2010

Curve Interpolation and Coding Theory

Darren B. Glass
Gettysburg College

Glass, D., Curve Interpolation and Coding Theory. Journal of Undergraduate Mathematics and its Application. 2010. 31(3): 189-196.

Curve Interpolation and Coding Theory

Abstract
Whether it is downloading files from the Internet, having conversations between cell phones, or sending information from a laptop to a printer, we often want to transmit data in situations where we need to worry about interference from other signals that may cause errors in the transmission. The branch of mathematics known as coding theory is dedicated to finding ways to tell when these are errors in transmission and, when possible, how to correct those errors. The goal of coding theory is to build as much redundancy as possible into a message without greatly increasing its length. [excerpt]

Disciplines
Mathematics

Curve Interpolation and Coding Theory

Darren Glass
Mathematics Dept.
Gettysburg College
Gettysburg, PA 17325–1486
dglass@gettysburg.edu

Introduction

Whether it is downloading files from the Internet, having conversations between cell phones, or sending information from a laptop to a printer, we often want to transmit data in situations where we need to worry about interference from other signals that may cause errors in the transmission.

The branch of mathematics known as coding theory is dedicated to finding ways to tell when there are errors in transmission and, when possible, how to correct those errors. The goal of coding theory is to build as much redundancy as possible into a message without greatly increasing its length. Much of coding theory uses deep mathematics to achieve this end, but a surprising amount of work follows from the following fact of Euclidean geometry, which is known by schoolchildren:

Two points determine a unique line.

More accurately, we will use the more sophisticated version of this fact that says:

Fact: *Any n points with different x-coordinates determine a unique $(n-1)$st-degree polynomial $y = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$.*

In the following section, we discuss why this fact is true and mention a couple of approaches to proving it. We then discuss how Reed and Solomon [1960] used it to create a family of error-correcting codes. In our opinion, Reed-Solomon codes are an underappreciated application of mathematics that is both extremely useful and very accessible to any student who has made it through the opening weeks of a linear algebra course.

Polynomial Interpolation

That through two points a line can be drawn is Postulate 1 of Euclid; that the line is unique, we know from analytic geometry applied to Euclidean geometry. But why is it that three points with distinct x-coordinates determine a unique quadratic equation?

We start by sketching one possible answer: In drawing a parabola, you could choose any two x-values to be the zeros. For example, let's choose the values $x_1 = -1$ and $x_2 = 3$. As you can see in **Figure 1**, many different parabolas have these two zeros. In fact, for any constant a, the curve $y = a(x+1)(x-3)$ passes through the two points $(-1, 0)$ and $(3, 0)$.

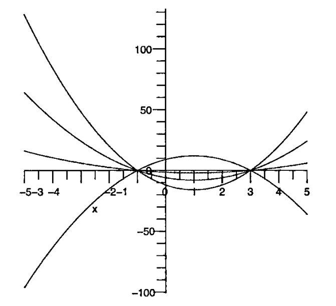

Figure 1. Many parabolas through three specified points.

According to the **Fact** cited above on p. 189, we can choose one more point so as to obtain a unique quadratic equation. Since we did not choose $x = 0$ as either of the zeros, we could choose the y-intercept as the third point. In particular, we could specify that the curve pass through $(0, 3)$, finding then that the unique quadratic equation passing through these three points is $y = -x^2 + 2x + 3$. (As an interesting aside, think about what would happen if we chose $(0, 0)$ as the y-intercept.)

This approach of determining a unique curve by picking points based on their x-coordinates generalizes to situations where we wish to have a parabola that does not have two distinct zeros or where we specify more than three points (and therefore are considering polynomials of higher degree). However, writing down a careful general proof of the **Fact**, using analytic geometry, can be tedious.

A different approach to proving the **Fact** in general, which many students see in a course in linear algebra, is to show that as long as $x_1, \ldots, x_n$ are distinct, the following *Vandermonde matrix* is invertible:

$$\begin{bmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & {x_n}^2 & \cdots & {x_n}^{n-1} \end{bmatrix}$$

Showing that this matrix is invertible is an exercise in many linear algebra texts [Lay 2003, Ch. 2; Shifrin and Adams 2002, Sect. 1.6; Leon 1980, Sect. 1.4]. From the invertibility, it follows that for any choice of $y_1, \ldots, y_n$, the following linear system has a unique solution $a_1, \ldots, a_n$:

$$\begin{bmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & {x_n}^2 & \cdots & {x_n}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

The uniqueness of the a_is yields a unique polynomial

$$P(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

of degree at most $n - 1$ that passes through the desired points (x_i, y_i).

