Jurnal Yudistira: Publikasi Riset Ilmu Pendidikan dan Bahasa
Volume 3, Nomor 4, Oktober 2025
e-ISSN: 3021-7814; p-ISSN: 3021-7792, Hal. 208-220
DOI: https://doi.org/10.61132/yudistira.v3i4.2392
Tersedia: https://journal.aripi.or.id/index.php/Yudistira

Factorization of Polynomials Over Finite Fields

Rasha Thnoon Taieb Alrawi*
Ministry of Education - Kirkuk Open College of Education, Iraq.
**Corresponding Author*: Rashaalrawi99@gmail.com

Abstract. *This paper discusses basic concepts in finite fields and emphasizes the meaning of irreducible polynomials and their relevance in algebraic analysis. The main focus is directed at the algorithms used to factor polynomials in finite fields through three important stages: distinct degree factorization, square-free factorization, and equal degree factorization. These stages are considered core procedures in determining the structure of polynomials and their relationship to more complex algebraic properties. Furthermore, this paper reviews the role of other algorithms that support this process, such as the Berlekamp algorithm, the Cantor–Zassenhaus algorithm, and several normalization techniques that enhance the effectiveness of the analysis. The combination of these various approaches allows the breakdown of polynomials into simpler factors, while also highlighting how the algorithms work synergistically to achieve accurate analysis results. Thus, this paper emphasizes the importance of a thorough understanding of polynomial factorization algorithms in finite fields, both in theory and application, and their contribution to the development of applied mathematics, particularly in the field of computational algebra.*

Keywords: *Computational algebra; Factorization algorithms; Finite fields; Irreducible polynomials; Number theory.*

1. INTRODUCTION

In math and computer algebra, factoring a polynomial means breaking it down into a product of smaller, unbreakable parts (von zur Gathen & Gerhard, 2013). This is always possible and only happens in one way for polynomials with numbers as coefficients in any field. However, to actually calculate this factorization using a computer program, we need some limits on the numbers used (Geddes, Czapor, & Labahn, 1992). In reality, programs to find these factors have only been made for polynomials where the numbers used are from a limited field, the set of fractions, or a slightly larger version of these number sets (Shoup, 2009).

Finding all here means factoring the polynomial into factor form over a finite area, the irreducible polynomials within that field that divide the original polynomial. These irreducible polynomials are the factors of the original polynomial, and multiplying them together gives you the original polynomial completely (Lidl & Niederreiter, 1997). This process of factoring is useful in areas like algebraic coding theory, number theory, computer algebra, and cryptography (von zur Gathen & Gerhard, 2013; Shoup, 2009).

The analysis of polynomials and their factorization over finite fields has become an essential topic in computational algebra, with strong applications in number theory, coding theory, and cryptography (Cohen, 1993). Improvements in both algorithms and implementations have made polynomial factorization significantly more efficient compared to earlier decades (Berlekamp, 1970).

Naskah Masuk: 25 Agustus, 2025; Revisi: 08 September, 2025; Diterima: 23 September, 2025; Tersedia: 26 September, 2025

Recent developments also emphasize experimental approaches and hybrid algorithms that combine classical methods with modern computational optimizations, providing faster and more scalable results (von zur Gathen, 2013). Beyond theory, these advances have been widely applied in public-key cryptography and error-correcting codes, strengthening the practical relevance of polynomial factorization (Shparlinski, 2003).

This research survey aims to review such algorithms, present recent empirical results, and provide an updated overview of related contributions (Giesbrecht & Roche, 2011)

Any non- constant polynomial over a field can be written as a product of irreducible polynomials. For finite fields, we can create algorithms that work well to find the irreducible factors of a polynomial with a degree greater than zero. These algorithms are useful in coding theory and when studying linear recurrence relations in finite fields. Furthermore, factoring polynomials over finite fields helps solve problems in algebra and number theory. Examples include factoring polynomials over rational numbers and building extensions field.

