```
% Matlab implementation of the "Geometric Mean Decomposition"
% version of Hager, December 3, 2003
%
%A = U*S*V' is the singular value decomposition of A
%         U, V unitary, S diagonal matrix with nonnegative
%         diagonal entries in decreasing order
%  = Q*R*P' is the geometric mean decomposition of A
%         P, Q unitary, R real upper triangular with r_ii =
%         geometric mean of the positive singular values of A,
%         1 <= i <= p, p = number of positive singular values
% All singular values smaller than tol treated as zero

function [Q, R, P] = gmd (U, S, V, tol)

if ( nargin < 4 )
   tol = 0 ;
end
[m n] = size (S) ;
R = zeros (m, n) ;
P = V ;
Q = U ;
d = diag (S) ;
l = min (m, n) ;
for p = l : -1 : 1
   if ( d (p) >= tol )
      break ;
   end
end
if ( p < 1 )
   return ;
end
if ( p < 2 )
   R (1, 1) = d (1) ;
   return ;
end
z = zeros (p-1, 1) ;
large = 2 ;          % largest diagonal element
small = p ;          % smallest diagonal element
perm = [1 : p] ;     % perm (i) = location in d of i-th largest entry
invperm = [ 1 : p ] ; % maps diagonal entries to perm
sigma_bar = (prod (d (1:p)))^(1/p) ;
for k = 1 : p-1
   flag = 0 ;
   if ( d (k) >= sigma_bar )
```

```
   i = perm (small) ;
   small = small - 1 ;
   if ( d (i) >= sigma_bar )
      flag = 1 ;
   end
else
   i = perm (large) ;
   large = large + 1 ;
   if ( d (i) <= sigma_bar )
      flag = 1 ;
   end
end

k1 = k + 1 ;
if ( i ~= k1 )          % Apply permutation Pi of paper
   t = d (k1) ;         % Interchange d (i) and d (k1)
   d (k1) = d (i) ;
   d (i) = t ;
   j = invperm (k1) ;    % Update perm arrays
   perm (j) = i ;
   invperm (i) = j ;
   I = [ k1 i ] ;
   J = [ i k1 ] ;
   Q (:, I) = Q (:, J) ; % interchange columns i and k+1
   P (:, I) = P (:, J) ;
end

delta1 = d (k) ;
delta2 = d (k1) ;
sq_delta1 = delta1^2 ;
sq_delta2 = delta2^2 ;
if ( flag )
   c = 1 ;
   s = 0 ;
else
   c = sqrt ((sigma_bar^2 - sq_delta2)/(sq_delta1 - sq_delta2)) ;
   s = sqrt (1-c*c) ;
end
d (k1) = delta1*delta2/sigma_bar ;            % = y in paper
z (k) = s*c*(sq_delta2 - sq_delta1)/sigma_bar ; % = x in paper
R (k, k) = sigma_bar ;
if ( k > 1 )
   R (1:k-1, k) = z (1:k-1)*c ; % new column of R
   z (1:k-1) = -z (1:k-1)*s ;   % new column of Z
```

```
   end

   G1 = [ c -s
        s  c ] ;
   J = [ k k1 ] ;
   P (:, J) = P (:, J)*G1 ;       % apply G1 to P

   G2 = (1/sigma_bar)*[ c*delta1 -s*delta2
                  s*delta2  c*delta1 ] ;
   Q (:, J) = Q (:, J)*G2 ;       % apply G2 to Q
end

R (p, p) = sigma_bar ;
R (1:p-1, p) = z ;
```