In addition to being useful in curve interpolation, the Vandermonde matrix also shows up in discrete Fourier transforms, representation theory of the symmetric group, and many other places in mathematics.

Finding the a_is can be done quickly via *Lagrange interpolation*, which defines the polynomial by the following formula:

$$P(x) = \sum_{j=1}^{n} y_j \prod_{i \neq j} \frac{x - x_i}{x_j - x_i}.$$

Coding Theory

The goal of coding theory is to build enough redundancy into a message so that errors can be corrected while keeping the message reasonably short.

For example, let's say that you want to transmit the message 8675309 but you are worried that the line is too noisy, so that errors are likely to occur. One naïve approach would be simply to transmit the message *twice*. If the message received is 8679300/8695329, then one can easily tell that errors have occurred because the two transmissions don't exactly match up. For example, we can be sure that there is an error in the third position in (at least) one of the two transmissions. Unfortunately, we have no way of knowing which message is correct in this position.

The slightly less naïve approach of transmitting the message *three times* may allow the recipient to correct errors. Assume that the message received is 8679300/8695329/8775309. Decode by majority rule: If two or all three of the transmissions agree in a position, assume that they are correct, since the odds of the same error occurring in the same place twice will be small.

You have probably hit on the next generalization: If you want to be surer that the message gets through correctly, repeat it more times! This will certainly work, but at the cost of increasing the transmission length, taking up more bandwidth (or cellphone minutes).

However, there are many other approaches to error correction; many use sophisticated mathematics in complicated ways. An interested beginner might start investigating in Gallian [1993] or Roman [1997].

Reed-Solomon Codes

Reed-Solomon codes require only simple polynomial interpolation to correct errors more efficiently than the naïve approach above.

Example. Assume that we want to communicate the message $(1, -2, -1)$. We first encode this message as the coefficients of a polynomial: $f(x) = x^2 - 2x - 1$. Next, we compute the value of this polynomial at a set of prescribed points and transmit those values. For our example, we assume that the pre-chosen points are $x = 0, 1, 2, 3, 4$; so we compute $f(0) = -1$, $f(1) = -2$, $f(2) = -1$, $f(3) = 2$, and $f(4) = 7$ and transmit the message $C_f = (-1, -2, -1, 2, 7)$.

If there are no errors, the recipient of the message can use Lagrange interpolation or other methods to discover the unique quadratic through the points $(0, -1)$, $(1, -2)$, $(2, -1)$, $(3, 2)$, and $(4, 7)$ and recover the intended message $(1, -2, -1)$ as its coefficients.

To see the advantage of this method, note what will happen if one of the values in the message is received incorrectly. For example, maybe the message received is $C = (-1, -2, 1, 2, 7)$.

If we plot the resulting five points $(0, -1)$, $(1, -2)$, $(2, 1)$, $(3, 2)$, and $(4, 7)$, we can easily detect that there has been an error in transmission, because the points no longer lie on a parabola (**Figure 2**). Moreover, we can *correct* the error by finding a polynomial that fits as many of the points as possible.

If we fit the first three points to a quadratic polynomial (using, for example, Lagrange interpolation), we get $f_1(x) = 2x^2 - 3x - 1$, which misses the last two points (**Figure 3**).

Similarly, if we fit the middle three points to a quadratic polynomial, we get $f_2(x) = -x^2 + 6x - 7$, which misses the other two points.

However, if we fit the first, second, and fourth points to a quadratic polynomial, we get $f(x) = x^2 - 2x - 1$, which also passes through the fifth point! And this is the best that we can do, since all five points do not lie on a quadratic.

