Code for SageMath used in
"Experiments with Ceresa classes of cyclic Fermat quotients"
(Proc. Amer. Math. Soc. 151 (2023), 931-947, DOI: 10.1090/proc/16178)
David T.-B. G. Lilienfeldt and Ari Shnidman

```
def reduced_triples(m):
    T = []
    for a in range(1,m-1):
        for b in range(1,m-1):
           if gcd(m,gcd(a,gcd(b,a+b))) == 1 and is_reduced_triple(m,a,b):
              if genus(m,a,b) > 2:
                 T.append([a,b,m-a-b])
    return T


def has_cyclic_quot(m):
    T = []
    for trip in reduced_triples(m):
        [a,b,c] = trip
        if has_goodaut2(m,a,b) == True:
            T.append(trip)
    return T

def has_good_auto(m,a,b):
    #print('(m,a,b,c) = ',(m,a,b,m-a-b),'genus is ', genus(m,a,b))
    for i in range(m):
        for j in range(m):
            if has_eig_1(m,a,b,i,j) == False:
 #                print(m,a,b,i,j)
                return True
    return False

def genus(m,a,b):
    return (m-gcd(m,a)-gcd(m,b) - gcd(m,a+b) + 2)/2

def mult(m,a,b,d):
    c = m-a-b
    aa = (a*d) % m
    bb = (b*d) % m
    cc = (c*d) % m
    if aa*bb*cc != 0:
        g = gcd(aa,gcd(bb,cc))
        mm = m/g
        return [mm, aa/g, bb/g, cc/g]

def has_goodaut2(m,a,b):
    g = genus(m,a,b)
    L = []
    for r in range (1,m):
       for s in range(1,m-r):
          if (b*r - a*s) % m == 0:
             L.append([r,s])
    if g != len(L):
       print("genus inconsistent with len(L)")
    i = 0
    boo = False
    while boo == False:
       [r,s] = L[i]
       if gcd(r,s) == 1:
           boo = True
```

```
        i = i + 1
    gg,i,j = xgcd(r,s)
    for k in range(1,m):
        if haseig2(m,a,b,k*i,k*j,L) == False:
            return True
    return False

def haseig2(m,a,b,i,j,L):
    g = genus(m,a,b)
    V = []
    for [r,s] in L:
        V.append((i*r + j*s) % m)
    Vd = [(-v) % m for v in V]
    for t in wedgeeigs(Vd,g,m):
        for u in wedgeeigs(V,g-3,m):
            if (t + u) % m == 0:
                return True
#       for t in wedgeeigs(Vd,g-1,m):
#           for u in wedgeeigs(V,g-2,m):
#             if (t+u) % m == 0:
#                return True
    return False

def has_eig_1(m,a,b,i,j):
    g = genus(m,a,b)
    L = []
    for r in range (1,m):
        for s in range(1,m - r):
            if (b*r - a*s) % m == 0:
                L.append([r,s])
    if g != len(L):
        print("genus inconsistent with len(L)")
    V = []
    for [r,s] in L:
        V.append((i*r + j*s) % m)
    Vd = [(-v) % m for v in V]
    for t in wedgeeigs(Vd,g,m):
        for u in wedgeeigs(V,g-3,m):
            if (t + u) % m == 0:
                return True
    for t in wedgeeigs(Vd,g-1,m):
        for u in wedgeeigs(V,g-2,m):
          if (t+u) % m == 0:
                return True
    return False

def wedgeeigs(W,k,m):
    g = len(W)
    if k == 0:
        return [0]
    if k == 1:
        return [w for w in W]
    pro = sum(w for w in W) % m
    if k == g:
        return [pro]
    n = g - k
    T = sublist(W,n)
    eigs = [(pro - sum(t0 for t0 in t)) % m for t in T]
    return eigs

def sublist(V,n):
```

```
    g = len(V)
    T = range(g)
    S = Subsets(T,n)
    LS = []
    for S0 in S:
        LS.append([V[s] for s in S0])
    return LS

def cmtype(trip):
    [r,s,t] = trip
    m = r + s + t
    HH = []
    for k in range(m):
        if gcd(k,m) == 1:
           rr = (k*r) % m
           ss = (k*s) % m
           tt = (k*t) % m
           if rr + ss + tt == m:
              HH.append(k)
    return HH

def Wgroup(trip):
    [r,s,t] = trip
    m = r + s + t
    WW = []
    HH = cmtype(trip)
    HH.sort()
    for k in range(m):
        if gcd(k,m) == 1:
           Hn = [(k*h) % m for h in HH]
           Hn.sort()
           if Hn == HH:
              WW.append(k)
    return WW

def is_reduced_triple(m,a,b):
    S = [a,b,m-a-b]
    SS = [a,b,m-a-b]
    SS.sort()
    if SS < S:
       return False
    for t in range(1,m):
        if gcd(t,m) == 1:
           aa = (t*a) % m
           bb = (t*b) % m
           if m - aa-bb > 0:
               T = [aa, bb, m-aa-bb]
               T.sort()
               if T < S:
                  return False
    return True

def reduce(m,a,b):
    S = [a,b,m-a-b]
    TT = []
    for t in range(1,m):
        if gcd(t,m) == 1:
           aa = (t*a) % m
           bb = (t*b) % m
           if m-aa-bb > 0:
               T = [aa,bb,m-aa-bb]
```

```
                T.sort()
                TT.append(T)
    TT.sort()
    return TT[0]

def is_cm_equiv(S,T):
    S.sort()
    for t in range(1,m):
        if gcd(t,m) == 1:
           TT = [(t*s) % m for s in T]
           TT.sort()
           if TT == S:
              return True
    return False
```