure 2.3. The *N-Q* plane (gray) passes through Earth's center and traces a great-circle (circumferential) arc on Earth's surface. Vectors normal to the *N-Q* plane are found by taking the cross product of the unit location vectors (**n** and **q**) to points *N* and *Q*.

Now let's imagine adding a point *P* on Earth's surface, such that *P* is *not* located along the *N-Q* great circle. The location vectors to points *N* and *P* (vectors **n** and **p**, respectively) define a plane we will call the *N-P* plane, whose normal vectors are

$$\text{normalPN} = \mathbf{p} \times \mathbf{n}$$

$$\text{normalNP} = \mathbf{n} \times \mathbf{p}$$

and the location vectors to points P and Q (vectors \mathbf{p} and \mathbf{q} , respectively) define the P - Q plane, whose normal vectors are

$$\text{normalPQ} = \mathbf{p} \times \mathbf{q}$$

$$\text{normalQP} = \mathbf{q} \times \mathbf{p}$$

We would like to know the dihedral angles between the P - Q and N - P planes (Fig. 2.4). The dihedral angles are supplementary angles to each other. For example, 30° and 150° are supplementary angles because they sum to 180° . In Figure 2.4, the supplementary angles between the P - Q and N - P planes are θ° and $180^\circ - \theta^\circ$.

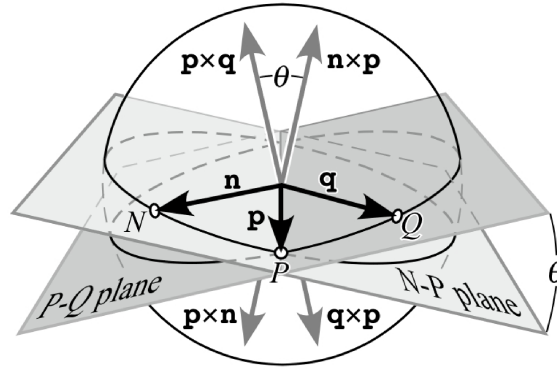


Figure 2.4. blah blah blah.

If we had located point P along the N - Q great circle, the P - Q and N - P planes would coincide -- remember that both share a common origin through Earth's center -- and so their respective normal vectors would be colinear. The dihedral angle between the planes, as well as the angle between the respective normal vectors, would be $0/180^\circ$. If, on the other hand, we imagined prying the two planes apart so that their supplementary dihedral angles were θ° and $(180^\circ - \theta^\circ)$, the normal vectors would also diverge from each other by θ° or $(180^\circ - \theta^\circ)$. By measuring the angle between a vector normal to one plane and a vector normal to another plane, we are measuring one of the two supplementary dihedral angles between the two planes.

Now let's think about a different trio of points and specify that the point N coincides with the north pole. The great circle from P to N is a meridian that we can use to measure an azimuth from the perspective of point P . In Figure 2.5, point Q is located to the east of point P . In this case, the azimuth of Q relative to P (**azimuthQrelP**) is the same as the angle between the normal vector

$$\text{normalPN} = \mathbf{p} \times \mathbf{n}$$

and

$$\text{normalPQ} = \mathbf{p} \times \mathbf{q}$$

$$\text{azimuthQrelP} = \text{VectorAngle}[\text{normalPN}, \text{normalPQ}] (180/\pi)$$

Had Q been located to the west of P , the azimuth of Q from P would have been

$$\text{azimuthQrelP} = 360 - \text{VectorAngle}[\text{normalPN}, \text{normalPQ}] (180/\pi)$$

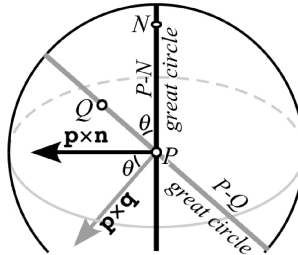


Figure 2.5. View looking along the line of intersection between the P - N plane and the P - Q plane, which is the best view for discerning the dihedral angle θ between those two planes. Point N is the north pole. The intersection of those planes and the surface of Earth are great circles. The vector normal to the P - N plane is defined by $\mathbf{p} \times \mathbf{n}$, and the vector normal to the P - Q plane is defined by $\mathbf{p} \times \mathbf{q}$, where \mathbf{p} and \mathbf{q} are the location vectors to points P and Q , respectively. The azimuth from P to Q is $360^\circ - \theta^\circ$ in this example because point Q is west of point P .