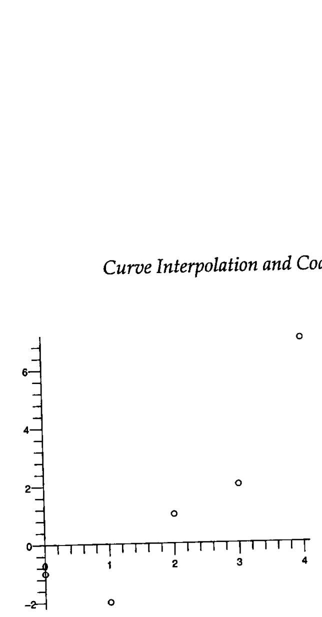

Figure 2. Because no parabola fits, there is an error.

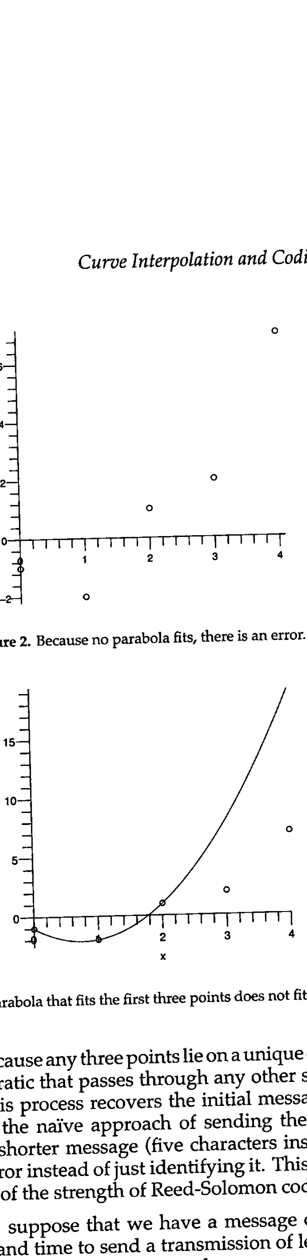

Figure 3. A parabola that fits the first three points does not fit the last two.

Moreover, because any three points lie on a unique quadratic curve, there is no quadratic that passes through any other set of four of the points. Thus, this process recovers the initial message. Comparing this example to the naïve approach of sending the message twice, here we send a shorter message (five characters instead of six) and can correct an error instead of just identifying it. This capability gives some indication of the strength of Reed-Solomon codes.

More generally, suppose that we have a message of length k but the technical capacity and time to send a transmission of length n. The Reed-Solomon approach begins by fixing n numbers $x_1, \ldots, x_n$. An n-tuple is a *k-valid codeword* if it can be generated by evaluating a polynomial of degree strictly less than k at these n predetermined points.

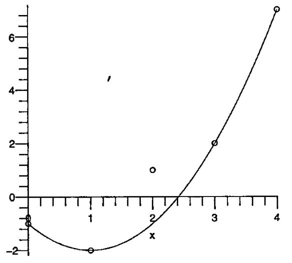

Figure 3. A parabola that fits the first, second, and fourth points also passes through the fifth point—but not through the third.

Example. If $n = 4$, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, and $x_4 = 4$, then

- $(-2, -1, 0, 1)$ is a 2-valid codeword, because it is $\big(f(1), f(2), f(3), f(4)\big)$ where $f(x) = x - 3$ is a linear equation; but
- $(0, 0, 0, 1)$ is not a 2-valid codeword, because there is no linear or constant equation with $f(1) = f(2) = f(3) = 0$ and $f(4) = 1$.

To send our message, we construct a polynomial $g(x)$ of degree $k - 1$. We evaluate the polynomial at n distinct predetermined points $x_1, \ldots, x_n$ and transmit the message

$$\big(g(x_1), \ldots, g(x_n)\big).$$

Because any two k-valid n-tuples agree in at most $k - 1$ coordinates—if two polynomials of degree less than k agree in k points, then they are the same polynomial!—the n-tuples must disagree in at least $n - (k - 1)$ of the n preselected points. Therefore, changing fewer than $n - k + 1$ of the values in the n-tuple will result in an invalid codeword, and *we can thus detect up to $n - k$ errors.*

To see this in an explicit example, try changing any two of the values in the 2-valid 4-tuple $(-2, -1, 0, 1)$—you will see that the resulting 4-tuple is invalid! Moreover, if you change only one of the values, I can correct it by finding the unique line through the other three points (I can do this even if I do not know which point you changed, since only one set of three of the four points will be collinear!). In this manner,

Reed-Solomon codes can correct up to $\lfloor\frac{n-k}{2}\rfloor$ errors.