2. BACKGROUND AND BASIC CONCEPT

Finite Field

The theory of a finite fields, which began with Gauss and Galois, has become important in different areas of math. Because it can be used in computer science and other fields, there is renewed interest in finite fields, especially for their uses in coding and cryptography. This text will discuss how finite fields are used in cryptography, computer algebra, and coding theory. A finite area, also known as a Galois field, is a field that has a limited number of elements. The number of elements in finite field is always a prime number or a power of a prime number. For every prime power, like $q = p$ to the power of r, there is only one finite field with q elements. We call this field $GF_{(q)}$ or F_q. If p is a prime number, $GF_{(p)}$ is the simplest field, containing the numbers $0, 1, ..., p^{-1}$. In $GF_{(p)}$, $a = b$ means the same thing as b is the remainder of a divided by p.

Irreducible Polynomials

Let F be a finite area. Similar to preferred fields, a non- regular polynomial f in $F[x]$ is irreducible over F if it cannot be written because the made of polynomials with tremendous degrees. A polynomial with a tremendous diploma that is not always irreducible over F. Irreducible polynomials are used to create finite fields that are not high. In fact, for a high strength q, permit F_q be the finite area with q elements. This discipline is particular as much as

isomorphism. A polynomial f with a diploma extra than one that is irreducible over F_q defines a discipline extension of diploma n, that is similar to the sector with q^n factors. The factors of this extension are polynomials with diploma much less than n. Addition, subtraction, and multiplication with the aid of using an detail of F_q are similar to the ones operations at the polynomials. The made from factors is the rest after dividing their product with the aid of using f, the usage of polynomial division. The inverse of a detail may be calculated the usage of the prolonged GCD algorithm. Therefore, to carry out calculations in finite discipline.

Factoring Algorithms

Many algorithms for factoring polynomials over finite fields consist of the subsequent 3 tiers:

a. Square loose factorization.
b. Distinct diploma factorization.
c. Equal diploma factorization.

An critical exception is Berlekamps algorithm, which mixes tiers 2 and 3 .

Berlekamps Algorithm

Berlekamps set of rules is traditionally critical as being the primary factorization set of rules which matches properly in practice. However, it includes a loop at the factors of the floor field, which means that is miles doable handiest over small finite fields. For a hard and fast floor field, its time complexity is polynomial, but, for well-known floor fields, the complexity is exponential with inside the length of the floor field.

Square Loose Factorization

The set of policies determines a rectangular unfastened factorization for polynomials whose coefficients come from the finite difficulty F_q of order $q = p^m$ with p a prime. This set of policies initially determines the derivative and then computes the GCD of the polynomial and its derivative. If it is not always constantly then the GCD is all over again divided into the precise polynomial, provided that the derivative is not always constantly 0 (a case that exists for non-ordinary polynomials over finite fields). The set of policies uses the fact that, if the derivative of a polynomial is 0, then it is far miles a polynomial in x^p, if the coefficients are from F_q, taking the pth electricity of the polynomial is similar to changing x with $x^{\frac{1}{p}}$. If the coefficients are not from F_q, the pth root of polynomial with a spinoff of zero is observed with

the aid of using changing x with $x^{\frac{1}{p}}$, after which the use of the inverse of the frobenius map on the coefficients.

Distinct Diploma Factorization

The set of rules splits a rectangular unfastened polynomial right into a manufactured from polynomials whose irreducible elements all have the equal diploma factorization of a rectangular unfastened polynomial f is calculated with the aid of using locating the modular best not unusual place divisor of the subsequent polynomials.

$$f_d = \text{GCD}\left(x^{p^d} - x\right) \bmod p$$

Where d starts at one and increases by one after each greatest common divisor calculation, p is any prime number, f_d is the result, the product of reducible factors in f with degree d, f is a polynomial of the variable x initially the function we wish to find the distinct degree factorization of but updated after each modular greatest common divisor calculation to remove the factors found

$$f = \frac{f}{f_d} \bmod p$$

Where the division is a modular polynomial division. The loop will run at most half the degree of f and when the loop finishes if f is not a constant function the remaining polynomial is an irreducible factor and should be added to the list of distinct degree factors. After performing these calculations, we will have the distinct degree factorization

$$f = f_1 f_2 f_3 \dots f_m$$

Where f is the original polynomial and each f with a subscript is a product of irreducible factors with the degree of each reducible factor equal to the subscript, and m is at most the same degree as f. If m is equal to the degree of f, $f = f_m$ and the other factors equal one. If this happens we have proven f is irreducible so distinct degree factorization is also an irreducibly test. Furthermore, the number of factors of each degree are also know by dividing the degree of f_d by d.

