```
                                        % MATLAB file MWLE.m %

function [theta,err,F,Ohat,H]=MWLE(S,theta0,fun,pr,bd,thrs,btol,maxiter,varargin)

% Maximum Wishart Likelihood
% Code last updated Dec 2013.

% Electronic Supplementary Material to
% QUANTIFYING ADVENTITIOUS ERROR IN A COVARIANCE STRUCTURE AS A RANDOM EFFECT
% Psychometrika, 80(3), 619-624
% by  Hao Wu (Boston College, hao.wu.5@bc.edu) and Michael W. Browne (The Ohio State University)

%%%%%% List of Input Arguments %%%%%%%%%%%%%%%%%%%%
% S is the sample covariance matrix.
% theta0 is the initial value of structure parameters in the model
% fun is a function handle of the Covariance structure ...
%   of the form [O,err,R,dOdx,iRdOdx]=fun(theta,...)
%   input and output arguments are
%   theta: a row vector of parameters
%   O: the covariance structure
%   [R,err]=chol(O);
%   dOdx: the derivative of O, a p^2 by q matrix
%   iRdOdx=(R'\otimesR')\dOdx.
% pr prints iteration details:
%   0:  none;
%   1:  only F, Res.Cos, cond.#.H, NPB and NEC;
%   2:  only parameter values;
%   3:  both
% bd is a matrix of length(theta) by 2,
%     giving boundary conditions for parameters
% thrs is the stopping threshold
% btol is the boundary threshold matrix.
%   It can be input as a scalar, a column vector or a 2-columned matrix.
% maxiter is the maximum iterations allowed.
% varargin passes arguments to fun, the function handle.

%%%%%% List of Output Arguments %%%%%%%%%%%%%%%%%%%%
% theta is the parameter estimate
```

```
% err>0 means nonconvergence
% Ohat is the implied covariance matrix
% F is the discrepancy function value
% H is the Fisher information matrix

%%%%%% List of Screen Outputs %%%%%%%%%%%%%%%%%%%
% #: iteration number
% F: discrepancy function value
% Res.Cos: residual cosine
% cond.#.H: conditional number of Hessian (unconstrained parameters)
% NPB: number of parameters on boundary
% NEC: number of effective constraints
% parameters: parameter values


other=varargin;

if nargin<3, error('Too few arguments.');end
s=length(theta0); % number of parameters in the covariance structure.
if nargin<4 || all(pr~=0:3),    pr=1;end
if nargin<5 || isempty(bd),    bd=kron([-Inf,Inf],ones(s,1));end
if nargin<6 || isempty(thrs),    thrs=.1^7; end
if nargin<7 || isempty(btol),    btol=.1^8; end
if nargin<8 || isempty(maxiter),    maxiter=200; end

if all(size(btol)==[1,1])
     btol=btol*ones(s,2);
elseif size(btol,2)==1
     btol=btol*[1,1];
end

[p,p1]=size(S);% p is the number of manifest variables.
if p~=p1|| ~all(all(isreal(S)))
    error('The covariance matrix should be real square matrix.');
elseif any([s,2]~=size(bd))|| any([s,2]~=size(btol))|| any(size(theta0)~=[1,s])
    error('The size of one or more input matrices is wrong.');
end
```

```
bd=[bd(:,1)+btol(:,1),bd(:,2)-btol(:,2)];
inc0=true(s,1);
ubd=~true(s,1);
lbd=ubd;

%------------------------------------------------
% Initial screening

if any(any(S~=S'))
    display('Covariance matrix not symmetric. Lower half used.');
    S=tril(S);
    S=S+triu(S',1);
end
[CS,err]=chol(S);
if err>0,    error('Covariance matrix not positive definite.'); end

detS=prod(diag(CS))^2;
theta=theta0';

if any(theta>bd(:,2)) || any(theta<bd(:,1))
        error('initial value on or out of bounds.');
end
[O,err]=feval(fun,theta0,other{:});
if err~=0
    error('initial value yields non-positive definite covariance');
end

switch pr
        case 0
        case 1
            display(' #       F         Res.Cos     cond.#.H    NPB NEC');
        case 2
            display(' #       F        parameters');
        case 3
            display(' #       F         Res.Cos     cond.#.H    NPB NEC     parameters');
end
```

```
%---------------------------------
% iterations
i=-1;
inc=inc0;
crit=Inf;
d=0;
f=Inf;
invH=zeros(s,s);
while (crit>thrs&&f>thrs&&i<=maxiter&&any(inc))
    i=i+1;
    theta1=theta+d;
    ubd1=find(theta1>bd(:,2));% locations of new effective bounds
    lbd1=find(theta1<bd(:,1));
    bd1=[lbd1;ubd1];
    if ~isempty(bd1) % constrain the proposed value in side the bound
        bds=[bd(lbd1,1);bd(ubd1,2)];
        if pr~=0
            display('back to the boundary.');
        end
        [r,ind]=min((bds-theta(bd1))./d(bd1));  % proportions of distance to the bound
        d=d*r;
        theta1=theta+d;
        theta1(bd1(ind))=bds(ind);
    end
    while 1 % guarantee the discrepancy function is smaller
        [O,err,R,~,iRdOdx]=feval(fun,theta1',other{:}); % R'R=O
        if  err==0
            detO=prod(diag(R))^2;
            A=CS/R;
            iOS=A'*A;
            f1=log(detO/detS)+trace(iOS)-p;
            f1=max([0,f1]);
            if isnan(f1)||isinf(f1)
                err=-1;
            elseif f1>f
                err=-2;
            end
        end
```

```
    if err~=0 && pr==0
        d=d/2;
        theta1=theta+d;
        continue;
    elseif err~=0
        display(['err=',num2str(err),' step halved.'])