Hamming Distance

One way to reconceptualize the situation is to view each k-valid n-tuple as a point in n-dimensional space. However, instead of measuring the distance between two points in the normal Euclidean way, we define a new distance measure, the number of coordinates in which they disagree, called the *Hamming distance*, after Richard Hamming, who is often credited as the father of coding theory.

In this conceptualization, two k-valid points obtained from polynomials as on p. 194 must be at least $n - k + 1$ apart in Hamming distance. Therefore, from any given point x, there is at most one k-valid point whose distance is at most $\lfloor\frac{n-k}{2}\rfloor$ from x. Indeed, if there were two k-valid points y and z whose distances were less than or equal to $\lfloor\frac{n-k}{2}\rfloor$, then the triangle inequality would tell us that the distance between y and z was at most $2\lfloor\frac{n-k}{2}\rfloor$, which is strictly less than $n - k + 1$.

In particular, if we start at a k-valid point and make changes in up to $\lfloor\frac{n-k}{2}\rfloor$ positions, our original point will be the only k-valid point within a radius of $\lfloor\frac{n-k}{2}\rfloor$. Therefore, we can correct up to this number of errors by moving to the unique k-valid point that is closest to the received codeword.

Generalizations

This approach can be generalized further to find other sets of valid codewords that work nicely, known as AG-codes, which were first developed by Goppa [1981]. The curve-interpolation approach starts with a polynomial of given degree, defined everywhere on the x-axis and with a single pole of fixed order at $x = \infty$. More-general AG-codes look at functions on other curves with prescribed zeros together with points (poles) where the function is allowed to be undefined but where the pole has a fixed order. The general idea is that by evaluating these functions at n different points on the curve, you get a codeword of length n. There may be other valid codewords, depending on the genus of the curve and the set of prescribed zeros and points where the function is not defined. Depending on the choices of curve and functions, there may also be efficiencies (or a lack thereof) in encoding and decoding. Fully understanding this approach involves learning some algebraic geometry as well as some analysis. An excellent introduction is Walker [2000].

There are other approaches to constructing codes. In fact, many people would say that the *real* goal of coding theory is to define large sets of points in n-dimensional space so that they are efficiently packed, with a large minimum distance but a small total volume. Sphere packing is a long-studied problem with many other applications. For a full discussion, see Pfender and Ziegler [2004].

References

Gallian, Joseph. 1993. How computers can read and correct ID numbers. *Math Horizons* (Winter 1993): 14–15.

Goppa, V. 1981. Codes on algebraic curves. *Doklady Akademii Nauk SSSR* 259 (6): 1289–1290.

Lay, David. 2003. *Linear Algebra and Its Applications.* 3rd ed. Reading, MA: Addison-Wesley.

Leon S. 1980. *Linear Algebra with Applications.* New York: Macmillan.

Pfender, Florian, and Günter M. Ziegler. 2004. Kissing numbers, sphere packings, and some unexpected proofs. *Notices of the American Mathematical Society* 51 (8): 873–883. `http://www.ams.org/notices/200408/fea-pfender.pdf` .

Reed, I., and G. Solomon. 1960. Polynomial codes over certain finite fields. *SIAM Journal* 8 (2): 300–304.

Roman, S. 1997. *Introduction to Coding and Information Theory.* New York: Springer-Verlag.

Shifrin, T., and M. Adams. 2002. *Linear Algebra: A Geometric Approach.* New York: W.H. Freeman.

Walker, J. 2000. *Codes and Curves.* IAS/Park City Mathematical Subseries. Providence, RI: American Mathematical Society, and Princeton, NJ: Institute for Advanced Study.

About the Author

Darren Glass received his B.A. from Rice University, with a double major in mathematics and mathematical economic analysis. He then went to the University of Pennsylvania, where he studied arithmetic geometry and received both an M.A. and a Ph.D. After several years as an NSF-VIGRE postdoc at Columbia University, Glass joined the faculty at Gettysburg College in 2005. His primary research interests are in the fields of number theory and algebraic geometry, with an eye towards applications in cryptography and coding theory. In his spare time, he enjoys spending time with his toddler son, watching baseball, and cooking Mexican food.