CS 621: Go-the-Distance 2. (An SVD Nearness Problem)

The Problem

Complete the following function so that it performs as specified.

```
function A0 = nearest(A)
% A is an n-by-n matrix
% A0 minimizes f(B) = norm(A - B,'fro') subject to the constraint that det(B) = abs(det(A))
```

Justify your method, preferably by submitting a proof.

Reduction to a Diagonal Problem

We first establish a "positive det" SVD. It diagonalizes A with orthogonal U and V matrices that have the same determinant.

Theorem

If $A \in \mathbb{R}^{n\times n}$ then there exists orthogonal U and V such that $U^TAV = S = \text{diag}(s_i)$ with $\det(U)\det(V) = 1$ and $|s_1| \geq \cdots \geq |s_n| \geq 0$.

Proof

Let $U_0AV_0 = \Sigma = \text{diag}(\sigma_i)$ be the SVD. If $\det(U_0)\det(V_0) = 1$ then set $U = U_0$, $V = V_0$ and $S = \Sigma$. Otherwise, let $\Delta = \text{diag}(\pm 1, \pm 1, \ldots, \pm 1)$ have the property that $\det(\Delta) = -1$. (This means that there are an odd number of -1 entries.) The theorem follows in this case by setting $U = U_0$, $V = V_0\Delta$, and $S = \Sigma\Delta$.□

Suppose $B_{min} \in \mathbb{R}^{n\times n}$ has positive determinant and minimizes $\| A - B \|_F$. If $U^TAV = \text{diag}(s_i)$ is a positive det SVD, then

$$\| A - B \|_F^2 \;=\; \| U^TAV - U^TBV \|_F^2 \;=\; \sum_{i=1}^{n}(s_i - \tilde{b}_{ii})^2 + \sum_{i\neq j}\tilde{b}_{ij}^2 \;\geq\; \sum_{i=1}^{n}(s_i - \tilde{b}_{ii})^2.$$

where $\tilde{B} = U^TB_{min}V$. It follows that $\tilde{B}$ must be diagonal. This shows that we can adopt the following solution framework for computing B_{min}:

Compute the SVD $A = U_0\text{diag}(\sigma_i)V_0^T$.

If $\det(U_0)\det(V_0) == 1$

  Set $B_{min} = A$ $(\det(A) >= 0)$

elseif $\sigma_n == 0$

  Set $B_{min} = A$

else

  Determine $\Delta = \text{diag}(\pm 1, \pm 1, \ldots, \pm 1)$ and positive scalars $b_1, \ldots, b_n$ so that

    (a) $\det(\Delta) = -1$

    (b) $b_1 \cdots b_n = |\sigma_1 \cdots \sigma_n|$

    (c) $\| \Sigma\Delta - \text{diag}(b_i) \|_F$ is minimized

  Set $B_{min} = U_0\text{diag}(b_i)(V_0\Delta)^T$

end

The 2-by-2 Problem

Consider the problem of minimizing $(s_1 - b_1)^2 + (s_2 - b_2)^2$ where s_1 and s_2 are given and satisfy $s_1 s_2 < 0$. The unknowns b_1 and b_2 are required to be positive and satisfy $b_1 b_2 = -s_1 s_2$. Suppose $\{b_1, b_2\}$ are feasible and consider the function

$$f(\theta) = (s_1 - \theta b_1)^2 + (s_2 - b_2/\theta)^2$$

If this is minimized by positive θ_* then $\{\theta_* b_1, b_2/\theta_*\}$ is feasible and optimal. Here is a function that does this:

```
  function [beta1,beta2,theta_star,minVal] = Best2(s1,s2,b1,b2)
% s1 and s2 are real scalars with s1*s2<0
% b1 and b2 are real scalars with b1>0, b2>0, and b1*b2 = -s1*s2
% Define f(theta) = (s1 - theta*b1)^2 + (s2 - b2/theta)^2.
% beta1 = b1*theta_star and beta2 = b2/theta_star where theta_star
% minimizes f.
% minVal = f(minVal)

  lower = .00001;
  upper = 10;
  theta_star = fminbnd(@(theta) f(theta,s1,b1,s2,b2),lower,upper);
  beta1 = b1*theta_star;
  beta2 = b2/theta_star;
  minVal = (s1-beta1)^2 + (s2-beta2)^2;

  function fVal = f(theta,s1,b1,s2,b2)
  fVal = (s1- theta*b1)^2 + (s2 - b2/theta)^2;
```

More sophisticated/intelligent values for `lower` and `upper` could be worked out. The script `ShowBest2` illustrates its use and reveals that a smaller sum-of-squares results when the negative s_i is the smaller s_i in absolute value. The script also shows that setting $b_1 = s_1$ and $b_2 = -s_2$ is *not* optimal.

