```
#################################################################\
    ############################
##This part is coding according to Lemma 1 on the
    paper###################################
#################################################################\
    ############################
Self_coeffs :=proc(f)
      local G;
      G := coeffs(f, x);
      if degree(f, x) ≥ 2 then return G;
      elif degree(f, x) = 1 then return G[2], G[1];
      fi;
 end proc:
 Self_nops :=proc(F)
    local f, a, b;
    f := F;
    if nops(f) = 2 then a, b := op(f); if degree(b) = 0 and a
    = x and degree(diff(f, x)) ≥ 2 then return 1;  fi;
    else return nops(f);
    fi;
 end proc:



LEM_1 :=proc(f, T, ith, y)
    local p_i, j, factor_deg, M, coeff_i , Coe,  f_degree, f_term,
    f_term_n, i;
        i := ith − 1;
      f_term := op(f);
```

```
  f_term_n := nops(f);
   p_i := 0;
   factor_deg := degree(f_term[i + 1])
  - degree(f_term[f_term_n]);
   Coe := Self_coeffs(f, x);
  for j from 1 to i do
   coeff_i := Coe[j];
   f_degree := degree(f_term[j]) - factor_deg;

   p_i := p_i + x^f_degree · (coeff_i mod 2·T);
  od;

 M := eval(p_i, x = y) mod 2·T;

 if M ≥ T then return M - 2·T;
 else return M;
 fi;


end proc:
#########################################################\
 ###########################

 ##Compute v, V, sigma from m, M and
 sigma###########################################

 ##########################################################\
 ##############################
Sign :=proc(X)
```

```
    if X > 0 then return 1;
    elif X < 0 then return -1;
    else if X = 0 then return 0 fi;
    fi;
 end proc:


Step_1 := proc(a_i, m, M, Sigma, x, f, ith)

   local l, v, V, T, p, s, sigma;
     sigma := Sigma;


 if sigma = 0 then l := ⌊log[2](abs(a_i))⌋;
                        v := l;
                        V := l + 1;
                        sigma := sign(a_i);
 elif sigma ≠ 0 then if 2^m ≥ 2·abs(a_i) then if sigma·a_i
    > 0 then v := m; V := M + 1;
                                                      elif sigma·a_i
    < 0 then v := m − 1; V := M;
                                                      fi;
                           elif 2^m < 2·abs(a_i) then T := 2^M + 1;
                                                      p
    := LEM_1(f, T, ith, x);
                                                      s := p + a_i;
                                                      if s = 0
   then sigma := 0;
```

```
                                              elif s ≠ 0 then l
    := ⌊log[2](abs(s))⌋;
                                                                v
    := l;
                                                                V
    := l + 1;
                                                            sigma
    := Sign(s);
                                              fi;

                        fi;
  fi;
  return v, V, sigma;
 end proc:
#######################################################\
   ##################################
##Compute m and M from v, V and
   sigma##################################################
   ##

   #######################################################\
   ###################################

Step_2 :=proc(x, v, V, sigma, alpha_i)
  local L, m, M;

  L := ⌊alpha_i·log[2](x)⌋;

  if sigma ≠ 0 then m := v + L − 1; M := V + L + 1; fi;
```

```
  return m, M;
 end proc:
######################################################\
  #################################
#Main Part of Theorem
  2###################################################
  #############

  ####################################################\
  ##################################
Sign_Main := proc(X, F)
 local y, f, f_term, f_term_n, sigma, m, M, i, v, V, alpha_i, Coe;

 if X < 0 then f := eval(F, x=-x); y := -X;
 elif X > 0 then f := F; y := X;
 else if X = 0 then return Sign(Self_coeffs(F)[nops(F)]);
   fi;
 fi;

 f_term := op(f);
 f_term_n := nops(f);
 Coe := coeffs(f, x);
 sigma := 0; m := 0; M := 0;


 for i from 1 to f_term_n - 1 do
     v, V, sigma := Step_1(Coe[i], m, M, sigma, y, f, i);
```

```
        alpha_i := degree(f_term[i]) - degree(f_term[i
    + 1]);
          m, M := Step_2(y, v, V, sigma, alpha_i);
 od:
  v, V, sigma := Step_1(Coe[f_term_n], m, M, sigma, y, f,
  f_term_n);

 return sigma;

 end proc:
######################################################\
   ###########################
##This part is for establishing the bracket of the
   intervals###############################
######################################################\
   ###########################

TurnBracket :=proc(C, M)
  local k, i, G;
   k := nops(C);
   G := [ ];
  if k = 1 then G := [[-M, C[1][1]], [C[1][2], M]];
  elif k ≠ 1 then

                  if C[1][1] = -M then G := [ ];
                     else G := [[-M, C[1][1]]];
                  fi:
                  for i from 1 to k − 1 do
                        G := [op(G), [C[i][2], C[i + 1][1]]];
```

```
            od;

            if C[k][2] ≠ M then G := [op(G), [C[k][2],
  M]]; fi;
  fi;

  return G;
end proc:
###########################################################\
  ############################
##This part is the bi-section algorithm for picking the
  interval############################
###########################################################\
  ############################

Bisection := proc(p, U, V)
    local u, v, N, mid;
     u := U; v := V;
     N := 0;
    while N ≤ V − U do
          mid := floor((u + v)/2);
          if eval(p, x = mid) = 0 then u := mid; v := mid;
  break; fi;
          if Sign_Main(u, p) = Sign_Main(mid, p) then u
  := mid;
          else v := mid;
```

```
            fi;
            N := N + 1;
            if  abs(u − v) = 1 then break; fi;
      end do;
     return u, v;

 end proc:
##########################################################\
   ############################
##This part is coding according to Prop 1 on the
   paper####################################
##########################################################\
   ############################

Prop_1 :=proc(C, p, M)
      local C_inv, k, i, G, u, v;
       C_inv := TurnBracket(C, M);
       k := nops(C_inv);
       G := [ ];
       for i from 1 to k do
           if C_inv[i][1] = C_inv[i][2] then G := [op(G),
   [C_inv[i][1], C_inv[i][2]]]; fi;
           if Sign_Main(C_inv[i][1], p)
   ·Sign_Main(C_inv[i][2], p) < 0
           then  u, v := Bisection(p, C_inv[i][1],
   C_inv[i][2]);
                     G := [op(G), [C_inv[i][1], u], [v,
    C_inv[i][2]]];
           else  G := [op(G), [C_inv[i][1], C_inv[i][2]]];
```

```
        fi;

    od;
    return TurnBracket(G, M);
end proc:
############################################################\
  ###########################
##This part is to get the derivatives, i.e. p_i  on the
  paper###############################
############################################################\
  ###########################

Dev := proc(F)
    local k, f, i, G;
      f := F;
      k := nops(F);
      G := [ ];
    for i from 1 to k do
       f := expand(f / x^ldegree(f, x));
      G := [op(G), f];
       f := diff(f, x);
    od;
   return G;
end proc:
############################################################\
  ###########################
##This part is coding according to Theorem 1 on the
  paper##############################
```

```
###########################################################\
######S####################

Thm_1_Main := proc(F)
 local f, G_dev, k, i, C, M, K, S;

  f := expand(F);
 if degree(f)  = 0 and f ≠ 0 then return { }; fi;
 if f = 0 then return {0}; fi;
 if degree(f) ≥ 1 and Self_nops(f) = 1 then return {0};
 fi;

 G_dev := Dev(f);
     k := nops(G_dev);
     C := [[0, 0]];
     M := abs(Self_coeffs(f, x)[nops(f) ]);


 for i from 1 to k − 1 do
  C := Prop_1(C, G_dev[k − i], M);
 od:

  S := [ ];
 K := nops(C);

 if eval(f, x =-M) = 0 then S := [op(S), -M]; fi;
 for i from 1 to K do
     if eval(f, x = C[i][1]) = 0 and C[i][1] ≠ -M  then S
  := [op(S), C[i][1]]; fi;
```