Problem

Given the geographic coordinates of arbitrary points P and Q located on the surface of an assumed-spherical Earth, (1) what the distance between P and Q expressed as an angle and as a circumferential or great-circle distance in kilometers, and (2) what is the azimuth from P to Q , and from Q to P ?

Input data

```
latP = 32.56;
longP = -119.28;
latQ = 34.25;
longQ = -120.34;
```

Given those input data, we begin the computational phase of our work by converting the geographic coordinates to 3-D Cartesian coordinates as specified in Figure 2.1.

Computation

```
p = {Cos[latP Degree] Cos[longP Degree],
     Cos[latP Degree] Sin[longP Degree], Sin[latP Degree]};
q = {Cos[latQ Degree] Cos[longQ Degree],
     Cos[latQ Degree] Sin[longQ Degree], Sin[latQ Degree]};
```

We define a unit location vector to the geographic north pole of Earth.

```
n = {0, 0, 1};
```

Now the fun begins. Unit location vectors \mathbf{p} and \mathbf{n} have a common origin at the center of Earth, and so they define a plane through the center of Earth. The vector normal to the \mathbf{p} - \mathbf{n} plane is defined by $\mathbf{p} \times \mathbf{n}$, and is called **normalPN**

```
normalPN = Cross[p, n];
```

Similarly, unit location vectors \mathbf{p} and \mathbf{q} have a common origin at the center of Earth, and so they define a plane through the center of Earth. The vector normal to that plane is defined by $\mathbf{p} \times \mathbf{q}$, and is called **normalPQ**

```
normalPQ = Cross[p, q];
```

The \mathbf{p} - \mathbf{q} plane intersects the \mathbf{p} - \mathbf{n} plane at point P . The dihedral angle between the \mathbf{p} - \mathbf{q} plane and the \mathbf{p} - \mathbf{n} plane at point P is the same as the angle between a line pointing north from P and a line pointing toward Q from P . A simple way to determine the dihedral angle between those two planes is to compute the angle between the vector normal to each plane: **normalPN** and **normalPQ**. Note that **normalPN** and **normalPQ** are not necessarily unit vectors, so the technique used to find the angle between them must be valid for vectors of any length. The angle between a line pointing north from P and a line pointing toward Q from P is called **theta1**

```
theta1 = VectorAngle[normalPN, normalPQ] * (180 /  $\pi$ );
```

Now we repeat the process to find the angle between a line pointing north from Q and a line pointing toward P from Q , so that we might be able to work-out the azimuth of P as observed from Q .

```
normalQN = Cross[q, n];
```

```
normalQP = Cross[q, p];
```

```
theta2 = VectorAngle[normalQN, normalQP] * (180 /  $\pi$ );
```

The angle between the location vector to P and the location vector to Q is called the *angular distance* between P and Q . The angular distance from P to Q is **theta3** in radian measure, as given by

```
theta3 = VectorAngle[p, q];
```

The formula for the circumference of a sphere or a circle is $2\pi r$ where r is the radius. There are 2π radians in a full circle. The angle **theta3** is expressed in radians. The fraction of a full circle represented by **theta3** is given by **theta3**/(2π). If the full circumference is $2\pi r$, the part of the circumference that corresponds to an angular distance of **theta3** must be

$$(2\pi r) \left(\frac{\text{theta3}}{2\pi} \right) = r \text{ theta3}$$

Earth isn't a sphere, or a spheroid, or an ellipsoid, or even a geoid in detail. However, it is reasonable to the first order to model it as if it is a sphere -- an assumption that makes the mathematics of plate kinematics a great deal simpler. Earth's flattening is currently understood to be 1 part in 298.257 (Yoder, 1995). Let's see what that would mean with respect to an accurate representation of Earth's reference ellipsoid if the equatorial radius was 10 cm.

```
flattening = 1 / 298.257;
```

```
a = 10;
```

```
b = a - (a flattening);
```

```
N[b]
```

```
9.96647
```

If we drew an Earth with an equatorial radius of 10 cm, the polar radius would be 9.97 cm or about a third of a millimeter less than the equatorial radius. If you used a drafting compass to draw a 10-cm-radius circle with an 0.5 mm thick line, the entire ellipse would be within the black line of the circle. So the spherical-Earth assumption is not only mathematically advantageous, it is a spatially reasonable simplification.

