SOLVING QUADRATIC CONGRUENCES MODULO A PRIME ON THE TI-89

Joseph Fadyn
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, Georgia 30060
jfadyn@spsu.edu

INTRODUCTION

We consider the problem of solving the quadratic congruence:

$$ax^2 + bx + c \equiv 0 (\mathrm{mod}\, p)$$

where p is an odd prime number using the TI-89. We assume that p does not divide a, for otherwise the congruence reduces to bx + c ≡ 0 (mod p), which is linear. As we shall see, the familiar quadratic formula:

$$x \equiv \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} (\mathrm{mod}\, p)$$

may actually be used to accomplish this task if we interpret the formula correctly. There are two difficulties that are apparent in using this formula to solve a quadratic congruence. First, we see a division by 2a in the quadratic formula. We will simply replace this division by a multiplication by the multiplicative inverse of the quantity 2a modulo p. Since the elements of Z[p] form a field under the operations of addition and multiplication modulo p, we know that every nonzero element of Z[p] has a multiplicative inverse. Since we are assuming that p is an odd prime, we have that 2a is **not** congruent to 0 modulo p so that 2a has an inverse modulo p. The second difficulty is the square root in the formula. We see that the quantity $b^2 - 4ac$ must have a square root modulo p. In the language of number theory we say that $b^2 - 4ac$ must be a quadratic residue modulo p. Let $\delta = b^2 - 4ac$. Then δ is the discriminant in the sense that if δ is not a quadratic residue modulo p then there is no solution to the congruence, if δ ≡ 0 (mod p) there is one solution to the congruence, and if δ is a quadratic residue modulo p then there are two solutions to the congruence. The Euler criterion says that if gcd(δ ,p) =1 then δ is a quadratic residue modulo p if and only if $\delta^{\frac{(p-1)}{2}} \equiv 1 \ (\mathbf{mod}\, p)$. In the case that δ is a quadratic residue modulo p, let r denote a square root of δ modulo p. There are three cases to consider:

Case 1: p ≡ 3 (mod 4). Then p = 4n + 3 for some positive integer n and we have:

$$r \equiv \pm\delta^{n+1} \ (\mathrm{mod}\ p)$$

Case 2: p ≡ 5 (mod 8). Then p = 8n + 5 for some positive integer n and we have:

$r \equiv \pm\delta^{n+1}$ (mod p) or $r \equiv \pm 2^{2n+1}\delta^{n+1}$ (mod p).

Case 3: $p \equiv 1 \pmod 8$. In this case we use Shank's Algorithm. We use the following version of Shanks Algorithm adopted with minor changes and corrections from the online *Library of Math*:

Let x be a solution to $x^2 \equiv \delta \pmod p$, let n, k be integers such that $p - 1 = 2^n k$ where $n \geq 1$ and k is odd, and let q be a quadratic *nonresidue* modulo p. The x can be found by repeating the loop:

1. Set $r \equiv \delta^k \pmod p$ and $t \equiv \delta^{\frac{(k+1)}{2}} \pmod p$.
2. Find the least i such that $r^{2^i} \equiv 1 \pmod p$.
3. If i = 0 then the solutions are $x \equiv \pm t \pmod p$.
4. If i > 0, set $u \equiv q^{k(2^{n-i-1})} \pmod p$, and go to step 2 and replace t by tu and r by ru^2.

EXAMPLE

Let us solve the congruence: $197x^2 - 2569x + 4894 \equiv 0 \pmod{13267}$. A naïve use of the quadratic formula gives:

$$x = \frac{(2569 \pm \sqrt{2743289})}{394}$$

Here $\delta \equiv 2743289 \equiv 10287 \pmod{13267}$ and (using modular exponentiation on the TI-89), $10287^{6633} \equiv 1 \pmod{13267}$, so there are two solutions. Since $13267 \equiv 3 \pmod 4$, the square roots of 10287 modulo 13267 are given by $\pm\ 10287^{3317} \pmod{13267} \equiv \pm 4508 \pmod{13267}$. In addition, since the inverse of 394 mod (13267) is 9462, our solutions are given by $9462\,(2569 \pm 4508) \pmod{13267}$. This reduces to $x \equiv 1443$ and $x \equiv 4025 \pmod{13267}$.

IMPLEMENTATION ON THE TI-89

From the example above we see that we will need, in addition to the main program, a program to perform modular exponentiation and a function to find inverses modulo p. Here is a program to compute b^e (mod n) where n is not necessarily a prime:

```
mdexp(b,e,n)
Prgm
1 → z : mod(b,n) → m
While e ≠ 0
  diva(e,2) → d: mod(z*m^d[1,2],n) → z: d[1,1] → e: mod(m^2,n) → m
Endwhile
EndPrgm
```

In the above program, "diva" is a user defined function (which is an implementation of the division algorithm) and is defined as follows:

```
diva(aa, bb)
Func
  [floor(aa/bb), aa – bb * floor(aa/bb)]
EndFunc
```

