Sage Implementation of LLL (Math 582e)

William Stein

February 4, 2009

Listing 1: Implemenation of LLL

```
def LLL(Q, verbose=True):
    ”””
    INPUT: Q −− Gram inner product matrix, so
                Q[i,j] = inner product of e_i and e_j.
    OUTPUT: LLL reduced basis for ZZˆn with respect to Q and
            Gram−Schmidt coefficients matrix mu
    EXAMPLES:
        sage: L = matrix(ZZ,2,[1,2,3,4])
        sage: Q = L * L.transpose()        # Gram matrix of L
        sage: Z, mu = matrix(LLL(Q)); Z
        k=1; swapping
        k=2; step 2 Lovasz condition satisfied
        [−2  1]
        [ 3 −1]
        sage: Z * L        # Z transforms A to LLL reduced form
        [1 0]
        [0 2]
        sage: Z*Q*Z.transpose()
        [1 0]
        [0 4]
        sage: LLL(Z*Q*Z.transpose())
        k=1; step 1 worked with mu=0
        [(1, 0), (0, 1)]
    ”””
    n = Q.nrows()
    assert n >= 1 and Q.is_square() and Q.is_symmetric(),
             ”Q must be square and symmetric”
    e = Q.change_ring(RDF).eigenvalues()
    assert len(e) == n and min(e) > 0, ”Q must be positive definite”

    # We start with the basis ZZˆn.
    V     = ZZˆn
    b     = list(V.basis())
    bstar = [V(0) for _ in range(n)]   # gram−schmidt basis
    mu    = matrix(Q.base_ring().fraction_field(), n, n)
    def dot(v,w): return (v*Q*w)
    def gram_schmidt(kmax=n−1):
        bstar[0] = b[0]
        for i in [1..kmax]:
            for j in [0..i-1]:
                mu[i,j] = dot(b[i], bstar[j]) / dot(bstar[j], bstar[j])
            bstar[i] = b[i] - sum(mu[i][j] * bstar[j] for j in [0..i-1])

    k = 1
    gram_schmidt()
    while k < n:
        # Step 1: If necessary, reduce b_k so that |mu(k,k-1)| <= 1/2
        # Now reduce b[k] by all other b[j] for j <= k-1
        for j in [k-1,k-2,..,0]:
            if abs(mu[k,j]) > 1/2:
                q = round(mu[k,j])
                b[k] = b[k] - q * b[j]
                mu[k,j] = mu[k,j] - q
                for i in [0..j-1]:
                    mu[k,i] = mu[k,i] - q*mu[j,i]
        # Step 2: Check Lovasz condition
        if dot(bstar[k], bstar[k]) >= \
                 (3/4 - mu[k,k-1]^2)*dot(bstar[k-1], bstar[k-1]):
            if verbose: print ”k=%s; step 2 Lovasz condition satisfied”%k
            k += 1
        else:
            if verbose: print ”k=%s; step 2 Lovasz failed, so swapping”%k
            b[k], b[k-1] = b[k-1], b[k]
            gram_schmidt()
            if k > 1: k -= 1
    # end while loop
    return b, mu
```