Algorithms implemented by Antonio Laface using MAGMA

```
// This function Computes Mr:
// Input: the cubic threefold X, a line L of X, an
integer r > 0
// Output: the number Mr

Mr := function(X,L,r)
 assert IsSubscheme(L,X);
 Q := BaseField(X);
 assert Characteristic(Q) ne 2;
 P<[x]> := Ambient(X);
 bb := Equations(L) cat
MinimalBasis(Ideal(NormalForm(x,Ideal(L))));
 g := map<P->P | bb>;
 h := Equation(g(X));
 l1 := Coefficient(h,4,2);
 l2 := Coefficient(Coefficient(h,4,1),5,1)/2;
 l3 := Coefficient(h,5,2);
 q1 := Coefficient(Coefficient(h,4,1),5,0)/2;
 q2 := Coefficient(Coefficient(h,5,1),4,0)/2;
 f := Coefficient(Coefficient(h,4,0),5,0);
 M := SymmetricMatrix([f,q1,l1,q2,l2,l3]);
 de :=
[Determinant(Submatrix(M,Remove([1..3],i),Remove([1..3],i
))) : i in [1..3]];
 P2<[x]> := ProjectivePlane(ext<Q|r>);
 C := Scheme(P2,Evaluate(Determinant(M),x cat [0,0]));
 pts := Points(C) diff SingularPoints(C);
 lis := [[Evaluate(q,Eltseq(p) cat [0,0]) : q in de] : p
in pts];
 lis1 := [p : p in lis | p[1] ne 0 and IsSquare(-p[1])];
 lis2 := [p : p in lis | p[1] eq 0 and &or[p[i] ne 0 and
IsSquare(-p[i]) : i in [2,3]]];
 return #lis - 2*(#lis1 + #lis2);
end function;


// This function return the curve Gamma defined
// by a line L on a cubic threefold X
// Input: the cubic threefold X, a line L of X
// Output: the curve Gamma
```

```
Gam := function(X,L)
 assert IsSubscheme(L,X);
 Q := BaseField(X);
 assert Characteristic(Q) ne 2;
 P<[x]> := Ambient(X);
 bb := Equations(L) cat
MinimalBasis(Ideal(NormalForm(x,Ideal(L))));
 g := map<P->P | bb>;
 h := Equation(g(X));
 l1 := Coefficient(h,4,2);
 l2 := Coefficient(Coefficient(h,4,1),5,1)/2;
 l3 := Coefficient(h,5,2);
 q1 := Coefficient(Coefficient(h,4,1),5,0)/2;
 q2 := Coefficient(Coefficient(h,5,1),4,0)/2;
 f := Coefficient(Coefficient(h,4,0),5,0);
 M := SymmetricMatrix([f,q1,l1,q2,l2,l3]);
 P2<[x]> := ProjectivePlane(Q);
 C := Scheme(P2,Evaluate(Determinant(M),x cat [0,0]));
 return Curve(C);
end function;



// This function returns the characteristic polynomial
P_1(F(X))
// Input: the cubic threefold X, a line L of X
// Output: the polynomial P_1(F(X))


ZetaG := function(X,L)
 f := Equation(X);
 Q := BaseField(X);
 q := #Q;
 R<t> := PolynomialRing(Rationals());
 n := [Mr(X,L,r) : r in [1..5]];
 u := -&+[n[r]/r*t^r : r in [1..5]];
 ll := Coefficients(R!&+[u^k/Factorial(k) : k in [0..5]])
[1..6];
 cf := ll cat [q^i*ll[6-i] : i in [1..5]];
 return &+[cf[i]*t^(i-1) : i in [1..11]];
end function;
```