Computer code for the general gradient projection rotation algorithms

What follows is a discription of some specific computer code for the general orthogonal and oblique gradient projection (GP) rotation algorithms. Also included are examples of how to use these and background references. The code used is Matlab (1995). This was choosen because it is a very transparant matrix language that is easily translated to S, R, and other environments or may be used directly. The code itself in editable form may be downloaded from `http://www.stat.ucla/research`. See Jennrich (2000, 2001) for a more complete discussion of GP algorithms. The algorithms given here differ in some minor ways from these references. They also differ from the algorithms specifically designed for rotation in factor analysis. The latter may be viewed as special cases of the general algorithms.

A general GP algorithm for orthogonal rotation

The general orthogonal rotation problem is to minimize

$$f(T) \quad \text{given} \quad T'T = I$$

where T need not be square. For orthogonal rotation in factor analysis T is square and

$$f(T) = Q(AT)$$

where Q is a factor analysis rotation criterion and A is an initial loading matrix. Here we deal with the general problem rather than this special case.

The computer code has two parts. The first is the basic GP minimization algorithm for orthogonal rotation. The second computes the value and gradient of f at T. The first is used without change for any criterion. The second is criterion specific and must be produced for each criterion of interest.

- What follows is Matlab code for the basic GP algorithm.

```
function [Th,table]=GPorth(T)

al=1;
table=[];
for iter=0:100
```

```
   [f,G]=vgf(T);
   M=T'*G;
   S=(M+M')/2;
   Gp=G-T*S;
   s=norm(Gp,'fro');

   table=[table;iter f log10(s) al];
   if s<10^(-5),break,end

   al=2*al;
   for i=0:10
      X=T-al*Gp;
      [U,D,V]=svd(X,0);
      Tt=U*V';
      [ft,Gt]=vgf(Tt);
      if ft<f-.5*s^2*al,break,end
      al=al/2;
      end
   T=Tt;
   end
Th=T;
```

We will comment on some of the more interesting lines of code. The first line

```
function [Th,table]=GPorth(T)
```

defines the calling sequence for the basic general GP algorithm which is named GPorth. The input matrix `T` is an initial value for the rotation matrix. The output matrix `Th` is the optimum value for the rotation matrix produced by the GPorth algorithm. The output value `table` is a convergence history.

The code line

```
[f,G]=vgf(T);
```

is call to the vgf subroutine. This call produces the value `f` of f at `T` and the value `G` of the gradient of f at `T`.

The remaining lines are standard code for the GP algorithm.

- What follows is Matlab code to compute the value of f and gradient G at an arbitrary T using the quartimax criterion. Because the basic GP algorithm is designed to minimize rather than maximize, the criterion f is the negative of the usual quartimax criterion.

```
function [f,G]=vgf(T)

global A

L=A*T;
L2=L.^2;

f=-sum(sum(L2.*L2));
G=-A'*(L.*L2);
```

The first line of code

```
function [f,G]=vgf(T)
```

defines the calling sequence for the subroutine vgf that computes the value `f` and gradient `G` of f at the input rotation matrix `T`.

The second line

```
global A
```

identifies the initial loading matrix `A`. This might be explicitly inserted here, but it is usually more convenient to obtain it from elsewhere using a global command.

The code lines

```
f=-sum(sum(L2.*L2));
G=-A'*(L.*L2);
```

give the value `f` and gradient `G` of f at `T`. See Jennrich (2001) for derivation of the gradient and other applications of the general GP algorithm.

An example: Thurstone's box problem

What follows is Matlab code for quartimax rotation of the initial loading matrix for Thurstone's (1947, p. 136) box problem. This uses the subroutines from the previous section. It may be used without change for any quartimax rotation problem. For some other form of orthogonal rotation, say that used to fit the DEDICOM model, only the vgf subroutine needs to be changed.

```
global A

A=[
.659 -.736  .138
.725  .180 -.656
.665  .537  .500
.869 -.209 -.443
.834  .182  .508
.836  .519  .152
.856 -.452 -.269
.848 -.426  .320
.861  .416 -.299
.880 -.341 -.354
.889 -.417  .436
.875  .485 -.093
.667 -.725  .109
.717  .246 -.619
.634  .501  .522
.936  .257  .165
.966 -.239 -.083
.625 -.720  .166
.702  .112 -.650
.664  .536  .488
];

T=eye(3);

[T,table]=GPorth(T);

table
L=A*T
```

The matrix `A` is the initial loading matrix for Thurstone's box problem. The `global A` statement in the first line makes `A` available to the vgf subroutine. The statement

```
T=eye(3);
```

sets the initial rotation matrix to the identity. The subroutine call

```
[T,table]=GPorth(T);
```

replaces `T` by its optimal quartimax value and generates `table`, the convergence history. The last line

```
L=A*T
```

generates the quartimax rotation `L` of the initial loading matrix `A`.

