```
NNLSJo[A_, f_] := Module[{weights, w, d = A, b = f, i},
  weights = Table[w[i], {i, Dimensions[A][[2]]}];
  {Sqrt[#[[1]]], weights /. #[[2]]} &[
   NMinimize[Prepend[((# >= 0 &) /@ weights), (d.weights - b).(d.weights - b)], weights]]
 ]

(*Coded by Michael Woodhams,from algorithm by Lawson and Hanson,*)
(*"Solving Least Squares Problems",1974 and 1995.*)
bitsToIndices[v_] := Select[Table[i, {i, Length[v]}], v[[#]] == 1 &];
NNLS[A_, f_] := Module[{x, zeroed, w, t, Ap, z,
    q, α, i, zeroedSet, positiveSet, toBeZeroed, compressedZ, Q, R},
   (*Use delayed evaluation so that these are recalculated on the fly as needed:*)
   zeroedSet := bitsToIndices[zeroed];
   positiveSet := bitsToIndices[1 - zeroed];
   (*Init x to vector of zeros,
   same length as a row of A*)debug["A=", MatrixForm[A]];
   x = 0 A[[1]];
   debug["x=", x];
   (*Init zeroed to vector of ones,same length as x*)zeroed = 1 - x;
   debug["zeroed=", zeroed];
   w = Transpose[A].(f - A.x);
   debug["w=", w];
   While[zeroedSet ≠ {} && Max[w[[zeroedSet]]] > 0, debug["Outer loop starts."];
    (*The index t of the largest element of w,*)(*subject to the constraint
     t is zeroed*)t = Position[w zeroed, Max[w zeroed], 1, 1][[1]][[1]];
    debug["t=", t];
    zeroed[[t]] = 0;
    debug["zeroed=", zeroed];
    (*Ap=the columns of A indexed by positiveSet*)
    Ap = Transpose[Transpose[A][[positiveSet]]];
    debug["Ap=", MatrixForm[Ap]];
    (*Minimize (Ap.compressedZ-f) by QR decomp*){Q, R} = QRDecomposition[Ap];
    compressedZ = Inverse[R].Q.f;
    (*Create vector z with 0 in
     zeroed indices and compressedZ entries elsewhere*)z = 0 x;
    z[[positiveSet]] = compressedZ;
    debug["z=", z];
    While[Min[z] < 0, (*There is a wart here:x can have zeros,
     giving infinities or indeterminates.They don't matter,
     as we ignore those elements (not in postitiveSet) but
      it will produce warnings.*)debug["Inner loop start"];
     (*find smallest x[[q]]/(x[[q]]-z[[q]])*)(*such that:q is not zeroed,
     z[[q]]<0*)α = Infinity;
     For[q = 1, q ≤ Length[x], q++, If[zeroed[[q]] == 0 && z[[q]] < 0,
        α = Min[α, x[[q]] / (x[[q]] - z[[q]])];
        debug["After trying index q=", q, " α=", α];];(*if*)];
     (*for*)debug["α=", α];
     x = x + α (z - x);
```

```
 debug["x=", x];
 toBeZeroed = Select[positiveSet, Abs[x[[#]]] < 10^-13 &];
 debug["toBeZeroed=", toBeZeroed];
 zeroed[[toBeZeroed]] = 1;
 x[[toBeZeroed]] = 0;
 (*Duplicated from above*)(*Ap=the columns of A indexed by positiveSet*)
 Ap = Transpose[Transpose[A][[positiveSet]]];
 debug["Ap=", MatrixForm[Ap]];
 (*Minimize (Ap.compressedZ-f) by QR decomp*){Q, R} = QRDecomposition[Ap];
 compressedZ = Inverse[R].Q.f;
 (*Create vector z with 0 in
  zeroed indices and compressedZ entries elsewhere*)z = 0 x;
 z[[positiveSet]] = compressedZ;
 debug["z=", z];]; (*end inner while loop*)x = z;
debug["x=", x];
w = Transpose[A].(f - A.x);
debug["w=", w];]; (*end outer while loop*)Return[x];];(*end module*)
```