Now here is a function to find the inverse of a modulo m (where m need not be prime):

```
mdinv(a,m)
Func
mod(a,m) → a
If gcd(a,m) ≠ 1 Then
  Return "NO INVERSE"
Endif
Local b: 1 → b : Local a1: a → a1 : Local b1: b → b1 : Local m1: m → m1
 Local k: 0 → k : Local di: gcd(a,m) → di
If floor (b/di) ≠ b/di Then
  Return "NO INVERSE"
Else
  Local s0: 1 → s0 : Local x0: 0 → x0 : Local s1: 0 → s1 : Local x1: 1 → x1 :  m → b
    While b ≠ gcd(a,b)
      Local d: diva(a,b) → d :Local s: s0 – d[1,1]*s1 → s : Local x: x0 – d[1,1]*x1 → x
      s1 → s0: x1 → x0: s → s1: x → x1 :  b → a:  d[1,2] → b
    EndWhile
  Local e: [s,x] → e : Local g: e[1,1] → g :  g*b1/gcd(a1,m) → g :  mod(g,m) → g
EndIf
Return mod(g,m)
EndFunc
```

Finally we have our main program to solve a quadratic congruence modulo a prime p. This program handles the cases p = 2 and p | a as special cases:

```
quadcong()
Prgm
ClrIO
Request "Enter a", a : expr(a) → a
Request "Enter b" , b : expr(b) → b
Request "Enter c" , c : expr(c) → c
Request "Enter Prime p ≥ 2" , p : expr(p) → p
If not isprime(p) Then : Disp "p is not prime" : Stop : EndIf
If mod(a,p) = 0 Then
  If mod(b,p)=0 Then
    If mod(c,p)=0 Then
```

```
        Disp "All n, 0 ≤ n ≤ p-1 are solutions"
        Stop
      Else
        Disp "No Solution"
        Stop
      Endif
    Endif
Disp "One Solution" : mod(b,p) → b : mod(-c,p)→ c : Disp mod(mdinv(b,p)*c,p) : Stop
Endif
If p = 2 Then
  If mod(a+b+c,2) = 0 Then : Disp "1 is a solution" : Endif
  If mod(c,2) = 0 Then : Disp "0 is a solution" : Endif
  If mod(a+b+c,2) ≠ 0 and mod(c,2) ≠ 0 Then: Disp "No Solution" : Endif
Stop
Endif
mod( b^2 – 4*a*c , p) → di : mdexp(di, (p-1)/2 , p)
If mod(di , p) = 0 Then
   Disp "One Solution" : Disp mod( mdinv (2*a , p) * -b , p) : Stop
ElseIf z = p -1 Then
   Disp "No Solution Exists" : Stop
Else
   Disp "Two Solutions Exist"
EndIf
If fPart(√(b^2-4*a*c) = 0 and b^2-4*a*c > 0 Then
   Disp "Solutions Are:" : Disp mod( mdinv (2*a , p)*(-b + √(b^2-4*a*c) ) , p)
   Disp "And:" : Disp mod( mdinv (2*a , p)*(-b - √(b^2-4*a*c) ) , p) : Stop
EndIf
If mod(p,4) = 3 Then
   mdexp(di, (p+1)/4,p)
   Goto stpe
EndIf
If mod(p,8) = 5 Then
   mdexp(di, (p+3)/8,p) : z → w
   If mod(z^2,p) = di Then
      Goto stpe
   EndIf
EndIf
If mod(p,8) = 5 Then
   mdexp(2, (p-1)/4,p) : mod(w*z , p) → z
   If mod(z^2,p) = di Then
      Goto stpe
   EndIf
EndIf
1 → n : p – 1 → h
While fPart(h/2) = 0
```

```
  h/2 → h : n+1 → n
EndWhile
n – 1 → n : (p  - 1)/2^n → k : 2 → q : mdexp(q, (p-1)/2 , p)
While z = 1
  q + 1 → q :  mdexp(q,(p-1)/2, p)
EndWhile
mdexp(di, k, p) : z → r : mdexp(di, (k+1)/2 , p) : z → t
Lbl stp1
0 → i : mdexp(r, 2^i, p)
While z ≠ 1
  i + 1 → i : mdexp(r, 2^i, p)
EndWhile
If i = 0 Then
  Disp "Solutions Are:"
  Disp mod(mdinv(2*a , p)*(-b + t , p)
  Disp "And:"
  Disp mod(mdinv(2*a , p)*(-b –t , p)
  Stop
Else
  k*2^(n-i-1)  → v : mdexp(q,v,p) : z → u : mod(t*u,p) → t : mod(r*u^2,p) → r
  Goto stp1
EndIf
Lbl stpe
     Disp "Solutions Are:"
     Disp mod( mdinv (2*a , p) * (-b + z) , p)
     Disp "And:"
     Disp mod (mdinv (2*a , p) * (-b - z) , p)
EndPrgm
```

As an example we solve: $45738775x^2 - 3978649x + 9183723 = 0 \ modulo\,(123456841)$.
Since 123456841 ≡ 1 (mod 8), this will test our implementation of Shank's Algorithm in the program quadcong(). This problem is substantial and takes about 35 seconds to solve on the TI-89. Here is a screen capture of the output:

To verify using the TI-89, we do: $45738775x^2$ -3978649x +9183723 → f(x). Then: mod(f(99746642), 123456841) =0 , and mod(f(11535544), 123457841) =0 as we expect.