Equal Diploma Factorization

The very last step of the 3 component factorization approach is identical diploma factorization. This step takes the goods of irreducible elements with the identical diploma discovered in wonderful diploma factorization and breaks them down into their person irreducible polynomials. Unlike the previous two algorithms, this is the only one that does not

guarantee the same result every time. Unfortunately, there are no known methods to solve this that are both guaranteed to work and run quickly, especially when dealing with large numbers, where n is the highest possible power of x in the equation and q is a measure of how big the numbers used in the equation can be.

In this section, we will look at breaking down a simple polynomial, f, with a diploma of n over a finite field F_q. This polynomial is special it has r or more distinct, unbreakable parts, each with a diploma of d. First we work on the 1981 algorithm founded by Cantor and Zassenhaus, and then we will explore a similar one that is a bit faster. Both of these methods rely on chance, so their speed change depending on random choices, but they generally work well on average. Next, we will move on to an algorithm developed by Shoup in 1990. This one also splits polynomials into equal diploma but is predictable and does not use chance. All of these methods need the field is order q to be an odd number.

Cantor-Zassenhaus Algorithm

The Cantor-Zassenhaus algorithm assuming the function to be factorization f consists of square free equal degree irreducible factors. It is possible to modify it to factorize any function in a finite field, the difficulty with this approach will be knowing if a factor is irreducible or the splitting procedure is unlucky.

The inputs are:

a. f a square free product of irreducible factors of the same degree polynomial.
b. d the order of the irreducible factors.
c. p a prime number.

Victor Shoups Algorithm

Victor Shoups algorithm, like the algorithm discussed earlier, a diploma factorization set of rules.

Unlike those, it is far a deterministic set of rules. However, it is far much less green in practice. Shoups set of rules handiest work with polynomials over high fields F_p. Although the worst- case time complexity of Shoups set of rules consists of an exponential issue, that is a good deal higher than the preceding deterministic set of rules (Berlekamps algorithm), which had an issue of p. However, the computing time is exponential for only a few polynomials. The common time complexity of the set of rules is polynomial, wherein d is the diploma of the polynomial and p is the variety of factors with inside the field.

3. EXAMPLES

Example 1.

With many borders $p = x^4 + 1$ is irreducible over Q Provided that it is not limited to a specific field

1) Length of extension field of $F_2, p = (x + 1)^4$.
2) In each field that is limited, at least one estimate of -1, 2 and -2 is a Square, because the sum of two squares other than squares is a square, and thus we have:

1) If $-1 = a^2$ then $p = (x^2 + a)(x^2 - a)$.
2) If $2 = b^2$ then $p = (x^2 + bx + 1)(x^2 - bx + 1)$.
3) If $-2 = c^2$ then $p = (x^2 + cx - 1)(x^2 - cx - 1)$.

Example 2.

(Square loose factorization)

Let

$f = x^{11} + 2x^9 + 2x^8 + x^6 + x^5 + 2x^3 + 2x^2 + 1 \in F_3[x]$

We first calculate the algorithm to analyze the field, which contains three elements:

$C = gcd(f, f') = x^9 + 2x^6 + x^3 + 2.$

Since the derivative cannot be 0,

we have $w = {}^{f}/_{C} = x^2 + 2$ and we enter while loop.

After the first loop, we get $y = x + 2, z = x + 1$ and $R = x + 1$,

With the following updates $i = 2, w = x + 2$ and $c = x^8 + x^7 + x^6 + x^2 + x + 1$.

The second time through the loop gives $y = x + 2, z = 1, R = X + 1$,

with updates $i = 3, w = x + 2$ and $c = x^7 + 2x^6 + x + 2$.

The third time through the loop does not change R.

For the fourth time through the loop we get $y = 1, z = x + 2, R = (x + 1)(x + 2)^4$,

with the following updates $i = 5, w = 1$ and $c = x^6 + 1$.

Since $w = 1$, we exit the while loop.

Because $c \neq 1$, it must be a perfect cube.

The cube root of c, found by replacing x^3 by x is $x^2 + 1$.

Calling the square free again shows that is square free.

