success program other than gemver.c (i.e. gemm.c)

```
#pragma scop
  /* C := alpha*A*B + beta*C */
  for (i = 0; i < _PB_NI; i++)
    for (j = 0; j < _PB_NJ; j++)
        C[i][j] = C[i][j]  * beta;
 for (i = 0; i < _PB_NI; i++)
    for (j = 0; j < _PB_NJ; j++)
        for (k = 0; k < _PB_NK; ++k)
        C[i][j] = C[i][j] + alpha * A[i][k] * B[k][j];
#pragma endscop

}
```

Above: original scop

>>>Right: constructed from trace (Loop Boundary set to 10), note that the constructed scop did not recover the first nested loop but the code is correct.

```
#pragma scop
for ( i0 = 0; i0 <= 99; i0++)
    M[ 146104704 + 8*i0 ] =
    M[ 146104704 + 8*i0 ] +
    M[ 3219847392 ] ;
for ( i0 = 0; i0 <= 9; i0++)
    for ( i1 = 0; i1 <= 9; i1++)
        for ( i2 = 0; i2 <= 9; i2++)
            M[ 146104704 + 80*i0 + 8*i1 ] =
            M[ 146104704 + 80*i0 + 8*i1 ] +
            M[ 3219847400 ] +
            M[ 146105536 + 80*i0 + 8*i2 ] +
            M[ 146106368 + 8*i1 + 80*i2 ] ;
#pragma endscop
~
```

failed program: gemm.c with imperfect loop nest

```
#pragma scop
  /* C := alpha*A*B + beta*C */
  for (i = 0; i < _PB_NI; i++)
    for (j = 0; j < _PB_NJ; j++)
      {
    C[i][j] *= beta;
    for (k = 0; k < _PB_NK; ++k)
      C[i][j] += alpha * A[i][k] * B[k][j];
      }
#pragma endscop
```

Above: original scop

>>>Right: constructed from trace (loop boundary set to 10), the code is not correctly reconstructed because the first 9 lines of code got repeated 100 times, where in reality it should be a loop (or a loop nest)

```
#pragma scop
M[ 162668928 ] =
M[ 162668928 ] +
M[ 3214545984 ] ;
for ( i0 = 0; i0 <= 9; i0++)
    M[ 162668928 ] =
    M[ 162668928 ] +
    M[ 3214545992 ] +
    M[ 162669760 + 8*i0 ] +
    M[ 162670592 + 80*i0 ] +
M[ 162668936 ] =
M[ 162668936 ] +
M[ 3214545984 ] ;
for ( i0 = 0; i0 <= 9; i0++)
    M[ 162668936 ] =
    M[ 162668936 ] +
    M[ 3214545992 ] +
    M[ 162669760 + 8*i0 ] +
    M[ 162670600 + 80*i0 ] +
```

failed program: covariance.c with complex loop bounds

Right: part of the original scop

```
/* Calculate the m * m covariance matrix. */
for (j1 = 0; j1 < _PB_M; j1++)
  for (j2 = j1; j2 < _PB_M; j2++)
    {
      symmat[j1][j2] = 0.0;
      for (i = 0; i < _PB_N; i++)
    symmat[j1][j2] += data[i][j1] * data[i][j2];
      symmat[j2][j1] = symmat[j1][j2];
    }
```

```
for ( i0 = 0; i0 <= 7; i0++)
  for ( i1 = 0; i1 <= 9 + -1*i0; i1++)
    val Write , 136754368 + 88*i0 + 80*i1 ,  , 136754368 + 88*i0 + 8*i1
val Write , 136755072 ,  , 136755072
val Write , 136755152 ,  , 136755080
val Write , 136755160 ,  , 136755160
```

Above: part of the reconstructed code. The original code is not correctly reconstructed from trace (loop boundary still set to 10). As you can see from above, the loop bounds are not correctly calculated.