Information about the size and shape of the Earth can be obtained from Ahrens (1995) and Herring (2009), from which we adopt a mean radius for an assumed-spherical Earth of 6371.01 ± 0.02 km.

```
earthMeanRadius = 6371.01;
```

The great-circle or circumferential distance from P to Q is given by **circumfDistP2Q**.

```
circumfDistP2Q = (earthMeanRadius) (theta3);
```

All of the results obtained thus far have been suppressed by ending all of the input lines with semicolons. Now that the computational work is done, we need to display the numerical solutions of our results.

Output data

Variable **theta1** is the angle between a line pointing north from P and a line pointing toward Q from P , expressed in degrees.

```
N[theta1]
```

```
27.3463
```

Variable **theta2** is the angle between a line pointing north from Q and a line pointing toward P from Q , expressed in degrees.

```
N[theta2]
```

```
152.07
```

Variable **circumfDistP2Q** is the great-circle distance between points P and Q , expressed in kilometers.

```
N[circumfDistP2Q]
```

```
212.117
```

Given the preceding angles expressed in degrees, what are the corresponding azimuths?

Azimuth from P to Q: _____ Azimuth from Q to P: _____

Can we modify the process so that the code gives us the azimuth between any two points directly, and not just the angle between the north-directed line and the line to the other point? If so, how?

Exercise 2.3 In the early Miocene ~23 million years ago, a volcano erupted in California. Sometime later in the Miocene, the San Andreas fault propagated through the volcanic field, and separated it into what is now the Pinnacles National Monument ($36^{\circ}29'13''\text{N}$, $121^{\circ}10'01''\text{W}$) on the west side of the fault and the Neenach volcanic field ($34^{\circ}44'24''\text{N}$, $118^{\circ}37'24''\text{W}$) on the east side (Matthews, 1973).

Write a *Mathematica* notebook that analyzes the input data to complete the following tasks.

- Convert the geographic coordinates to unit location vectors, recalling that south latitudes and west longitudes are negative numbers.
- Determine the angular distance between the Pinnacles and Neenach.
- Find the circumferential distance between the Pinnacles and Neenach, assuming that Earth is a sphere of radius 6,371.01 km.

- (d) Find the azimuth of Pinnacles as viewed from Neenach, and of Neenach as viewed from Pinnacles.
- (e) What geological statement can you make about displacement along the San Andreas fault?

2.9 References

Ahrens, T.J., editor, 1995, *Global Earth physics -- a handbook of physical constants*: Washington, D.C., American Geophysical Union, AGU Reference Shelf 1, 376 p., ISBN 0-87590-851-9.

Cronin, V.S., 1991, The cycloid relative-motion model and the kinematics of transform faulting, *in* Hilde, T.W.C., and Carlson, R.L., editors, Special issue, Silver Anniversary of Plate Tectonics, *Tectonophysics* v. 187, no. 1-3, p. 215-249. Corrections to publisher errors, 1991, *Tectonophysics*, v. 192, no. 3/4, p. 401.

Matthews, V., III, 1973, Pinnacles-Neenach correlation -- A restriction for models of the origin of the transverse ranges and the big bend in the San Andreas fault: *Geological Society of America Bulletin*, v. 84, p. 683-688.

Herring, T., editor, 2009, *Geodesy*: Amsterdam, Elsevier, *Treatise on Geophysics*, v. 3, 446 p., ISBN 978-0-444-53460-6.

Yoder, C.F., *Astrometric and geodetic properties of Earth and the Solar System*, *in* Ahrens, T.J., editor, 1995, *Global Earth physics -- a handbook of physical constants*: Washington, D.C., American Geophysical Union, AGU Reference Shelf 1, 376 p., ISBN 0-87590-851-9.

Web resources

Rowland, T., and Weisstein, E. W., *Tensor*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Tensor.html>

Weisstein, E.W., *Cross Product*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/CrossProduct.html>

Weisstein, E.W., *Determinant*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Determinant.html>

Weisstein, E.W., *Dot Product*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/DotProduct.html>

Weisstein, E.W., *Kronecker Delta*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/KroneckerDelta.html>

Weisstein, E.W., *Norm*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Norm.html>

Weisstein, E.W., *Pseudovector*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Pseudovector.html>

Weisstein, E.W., *Right-Hand Rule*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Right-HandRule.html>

Weisstein, E.W., *Vector*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/Vector.html>

Weisstein, E.W., *Vector Multiplication*: available online via *MathWorld* -- a Wolfram web resource, and accessed 18 January 2012. <http://mathworld.wolfram.com/VectorMultiplication.html>