

The m-files in this collection compute square root free Cholesky factorizations of the form $A=L*D*L'$, modified Cholesky factorizations for matrices which may not quite be positive definite, and rank-one updates of the factorizations. These routines were developed for a course in nonlinear programming based on "Practical Optimization" by Gill, Murray, and Wright.

Routines:

```
ldlt      L*D*L' factorization.
ldltdown  L*D*L' rank one down date of factorization
ldltup    L*D*L' rank one up date
mchol     Modified L*D*L' Cholesky factorization for matrices that
          aren't quite positive definite.
testmchol Routine to demonstrate/test the above routines.
```

History:

8/19/94 - Fixed the routines to work under version 4.1.1 of Matlab in addition to version 4.2.

12/20/99 - Updated to work under MATLAB 5.x without warnings.

9/6/02 - Now MATLAB 6.1, with some speed enhancements by Michael Zibulevsky.

Brian Borchers
borchers@nmt.edu
Department of Mathematics
New Mexico Tech
Socorro, NM 87801

```
%
% [L,D]=ldlt(A)
%
% This function computes the square root free Cholesky factorization
%
%   A=L*D*L'
%
% where L is a lower triangular matrix with ones on the diagonal, and D
% is a diagonal matrix.
%
% It is assumed that A is symmetric and postive definite.
%
% Reference: Golub and Van Loan, "Matrix Computations", second edition,
%           p 137.
% Author: Brian Borchers (borchers@nmt.edu)
%
```

```

function [L,D]=ldlt(A)
%
% Figure out the size of A.
%
n=size(A,1);
%
% The main loop. See Golub and Van Loan for details.
%
L=zeros(n,n);
for j=1:n,
    if (j > 1),
        v(1:j-1)=L(j,1:j-1).*d(1:j-1);
        v(j)=A(j,j)-L(j,1:j-1)*v(1:j-1)';
        d(j)=v(j);
        if (j < n),
            L(j+1:n,j)=(A(j+1:n,j)-L(j+1:n,1:j-1)*v(1:j-1)')/v(j);
        end;
    else
        v(1)=A(1,1);
        d(1)=v(1);
        L(2:n,1)=A(2:n,1)/v(1);
    end;
end;
%
% Put d into a matrix.
%
D=diag(d);
%
% Put ones on the diagonal of L.
%
L=L+eye(n);

%
% [newL,newD]=ldltdown(L,D,v)
%
% This function downdates the Cholesky factorization of A to the Cholesky
% factorization of A-vv'. i.e. If
%
%  $A=L*D*L'$ 
%
% then
%
%  $A-v*v'=\text{newL}*\text{newD}*\text{newL}'$ 
%

```

```

% It is assumed that A is symmetric and positive definite. If A-v*v'
% would not be positive definite, then the diagonal elements will
% be adjusted to make sure that D is > 0.
%
% Reference: Gill, Murray, and Wright, "Practical Optimization", p43.
% Author: Brian Borchers (borchers@nmt.edu)
%
function [newL,newD]=ldltdown(L,D,v)
%
% First, find the size of the matrix.
%
n=size(D,1);
%
% Initialize newL and newD.
%
newL=L;
newD=D;
%
% Find the initial values of p and t.
%
p=L\v;
oldt=1-p'*inv(D)*p;
%
% Make sure that D is > 0.
%
if (oldt <= eps),
    oldt=eps;
end;
%
% The main loop. See Gill, Murray, and Wright for details.
%
for j=n:-1:1
    t=oldt+p(j)^2/D(j,j);
    newD(j,j)=D(j,j)*oldt/t;
    beta=-p(j)/(D(j,j)*oldt);
    v(j)=p(j);
    if (j < n),
        newL(j+1:n,j)=L(j+1:n,j)+beta*v(j+1:n);
        v(j+1:n)=v(j+1:n)+p(j)*L(j+1:n,j);
    end;
    oldt=t;
end;

```

```

%
% [newL,newD]=ldltup(L,D,v)
%
% This function updates the Cholesky factorization of A to the Cholesky
% factorization of A+vv'. i.e. If
%
%   A=L*D*L'
%
% then
%
%   A+v*v'=newL*newD*newL'
%
% It is assumed that A is symmetric and positive definite.
%
% Reference: Gill, Murray, and Wright, "Practical Optimization", p43.
% Author: Brian Borchers (borchers@nmt.edu)
%
function [newL,newD]=ldltup(L,D,v)
%
% First, find the size of the matrix.
%
n=size(L,1);
%
% Initialize newL, newD, and t.
%
newL=L;
newD=D;
oldt=1;
%
% The main loop. See Gill, Murray, and Wright for details.
%
for j=1:n,
    p=v(j);
    t=oldt+p^2/D(j,j);
    newD(j,j)=D(j,j)*t/oldt;
    beta=p/(D(j,j)*t);
    if (j < n),
        v(j+1:n)=v(j+1:n)-p*L(j+1:n,j);
        newL(j+1:n,j)=L(j+1:n,j)+beta*v(j+1:n);
    end;
    oldt=t;
end;
end;