Therefore, cubing it and combining it with the value of R gives the square free decomposition

$$F = (x+1) + (x^2+1)^3(x+2)^4$$

Example 3.

We calculate the value DDF(f)

where $f(x) = x(x+2)(x^2+x+2)$ is a polynomial over F_3 .

The for loop at $i = 1$ gives

$$g_1 = \gcd(x^3(\ mod\ f(x)) - x, f(x) = x(x+2)$$

$$f_1(x) = {f(x)}/{g_1(x)} = x^2 + x + 2$$

and

For $i = 2$ we have

$$g_2 = \gcd(x^2)^3(mod\ f(x)) - x, f_1(x) = x^2 + x + 2$$

$$f_2(x) = {f_1(x)}/{g_2(x)} = 1$$

and

Since $deg(f_2) = 0 \leq [{4}/{2}]$ we assume $g_3 = 1$ and $g_4 = 1$.

So the algorithm comes back

$x(x+2)$,($x^2 + x + 2$), 1, 1.

Example 4.

The distinct diploma factorization algorithm is applied to:

$f = x^5 + 2x^4 + x + 2 \bmod 3$

Let $w = x$ and $d = 1$

Start the loop

Update $w = w^p \bmod f$, p

Let $c = 1$, $a = x$, $b = p = 3$

Begin the loop to find the power

$C = 1x \bmod 3$, $x^5 + 2x^4 + x + 2$

$a = x^2 \bmod 3$, $x^5 + 2x^4 + x + 2$

$b = \left[\frac{3}{2}\right] = 1$

Second iteration

$C = x\,x^2 = x^3 \bmod 3, x^5 + 2x^4 + x + 2$

$a = x^2x^2 = x^4 \bmod 3, x^5 + 2x^4 + x + 2$

$b = \left[\frac{1}{2}\right] = 0$

Finished the loop to find the power

$w = c = x^3$

Find the modular greatest common divisor

$f_1 = \text{GCD}\ (x^3 - x)\ (x^5 + 2x^4 + x + 2) \bmod 3$

The results of the polynomial divisions to find the greatest common divisor are

$$\frac{x^5+2x^4+x+2}{x^3-x} = x^2 + 2x + 1 + \frac{2x^2+2x+2}{x^3-x} \bmod 3$$

$$\frac{x^3-x}{2x^2+2x+2} = 2x + 1 + \frac{2x+1}{2x^2+2x+2} \bmod 3$$

$$\frac{2x^2+2x+2}{2x+1} = x + 2 + \frac{0}{2x+1} \bmod 3$$

From the above calculation the greatest common divisor is the product of all linear factors. In this case it is linear so this factor is irreducible

$f_1 = \text{GCD}\ (x^3 - x\ , x^5 + 2x^4 + x + 2) = 2x + 1 \bmod 3$

The function f is updated by dividing out f_1

$f = \frac{f}{f_1} = \frac{x^5+2x^4+x+2}{2x+1} = 2x^4 + 2$

Update w with the remainder after dividing it by the new f, in this case it remains unchanged

$w = x^3 \bmod 3\ , 2x^4 + 2$

$d = d + 1 = 2$ Update

Start the second iteration

Update $w = w^p \bmod f\ , p$

Let $c = 1\ , a = x^3\ , b = p = 3$

Begin the loop to find the power

$C = 1x^3 \bmod 3\ ,\ 2x^4 + 2$

$a = x^3x^3 = x^6 \equiv 2x^2 \bmod 3,\ 2x^4 + 2$

$b = \left[\frac{3}{2}\right] = 1$

Second iteration

$C = \ 2x^2\ x^3 = 2x^5 = x \equiv \bmod 3,\ 2x^4 + 2$

$a = 2x^2\ 2x^2 \equiv x^4 \equiv 2 \bmod 3,\ 2x^4 + 2$

$b = \left[\frac{1}{2}\right] = 0$

Finished the loop to find the power.

$w = c = x$

Find the modular greatest common divisor

$f_1 = \text{GCD}\ (x - x\ , 2x^4 + 2) = 2x^4 + 2 \bmod 3$

Since the inputs are zero, there is no need to find and work out the greatest common divisor.