        d=d/2;
        theta1=theta+d;
        continue;
    end
    break;
end
theta=theta1;
f=f1;
d=zeros(s,1);

dif=iOS-eye(p);
g=-iRdOdx'*dif(:);
H=iRdOdx'*iRdOdx;


ubd(inc0)=(theta(inc0)==bd(inc0,2)); % current effective upper bounds
lbd(inc0)=(theta(inc0)==bd(inc0,1)); % current effective lower bounds
invH(inc0,inc0)=inv(H(inc0,inc0));
d(inc0)=-invH(inc0,inc0)*g(inc0); % unconstrained search direction
inc=inc0;
if any(d(ubd)>0) ||any(d(lbd)<0) % outside the boundary
    lambda=zeros(s,1);
    effbd=~true(s,1);
    effbd(ubd)=true;
    effbd(lbd)=true;
    lambda(effbd)=invH(effbd,effbd)\d(effbd); % would-be lagrange multiplier
    [rmbd,ind]=max((-lambda.*ubd+lambda.*lbd).*effbd); % removable bounds
    while  rmbd>0 % some effective bounds can be removed
        effbd(ind)=false; % remove inequality constraints;
        lambda(effbd)=invH(effbd,effbd)\d(effbd); % would-be lagrange multiplier
        [rmbd,ind]=max((-lambda.*ubd+lambda.*lbd).*effbd); % removable bounds
    end
```

```
        inc(effbd)=false;
        d(inc)=-H(inc,inc)\g(inc);
        d(~inc)=0;
    end

    condH=cond(H(inc,inc));
    crit=max(g(inc).*g(inc)./diag(H(inc,inc))/f);
    switch pr
        case 0
        case 1
            str1=sprintf('%3d',i);
            str2=sprintf('%0.5e',f);
            str3=sprintf('%0.5e',crit);
            str4=sprintf('%0.5e',condH);
            str5=sprintf('%2d',sum(ubd)+sum(lbd));
            str6=sprintf('%2d',s-sum(inc));
            display([str1,' ',str2,' ',str3,' ',str4,' ',str5,'  ',str6]);
        case 2
            str1=sprintf('%3d',i);
            str2=sprintf('%0.5e',f);
            str7=sprintf('%+15.5e',theta);
            display([str1,' ',str2,' ',str7]);
        case 3
            str1=sprintf('%3d',i);
            str2=sprintf('%0.5e',f);
            str3=sprintf('%0.5e',crit);
            str4=sprintf('%0.5e',condH);
            str5=sprintf('%2d',sum(ubd)+sum(lbd));
            str6=sprintf('%2d',s-sum(inc));
            str7=sprintf('%+15.5e',theta);
            display([str1,' ',str2,' ',str3,' ',str4,' ',str5,'  ',str6,'  ',str7]);
    end
end

%---------------------
% output
if i<=maxiter
    F=f;
```

```
    Ohat=O;
    err=0;
elseif pr~=0
    display('maximum numbers of iterations reached without convergence.');
    err=1;
    H=nan(s,s);
    theta=nan(s,1);
    Ohat=nan(p,p);
    F=NaN;
else
    err=1;
    H=nan(s,s);
    theta=nan(s,1);
    Ohat=nan(p,p);
    F=NaN;
end
```