```

```

%
% [L,D,E,pneg]=mchol1(G)
%
% Given a symmetric matrix G, find a matrix E of "small" norm and c
% L, and D such that G+E is Positive Definite, and
%
%   G+E = L*D*L'
%
% Also, calculate a direction pneg, such that if G is not PD, then
%
%   pneg'*G*pneg < 0
%
% Note that if G is PD, then the routine will return pneg=[].
%
% Reference: Gill, Murray, and Wright, "Practical Optimization", p111.
% Author: Brian Borchers (borchers@nmt.edu)
%
function [L,D,E,pneg]=mcholmz1(G)
%
% n gives the size of the matrix.
%
n=size(G,1);
%
% gamma, zi, nu, and beta2 are quantities used by the algorithm.
%
gamma=max(diag(G));
zi=max(max(G-diag(diag(G)))));
nu=max([1,sqrt(n^2-1)]);
beta2=max([gamma, zi/nu, 1.0E-15]);
%
% Initialize diag(C) to diag(G).
%
C=diag(diag(G));
%
% Loop through, calculating column j of L for j=1:n
%

L=zeros(n);
D=zeros(n);
E=zeros(n);

for j=1:n,
    bb=[1:j-1];

```

```

ee=[j+1:n];

%
% Calculate the jth row of L.
%
if (j > 1),
    L(j,bb)=C(j,bb)./diag(D(bb,bb));
end;
%
% Update the jth column of C.
%
if (j >= 2),
    if (j < n),
        C(ee,j)=G(ee,j)-(L(j,bb)*C(ee,bb));
    end;
else
    C(ee,j)=G(ee,j);
end;
%
% Update theta.
%
if (j == n)
    theta(j)=0;
else
    theta(j)=max(abs(C(ee,j)));
end;
%
% Update D
%
D(j,j)=max([eps,abs(C(j,j)),theta(j)^2/beta2]);
%
% Update E.
%
E(j,j)=D(j,j)-C(j,j);

%
% Update C again...
%
%%%%%%%%%%%% M.Zibulevsky: begin of changes, old version is commented
%%%%%%%%%%%%

%for i=j+1:n,
%    C(i,i)=C(i,i)-C(i,j)^2/D(j,j);

```

```

%end;

ind=[j*(n+1)+1 : n+1 : n*n]';
C(ind)=C(ind)-(1/D(j,j))*C(ee,j).^2;

end;

%
% Put 1's on the diagonal of L
%
%for j=1:n,
%  L(j,j)=1;
%end;

ind=[1 : n+1 : n*n]';
L(ind)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% M.Zibulevsky: end of changes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% if needed, find a descent direction.
%
if ((nargout == 4) & (min(diag(C)) < 0.0))
    [m,col]=min(diag(C));
    rhs=zeros(n,1);
    rhs(col)=1;
    pneg=L\'rhs;
else
    pneg=[];
end;

return

%
% m-file to test the ldlt m-files.
%
A=rand(5,5);

```

```

while (rank(A) ~= 5),
    A=rand(5,5);
end;
A=A*A';
%
% Testing ldlt.
%
[L,D]=ldlt(A);
disp('ldlt test: should be close to 0')
norm(A-L*D*L')
%
% testing ldltup.
%
v=rand(5,1);
[newL,newD]=ldltup(L,D,v);
disp('ldltup test: should be close to 0')
norm(A+v*v'-newL*newD*newL')
%
% ldlt down test.
%
w=v/1000;
[newL,newD]=ldltdown(L,D,w);
disp('ldltdown test: should be close to 0')
norm(A-w*w'-newL*newD*newL')
%
% mchol test.
%
[L,D,E,pneg]=mchol(A);
disp('mchol test 1: should be close to 0')
norm(A-L*D*L')
B=A-min(eig(A))*1.5*eye(5);
[L,D,E,pneg]=mchol(B);
disp('mchol test 2: should be close to 0')
norm(B+E-L*D*L')
disp('mchol test 3: should be negative')
pneg*B*pneg
%
%
%
%
```