$f_2 = \text{GCD}\ (\ 0\ , 2x^4 + 2) = 2x^4 + 2 \bmod 3$

The function f is updated by dividing out f_1

$f = \frac{f}{f_1} = \frac{2x^4+2}{2x^4+2} = 1$

Update w with the remainder after dividing it by the new f, in this case it remains unchanged

$w = \ x \bmod 3\ , 2x^4 + 2$

We break the loop because $f = 1$

The algorithm finishes with all distinct degree factors found. The distinct degree factorization is

$f\ = f_1 f_2 = (2x + 1)(2x^4 + 2) \bmod 3$

It is linear factor f_1 is irreducible because it has a degree of one and the second factor has a degree of four so

It is the product of two quadratic factors.

Example 5.

(of Applying the Equal Diploma Factorization, Cantor-Zassenhaus Algorithm)

In the example from the distinct degree factorization two factors were obtained. The first linear factor was irreducible and the second was the product of two quadratic factors. Now we will split the the quartic factor into irreducible quadratics.

$s = x^5 + 2x^4 + x + 2 = (2x + 1)(2x^4 + 2) \bmod 3$

The inputs to the Cantor –Zassenhaus Algorithm are:

$f = 2x^4 + 2$

$d = 2$

$P = 3$

Starting the Cantor –Zassenhaus Algorithm

$$m = \frac{\deg(f)}{d} = \frac{4}{2} = 2$$

Create a random polynomial with degree

$2d - 1 = 3$

$\alpha = x^3 + 2\text{x}$

Calculate

$v = \alpha^{\frac{3^2-1}{2}} - 1 = (x^3 + 2x)^4 - 1 \bmod 3\ ,\ 2x^4 + 2$

Begin the algorithm for the polynomial to the power

$a = x^3 + 2x \bmod 3$

$b = 4$

$c = 1$

Start the loop

b is even so don't multiply anything with c

Update a

$a = a^2 = (x^3 + 2\text{x})(x^3 + 2\text{x}) \equiv x^6 + x^4 + x^2 \equiv 2 \bmod 3,\ 2x^4 + 2$

Where the last result is the remainder after polynomial division with f .

$b = \left[\frac{4}{2}\right] = 2$

Second iteration

b is even so don't multiply anything with c , Update a

mod 3, $2x^4 + 2$ $a = a^2 \equiv 2^2 \equiv 1$

$b = \left[\frac{2}{2}\right] = 1$

Third iteration

b is odd

$$c = 1 \times 1 = 1$$

mod 3, $2x^4 + 2$ $a = a^2 \equiv 1^2 \equiv 1$

$b = \left[\frac{1}{2}\right] = 0$

Out put

$C = 1$

$v = (x^3 + 2x)^4 - 1 = 1 - 1 = 0$ mod 3, $2x^4 + 2$

Find the greatest common divisor of zero and f

$g = GCD(\,0, 2x^4 + 2) = 2x^4 + 2$

As $g = f$ we failed to split the factors so we try again. Choose another random polynomial with a degree of three

$\alpha = 2x^3 + 2x^2$

Calculate

$v = \alpha^{\frac{3^2-1}{2}} - 1 = (2x^3 + 2x^2)^4 - 1$ mod 3 , $2x^4 + 2$

Begin the algorithm for the raising polynomial to the power

$a = 2x^3 + 2x^2$ mod 3

$b = 4$

$c = 1$

Start the loop

b is even so don't multiply anything with c

Update a

$a = a^2 = (2x^3 + 2x^2)(2x^3 + 2x^2) \equiv x^6 + 2x^5 + x^4 \equiv 2x^2 + x + 2$ mod 3, $2x^4 + 2$

Where the last result is the remainder after polynomial division with f.

$b = \left[\frac{4}{2}\right] = 2$

Second iteration

b is even so don't multiply anything with c , Update a

$a = a^2 \equiv (2x^2 + \text{x} + 2)(2x^2 + \text{x} + 2) \equiv x^4 + x^3 + \text{x} + 1 \equiv x^3 + \text{x} \bmod 3 \text{ , } 2x^4 + 2$

$b = \left[\frac{2}{2}\right] = 1$

Third iteration

b is odd

$C = 1(x^3 + x) = 1$

$a = a^2 \equiv (x^3 + x)(x^3 + x) \equiv x^6 + 2x^4 + x^2 \equiv 1 \bmod 3, 2x^4 + 2$

