

PSEUDO-CODE FOR THE MLLL ALGORITHM (après de Weger)

Author: Keith Matthews (6th October 1997)

The following pseudo-code is extracted from the CALC source file III.c (available at http://www.maths.uq.edu.au/~krm/krm_calc.html) for the function *BASIS_REDUCTION()*, which performs the MLLL algorithm of M. Pohst, J. Symbolic Computation (1987) 4, 123–127.

In Pohst's MLLL algorithm, an integer matrix A whose rows are not necessarily LI over \mathbb{Q} is reduced to a matrix A' , whose first ρ rows constitute a LLL reduced matrix B and whose remaining σ rows are zero. A transformation matrix H , where $HA = A'$, is also returned,

The Gram–Schmidt process plays an additional role to its usual one in the LLL algorithm (where its role is restricted to vectors which are LI) and is used to detect when row β is a LC of the preceding LI rows. The termination of the algorithm is guaranteed by an ingenious trick whereby the possibility that a dependency $\mathbf{b}_k = 0$ and $\mu_{k,k-1} \neq 0$ can occur only finitely many times during the course of the algorithm.

```

INPUT:  $n \times m$  integer matrix  $A$ ;
 $m_1 := 1$ ;  $n_1 := 1$ ;           /*  $\alpha = m_1/n_1$  */
 $D_0 := 1$ ;
 $B := A$ ;  $H := I_n$ ;
found:
 $n := \#$  rows of  $B$ ;
 $K_1 := 0$ ;  $\tau := 2$ ;  $\sigma := 0$ ;
if ( $K_1 = 0$ )  $i := 1$ ;
else  $i := K_1$ ;
/*  $K_1 = \#$  of rows of  $B$  not needing updating by G-S */
for  $i = 1, \dots, m$  {
     $\mathbf{c}_i := \mathbf{b}_i$ ;           /*  $\mathbf{c}_i = D_{i-1} \mathbf{b}_i^*$  */
    for  $j = 1, \dots, i - 1$  {
         $\lambda_{ij} := \mathbf{b}_i \cdot \mathbf{c}_j$ ;       /*  $\lambda_{ij} = D_j \mu_{ij}$  */
         $\mathbf{c}_i := (D_j \mathbf{c}_i - \lambda_{ij} \mathbf{c}_j) / D_{j-1}$ ;
    }
     $flag := 1$ ;
    if ( $\mathbf{c}_i \neq 0$ )  $flag := 0$ ;
    if ( $flag = 1$ ) break;
    else {
         $D_i := (\mathbf{c}_i \cdot \mathbf{c}_i) / D_{i-1}$    /*  $\|\mathbf{b}_i^*\|^2 = D_i / D_{i-1}$  */
    }
}
if ( $flag = 1$ )  $\beta := i$ ;
else  $\beta := i - 1$ ;
 $\rho := K_1 = i - 1$ ;

```

/ K_1 = # of LI rows of B found after G–S process */*
/ $flag = 0$ means the $\rho = \beta$ rows of B are LI */*
/ $flag = 1$ means the first $\rho = \beta - 1$ rows of B are LI, */*
/ but row β is a LC of the preceding rows */*
/ β is the # of rows of B currently being examined */*

```

k :=  $\tau$ ;
while  $k \leq \beta$  {
    Flag := Reduce( $k, k - 1$ );
    if (Flag = 1) { /* Step 9 of Pohst */
         $\sigma := \sigma + 1$ ;
/* relation vector #  $\sigma$  found */
         $k := k + 1$ ;
         $\tau := k$ ;
        goto found;
    }
    if { $n_1(D_{k-2}D_k + \lambda_{kk-1}^2) < m_1D_{k-1}^2$ }{
        if ( $D_k = 0$  &  $\lambda_{k,k-1} = 0$ ) {
             $D_{k-1} := 0$ ;
            Swap1( $k$ );
            if( $k - 1 < K_1$ )  $K_1 := k - 1$ ;
/* the swap may have changed the 2nd last row of B */
             $c_{k-1} := 0$ ;
             $\beta := \beta - 1$ ;
            if ( $k > 2$ )  $k := k - 1$ ;
            continue;
        }
        Swap2( $k, \beta$ );
        Swap1( $k$ );
        if ( $k - 2 < K_1$ )  $K_1 := k - 2$ ;
/* the swap will change last two row of B */
        if ( $k > 2$ )  $k := k - 1$ ;
    }
}

```

```

else {
  for  $i = k - 2, \dots, 1$  {
     $Flag := Reduce(k, i)$ ;
    if ( $Flag = 1$ ) {
       $\sigma := \sigma + 1$ ;
/* relation vector #  $\sigma$  found */
       $\tau := \tau + 1$ ;
      goto found;
    }
  }
   $k := k + 1$ ;
}

```

OUTPUT: $\rho \times m$ matrix B , whose rows are LLL reduced and form a lattice basis for row lattice of A .

$\sigma = n - \rho$, $\mathbf{h}_{\rho+1}, \dots, \mathbf{h}_n$ form a lattice basis for the lattice $XA = 0$.

```

Reduce( $k, i$ )
  flag := 1;
  if  $2|\lambda_{ki}| > D_i$ 
     $q := \lceil \lambda_{ki}/D_i \rceil$ ;
    else  $q := 0$ ;

  if  $q \neq 0$  {
     $\mathbf{b}_k := \mathbf{b}_k - q\mathbf{b}_i$ ;
     $\mathbf{h}_k := \mathbf{h}_k - q\mathbf{h}_i$ ;
     $\lambda_{ki} := \lambda_{ki} - qD_i$ ;

    for  $j = 1, \dots, i - 1$ 
       $\lambda_{kj} := \lambda_{kj} - q\lambda_{ij}$ ;
    }
    if  $(\mathbf{b}_k \neq 0)$  flag := 0;

    if  $(flag = 1)$  {
       $B := DeleteRow(k, B)$ ;
      for  $j = k - 1, \dots, n - 2$ 
         $H := SwapRows(j, j + 1, H)$ ;
      }
    return (flag);

```

Swap1 (k)

$\mathbf{b}_k \leftrightarrow \mathbf{b}_{k-1};$

$\mathbf{h}_k \leftrightarrow \mathbf{h}_{k-1};$

for $j = 1, \dots, k - 2$

$\lambda_{kj} \leftrightarrow \lambda_{k-1j};$

Swap2 (k, β)

for $i = k + 1, \dots, \beta$ {

$t := \lambda_{ik-1}D_k - \lambda_{ik}\lambda_{kk-1};$

$\lambda_{ik-1} := (\lambda_{ik-1}\lambda_{kk-1} + \lambda_{ik}D_{k-2})/D_{k-1};$

$\lambda_{ik} := t/D_{k-1};$

}

$D_{k-1} := (D_{k-2}D_k + \lambda_{kk-1}^2)/D_{k-1};$