Gauss AGM Computation of Ω_E

William Stein

February 23, 2009

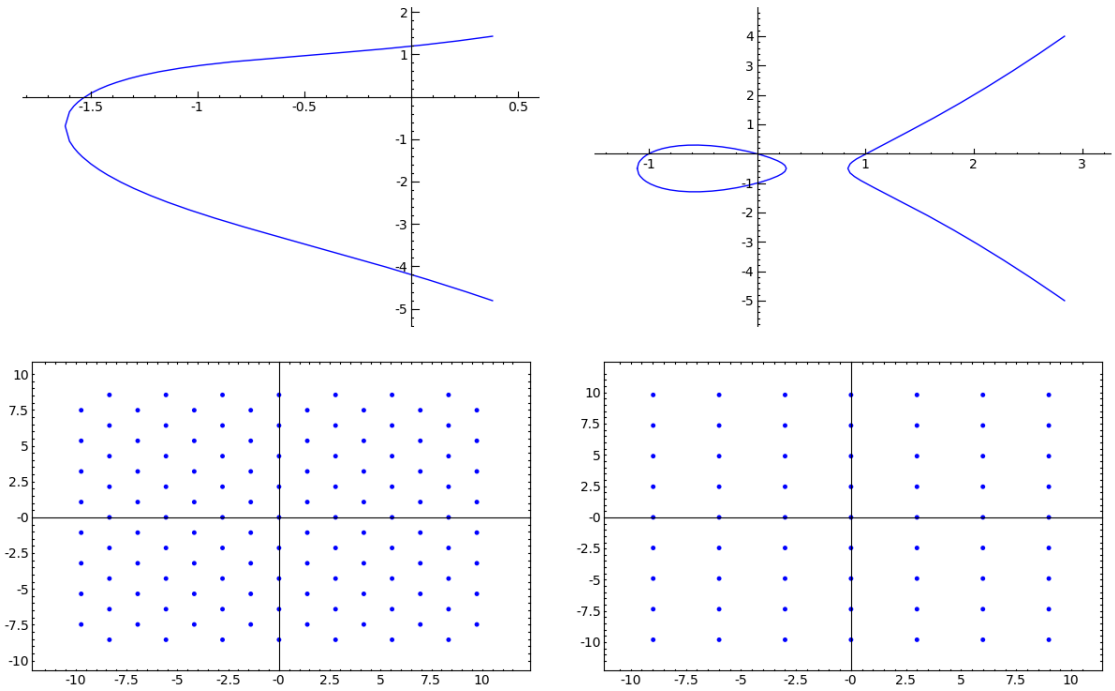

Figure 1: Period Lattices of $y^2 + xy + 3y = x^3 + 2x^2 + 4x + 5$ and $y^2 + y = x^3 - x$

Listing 1: AGM

```
def AGM(a, b):
    """
    Compute arithmetic-geometric mean of a and b, iterating until
    two successive values of a are the same.

    INPUT:
        a, b -- numbers
    OUTPUT:
        the AGM of a and b

    EXAMPLES:
        sage: AGM(1.0, 1.0)
        1.00000000000000
        sage: AGM(1, sqrt(2,prec=53))
        1.19814023473559
    """
    if b < a:
        a, b = b, a
    last_a = b
    while a != last_a:
        last_a = a
        a, b = (a+b)/2, (a*b).sqrt()
    return (a+b)/2
```

Listing 2: Periods

```
def periods(E, real_field=RDF):
    """
    Return generators for the period lattice of the elliptic
    curve E as elements of the given real_field.

    The AGM iteration is done until it stabilizes.

    INPUT:
        E -- elliptic curve over subfield of reals
        real_field -- a fixed precision real field
                      (e.g., RDF, RealField(prec))
    OUTPUT:
        omega1 -- least real period
        omega2 -- complex period
```

```
    EXAMPLES:
        sage: E = EllipticCurve([1..5])
        sage: E.discriminant()
        -10351
        sage: periods(E)
        (2.78074001377, -1.39037000688 + 1.06874977636*I)
        sage: periods(E, RealField(100))
        (2.7807400137662977710631976272,
-1.3903700068833648855315988136+1.0687497763561930661592635474*I)
        sage: time w = periods(E, RealField(10000))
        Time: CPU 0.03 s, Wall: 0.03 s
        sage: time w = periods(E, RealField(10^5))
        Time: CPU 0.92 s, Wall: 0.92 s
        sage: time w = periods(E, RealField(10^6))
        Time: CPU 46.68 s, Wall: 46.75 s
        sage: E.period_lattice().basis()
        (2.78074001376673, -1.39037000688336 + 1.06874977635619*I)

    An example with positive discriminant:
        sage: E = EllipticCurve([0,0,1,-1,0])
        [0, 0, 1, -1, 0]
        sage: E.discriminant()
        37
        sage: periods(E)
        (2.99345864623, 2.45138938199*I)
        sage: E.period_lattice().basis()
        (2.99345864623196, 2.45138938198679*I)

    """
    b2, b4, b6, _ = E.b_invariants()
    Delta = E.discriminant()
    R.<x> = real_field[]
    f = 4*x^3 + b2*x^2 +2*b4*x + b6
    PI = real_field(pi)
    I = real_field.complex_field().gen()
    if Delta > 0:
        # Disconnected case (2 real components)
        roots = [r for r,e in f.roots(real_field)]
        roots.sort()
        e3, e2, e1 = roots
        omega1 = PI / AGM(sqrt(e1-e3),sqrt(e1-e2))
        omega2 = I*PI/AGM(sqrt(e1-e3),sqrt(e2-e3))
```

```
    else:
        e1 = f.roots(real_field)[0][0]
        a = 3*e1 + b2/4
        b = sqrt(3*e1^2 + (b2/2)*e1 + b4/2)
        omega1 = 2*PI/AGM(2*sqrt(b), sqrt(2*b+a))
        omega2 = -omega1/2 + I*PI/AGM(2*sqrt(b), sqrt(2*b-a))
    return omega1, omega2
```

Listing 3: The Birch and Swinnerton-Dyer Ω_E

```
def Omega(E, real_field=RDF):
    r"""
    Compute the quantity $\Omega_E$ in the BSD conjecture.

    INPUT:
        E -- elliptic curve over a subfield of the real field
        real_field -- fixed precision real field
    OUTPUT:
        the BSD Omega_E

    EXAMPLES:
        sage: E = EllipticCurve([1..5])
        sage: Omega(E)
        2.78074001377
        sage: E.period_lattice().omega()
        2.78074001376673
        sage: E = EllipticCurve([0,0,1,-1,0])
        sage: Omega(E)
        5.98691729246
        sage: E.period_lattice().omega()
        5.98691729246392
    """
    om = periods(E, real_field)
    if E.discriminant() > 0:
        return 2*om[0]
    else:
        return om[0]
```