Solving the General Diagonal Problem

Let $S = \Sigma\Delta = \text{diag}(s_i)$ and consider the sum of squares $(s_1 - b_1)^2 + \cdots + (s_n - b_n)^2$ which must be minimized subject to the constraints $b_i > 0$ and $b_1 \cdots b_n = |s_1 \cdots s_n|$. Suppose $s_{i_1} < 0$,$s_{i_2} < 0$, and $s_{i_3} < 0$. Note that

$$(s_{i_1} - b_{i_1})^2 + (s_{i_2} - b_{i_2})^2 + (s_{i_3} - b_{i_3})^2 > (-s_{i_1} - b_{i_1})^2 + (-s_{i_2} - b_{i_2})^2 + (s_{i_3} - b_{i_3})^2$$

This shows that if you have three negative s_i (i.e., three -1's on the diagonal of Δ) then you can reduce the sum of squares by changing two of the -1's to +1's. Conclusion: The optimizing Δ should have exactly one -1 and we know from simulations that $\Delta_{opt} = \text{diag}(1, 1, \ldots, 1, -1)$.

Consider the problem of minimizing

$$\phi_k(b) = \sum_{i=1}^{n} (s_i - b_i)^2$$

subject to the constraint that $b_i > 0,\ i = 1{:}n$ and $b_1 \cdots b_n = -s_1 \cdots s_n$. Assume that $s_1, \ldots, s_{n-1} > 0$ and that $s_n <= 0$.

Suppose b is feasible and that we wish to improve it by modifying b_i and b_j where i and j are given integers that satisfy $1 \leq i < j \leq n$. This can be done by minimizing

$$f(\theta) = (s_i - \theta b_i)^2 \ + \ (s_j - b_j/\theta)^2 \qquad \theta > 0$$

and then updating $b_i \leftarrow \theta_{opt} b_i$ and $b_j \leftarrow b_j/\theta_{opt}$. Note that `Best2` can be used for this purpose. We can repeat the process over all possible (i, j) pairs continuing until the reduction in the sum-of-squares becomes negligible...

```
  function [beta,sumVal,its] = BestBeta(s)
% s is a column n-vector that is positive except in a single component.
% beta is a positive column n-vector such that
%
%       (a)  b(1)*...*b(n) = -s(1)*...*s(n)
%       (b)  sumVal = (s(1) - b(1))^2 + ... + (s(n) - b(n)^2 is minimum

   tol = .00001;     % termination criteria; analysis would be useful
   itsMax = 20;      % Maximum number of iterations
   n = length(s);
   b = abs(s);       % Initial guess
   sumVal = sum((b-s).^2);
   sumVal1 = norm(s);

   its=0;
   while (sumVal1-sumVal>tol&& its<itsMax)
      % Continue because last iteration reduced sum-of-squares by
      % an amount more than tol
      sumVal1 = sumVal;
      % Improve over all possible (i,j) pairs
      for i=1:n-1
         for j=i+1:n
            [b(i),b(j),t0,minVal] = Best2(s(i),s(j),b(i),b(j));
         end
      end
      sumVal = sum((b-s).^2);
      its = its+1;
   end
   beta = b;
```

The script `ShowBestBeta` provides insight into its behavior.

Putting it All Together

The overall solution process is thus

```
  function A0 = CVNearest(A)
% A is an n-by-n matrix
% A0 minimizes norm(A-A0,'fro') subject to the constraint that
%   abs(det(A0)) > 0.
[n,n] = size(A);
[U,S,V] = svd(A);
if det(U*V) > 0
    % Determinant is positive. Nothing to do.
    A0 = A;
else
    s = diag(S);
    s(n) = -s(n);
    [beta,sumVal,its] = BestBeta(s);
    V(:,n) = -V(:,n);
    A0 = U*diag(beta)*V';
end
```