The output from this problem is:

```
table =

         0  -10.7152   -0.7427    1.0000
    1.0000  -11.4092   -0.1290    2.0000
    2.0000  -12.7124   -0.0046    2.0000
    3.0000  -13.7183   -0.1260    1.0000
    4.0000  -14.1837   -0.7702    0.5000
    5.0000  -14.2016   -1.1859    0.5000
    6.0000  -14.2042   -1.5949    0.5000
    7.0000  -14.2046   -2.0028    0.5000
    8.0000  -14.2046   -2.4107    0.5000
    9.0000  -14.2046   -2.8185    0.5000
   10.0000  -14.2046   -3.2263    0.5000
   11.0000  -14.2046   -3.6341    0.5000
   12.0000  -14.2046   -4.0419    0.5000
   13.0000  -14.2046   -4.4497    0.5000
   14.0000  -14.2046   -4.8575    0.5000
   15.0000  -14.2046   -5.2653    0.5000


L =
```

```
0.0105   -0.9934   -0.0899
0.1585   -0.1673   -0.9671
0.9823   -0.0950   -0.0819
0.1250   -0.5971   -0.7893
0.8696   -0.4716   -0.0904
0.8757   -0.1410   -0.4523
0.0679   -0.8114   -0.5886
0.4067   -0.9079   -0.1157
0.5771   -0.1424   -0.8065
0.1013   -0.7233   -0.6946
0.5001   -0.9497   -0.0468
0.7413   -0.1403   -0.6636
0.0056   -0.9838   -0.1200
0.2142   -0.1194   -0.9474
0.9551   -0.1083   -0.0392
0.7823   -0.4054   -0.4393
0.3627   -0.7531   -0.5463
0.0163   -0.9662   -0.0521
0.1077   -0.2067   -0.9346
0.9744   -0.0926   -0.0908
```

The algorithm converged smoothly in 15 iterations.

The general GP algorithm for oblique rotation

The general oblique rotation problem is to minimize

$$f(T) \quad \text{given} \quad \text{dg}(T'T) = I$$

where T is p by k with $p \geq k$. In factor analysis applications T is square and

$$f(T) = Q(A(T')^{-1})$$

where Q is a factor analysis rotation criterion and A is an initial loading matrix. Here we deal with the general problem rather than this special case.

As in the orthogonal case, the Matlab code consists of two subroutines. These are very similar to those for the orthogonal case and are listed without comment.

- The basic GPoblq subroutine.

```
function [T,table]=GPoblq(T)

al=1;
table=[];
for iter=0:500
   [f,G]=vgf(T);
   Gp=G-T*diag(sum(T.*G));
   s=norm(Gp,'fro');

   table=[table;iter f log10(s) al];
   if s<10^(-5),break,end

   al=2*al;
   for i=0:10
      X=T-al*Gp;
      v=1./sqrt(sum(X.^2));
      Tt=X*diag(v);
  [ft,Gt]=vgf(Tt);
      if ft<f-.5*s^2*al,break,end
      al=al/2;
  end
   T=Tt;
   end
```

- The vgf subroutine for quartimin rotation.

```
function [f,G]=vgf(T)

global A

[p,k]=size(A);
Ti=inv(T);
L=A*Ti';
L2=L.^2;
N=ones(k,k)-eye(k);

f=sum(sum(L2.*(L2*N)))/4;
```

```
    Gq=L.*(L2*N);
    G=-(L'*Gq*Ti)';
```