if $eval(f, x = C[i][2]) = 0$ **and** $C[i][1] \neq C[i][2]$
and $C[i][1] \neq C[i-1][2]$
then $S := [op(S), C[i][2]]$;**fi**;
od;
if $eval(f, x = M) = 0$ **then** $S := [op(S), M]$; **fi**;

return $\{op(S)\}$;

end proc:
##\
###################################
##############################*Example*####################\
###################################

##\
####################################

$Thm\_1\_Main\left(x^{1999} - 2x + 1\right)$

$$\{1\} \tag{1}$$

$Thm\_1\_Main\left(x^{2^5} + 44x^{190} - 522x^3 + 3442\right)$

$$\{\ \}$$

$f := expand\left((x-2)\cdot(x-1)\cdot(x+10)\cdot(x-33)\cdot(x+45)\cdot x^3\cdot(x-111)\right)$:
$Thm\_1\_Main(f)$;
$roots(f, 1)$;

$$\{-45, -10, 0, 1, 2, 33, 111\}$$

$$[[-10, 1], [2, 1], [1, 1], [111, 1], [33, 1], [-45, 1], [0, 3]]$$

$$\{-45, -10, 0, 1, 2, 33, 111\}$$

$$[[-10, 1], [2, 1], [1, 1], [111, 1], [33, 1], [-45, 1], [0, 3]] \tag{4}$$


```
CommonRoot := proc(F, k)
  local f, i, g, h, A, B, f_term;
   f := F;
   f_term := op(f);
   h := 0;
   for i from 1 to k do
       h := h + f_term[i];
   od;
   h := expand(h);
   g := expand(f − h);
   A := Thm_1_Main(g);
   B := Thm_1_Main(h);

   return {op(A)} ∩ {op(B)};
end proc:
Prop_2 := proc(F)
     local f, M, i , k, gap, f_term, f_term_n, Newgap;
      f := F;
      M := max(Self_coeffs(f) );
     f_term := op(f);
     f_term_n := Self_nops(f);
```

```
  k := 1;
  gap := 0;

  for i from 1 to f_term_n − 1 do
      Newgap := degree(f_term[i]) − degree(f_term[i
  + 1]);
      if gap < Newgap then gap := Newgap; k := i; fi;
  od;
  if gap > 1 + evalf(log[2](M)) then
  return CommonRoot(f, k);
  else return Thm_1_Main(f);
  fi;
end proc:
```

$Prop\_2(x^{200} - x^{198} + x - 1);$

$$\{1\} \tag{5}$$

$Prop\_2(f);$

$$\{-45, -10, 0, 1, 2, 33, 111\} \tag{6}$$

$Prop\_2(x^{1999} - 2x + 1);$

$$\{\ \} \tag{7}$$

$Prop\_2(x^{1999});$

$$\{0\} \tag{8}$$

$Prop\_2(2)$

$$\{\ \} \tag{9}$$