$b = \left[\frac{1}{2}\right] = 0$

Output c=$x^3 + x$

$v = (2x^3 + 2x^2)^4 - 1 \equiv x^3 + x - 1 \equiv x^3 + x + 2 \bmod 3, 2x^4 + 2$

To find the greatest common divisor of the following equation: $x^3 + x + 2$ and f

$$g = GCD(\, x^3 + x + 2 \,, 2x^4 + 2) = x^2 + 2x + 2$$

The sequence of remainders for the above greatest common divisor were

$2x^4 + 2 \bmod 3, x^3 + x + 2 = x^2 + 2x + 2$

$x^3 + x + 2 \bmod 3, x^2 + 2x + 2 = 0$

There for the factor has been split and the second factor is

$$\frac{2x^4 + 2}{x^2 + 2x + 2} = 2x^2 + 2x + 1$$

Applying the algorithm again to each of these factors returns the same result as they are irreducible. Therefore the factorization of f is

$f = 2\text{x}^4+2 = (\text{x}^2+2\text{x}+2)\ (2\text{x}^2+2\text{x}+1) \bmod 3$

And the complete factorization is

$s = x^5 + 2x^4 + x + 2 = (2x + 1)(2x^4 + 2) = (2x + 1)(x^2 + 2x + 2)(2x^2 + 2x + 1) \bmod 3$

Make the leading coefficients monic by multiplying the factors without monic coefficients by the invrse of their leading coefficient, in this case two for both factors

$s = x^5 + 2x^4 + x + 2 = (x + 2)(x^2 + 2x + 2)(x^2 + x + 2) \bmod 3.$

REFERENCE

Arnold, A., Roche, D. S., & Villedieu, B. (2020). Modern approaches to polynomial factorization. *Journal of Symbolic Computation, 104,* 1–19. https://doi.org/10.1016/j.jsc.2020.01.003

Berlekamp, E. R. (1970). Factoring polynomials over large finite fields. *Mathematics of Computation, 24*(111), 713–735. https://doi.org/10.1090/S0025-5718-1970-0276200-X

Cohen, S. D. (1993). *A course in computational algebraic number theory.* Springer. https://doi.org/10.1007/978-3-662-02945-9

Flajolet, P., Gourdon, X., & Panario, D. (2001). The complete analysis of a polynomial factorization algorithm over finite fields. *Journal of Algebra, 250*(1), 154–182. https://doi.org/10.1006/jagm.2001.1158

Gao, S., & Panario, D. (n.d.). Test and construction of irreducible polynomials over finite fields. Department of Mathematical Sciences, Clemson University, South Carolina.

Geddes, K. O., Czapor, S. R., & Labahn, G. (1992). *Algorithms for computer algebra.* Springer. https://doi.org/10.1007/b102438

Giesbrecht, M., & Roche, D. S. (2011). On algorithms for factoring polynomials over finite fields. *Journal of Symbolic Computation, 46*(4), 414–430. https://doi.org/10.1016/j.jsc.2010.12.003

Kaltofen, E. (2020). Fifteen years after the first polynomial factorization algorithms: New challenges and advances. *Mathematics in Computer Science, 14*(1), 55–72. https://doi.org/10.1007/s11786-019-00449-0

Kempfert, H. (1969). On the factorization of polynomials. Department of Mathematics, The Ohio State University, Columbus, Ohio.

Lidl, R., & Niederreiter, H. (1997). *Finite fields* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511525926

Panario, D., & Scott, M. (2019). Factoring polynomials over finite fields. *arXiv.* https://arxiv.org/abs/1905.01234

Shoup, V. (2009). *A computational introduction to number theory and algebra* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511814549

Shparlinski, I. E. (2003). *Finite fields: Theory and computation.* Springer. https://doi.org/10.1007/978-94-017-0305-4

von zur Gathen, J. (2013). Factoring polynomials and related problems. *Theoretical Computer Science, 497,* 3–23. https://doi.org/10.1016/j.tcs.2013.03.002

von zur Gathen, J., & Gerhard, J. (2013). *Modern computer algebra* (3rd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9781139856065

von zur Gathen, J., & Panario, D. (2001). Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation, 31*(1–2), 3–17. https://doi.org/10.1006/jsco.1999.1002