An example: Thurstone's box problem

The following Matlab code produces a quartimin rotation of the initial loading matrix from Thurstone's box problem.

```
global A

A=[
.659 -.736  .138
.725  .180 -.656
.665  .537  .500
.869 -.209 -.443
.834  .182  .508
.836  .519  .152
.856 -.452 -.269
.848 -.426  .320
.861  .416 -.299
.880 -.341 -.354
.889 -.417  .436
.875  .485 -.093
.667 -.725  .109
.717  .246 -.619
.634  .501  .522
.936  .257  .165
.966 -.239 -.083
.625 -.720  .166
.702  .112 -.650
.664  .536  .488
];

T=eye(3);

[T,table]=GPoblq(T);

table
```

```
L=A*inv(T')
phi=T'*T
```

The output from this code is

```
table =

         0    2.2302   -0.4516    1.0000
    1.0000    2.2252   -0.6220    0.0625
    2.0000    2.2180   -0.4889    0.1250
    3.0000    2.1911   -0.1595    0.2500
    4.0000    2.1581   -0.0039    0.1250
    5.0000    2.1142   -0.0835    0.0625
    6.0000    2.0236    0.0222    0.1250
    7.0000    1.7587    0.2225    0.2500
    8.0000    1.5699    0.2473    0.1250
    9.0000    1.4333    0.0518    0.0625
   10.0000    1.2947   -0.0694    0.1250
   11.0000    1.1727   -0.1621    0.2500
   12.0000    1.0062    0.1060    0.5000
   13.0000    0.8992   -0.1121    0.1250
   14.0000    0.8463   -0.2186    0.1250
   15.0000    0.7892   -0.2114    0.2500
   16.0000    0.7654   -0.4110    0.1250
   17.0000    0.7529   -0.5741    0.1250
   18.0000    0.7430   -0.6098    0.2500
   19.0000    0.7404   -0.9247    0.0625
   20.0000    0.7391   -1.0954    0.1250
   21.0000    0.7381   -1.2459    0.2500
   22.0000    0.7379   -1.5369    0.0625
   23.0000    0.7379   -1.7207    0.1250
   24.0000    0.7378   -1.8828    0.2500
   25.0000    0.7378   -2.1907    0.0625
   26.0000    0.7378   -2.3817    0.1250
   27.0000    0.7378   -2.5165    0.2500
   28.0000    0.7378   -2.8518    0.0625
   29.0000    0.7378   -3.0489    0.1250
   30.0000    0.7378   -3.1464    0.2500
   31.0000    0.7378   -3.5115    0.0625
```

```
   32.0000    0.7378   -3.7155    0.1250
   33.0000    0.7378   -3.9017    0.1250
   34.0000    0.7378   -4.0678    0.2500
   35.0000    0.7378   -4.3798    0.0625
   36.0000    0.7378   -4.5713    0.1250
   37.0000    0.7378   -4.6980    0.2500
   38.0000    0.7378   -5.0412    0.0625


L =

   -0.0996   -1.0236    0.0171
   -0.0071    0.0428   -1.0100
    1.0129    0.0332    0.0504
   -0.0548   -0.4493   -0.7723
    0.8563   -0.3740    0.0694
    0.8356    0.0487   -0.3604
   -0.1029   -0.7227   -0.5375
    0.3221   -0.8817    0.0312
    0.4628    0.0852   -0.7838
   -0.0766   -0.6043   -0.6583
    0.4278   -0.9289    0.1229
    0.6594    0.0772   -0.6073
   -0.1088   -1.0080   -0.0174
    0.0595    0.0956   -0.9868
    0.9899    0.0072    0.0947
    0.7137   -0.2427   -0.3283
    0.2203   -0.6354   -0.4597
   -0.0847   -1.0022    0.0557
   -0.0592   -0.0113   -0.9769
    1.0034    0.0365    0.0394


phi =

    1.0000   -0.2568   -0.3216
   -0.2568    1.0000    0.3366
   -0.3216    0.3366    1.0000
```

References

Jennrich, R.I. (2001). A simple general proceedure for orthogonal rotation. *Psychometrika, 66*, 289-306.

Jennrich, R.I. (2002). A simple general method for oblique rotation. *Psychometrika, 67*, 7-19.

Matlab (1995). The MathWorks Inc., 24 Prime Park Way, Natick, MA, 01760.

Thurstone, L.L. (1947). *Multiple Factor Analysis*. Chicago: University of Chicago Press.