CS 322: Background for A6A

1. Vector Norms

Norms are used to measure the size of vectors and matrices. For a vector

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

we have these important examples:

$$\begin{aligned} \| x \|_2 &= \sqrt{x_1^2 + \cdots + x_n^2} \\ \| x \|_1 &= |x_1| + \cdots + |x_n| \end{aligned}$$

These are the *2-norm* and *1-norm* respectively. There are many other examples. One often has to "choose the right norm" in a particular application. For us that will almost always be the 2-norm or a *weighted* 2-norm:

$$\| x \|_w = \sqrt{w_1|x_1| + \cdots + w_n|x_n|}$$

Here, one chooses positive weights $w_1, \ldots, w_n$ to emphasize (or de-emphasize) the importance of particular components.

In MATLAB norm computations are easy. If `x` and `w` are vectors having the same length (and orientation) then `norm(x)`, `norm(x,1)` and `norm(x.*w)` return $\| x \|_2$, $\| x \|_1$, and $\| x \|_w$ respectively.

If the particular norm being used is obvious or unimportant, we drop the subscript. Thus, $\| x - y \|$ measures the distance between two vectors in some obvious or unimportant norm. A vector norm must satisfy three properties:

1. $\| x \| \geq 0$ with equality only if x is the zero vector.
2. $\| x + y \| <= \| x \| + \| y \|$ where x and y are vectors.
3. $\| \alpha x \| = |\alpha| \| x \|$ where α is a scalar and x is a vector.

2. Matrix Norms

Matrix norms are similar. Our first example is the *Frobenius norm.* If $A \in \mathbb{R}^{m\times n}$ then

$$\| A \|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}^2}$$

The command `norm(A,'fro')` returns the Frobenius norm of `A`. Weighting the columns in this summation produces a weighted Frobenius norm and that will be useful in some of our applications:

```
  function t = normWF(A,w)
% A is an m-by-n matrix and w is an n-vector with positive entries. Returns
% the correspondiing column weighted Frobenius norm of A. A call of the form
% normWF(A) is legal and simply sets all the weights to one.

   [m,n] = size(A);
   if nargin==1
      w = ones(n,1);
   end
   t = 0;
   for k=1:n
      t = t + w(k)*(A(:,k)'*A(:,k));
   end
   t = sqrt(t);
```

Another matrix norm is the 2-norm:

$$\| A \|_2 = \max_{\| x \|_2 = 1} \| Ax \|_2$$

The idea here is to see how much A can stretch a unit vector. The image of the unit sphere under the "map" $x \rightarrow Ax$ is an egg-shaped object–a hyperellipsoid. The 2-norm of the matrix A is just the length of the longest semiaxis.

3. Translation in 3-space

Suppose we have a bunch of 3-vectors. Let's assemble them in a 3-by-n matrix A. If we add a given 3-vector v to each vector, then the set of vectors is said to be *translated* by v. In MATLAB:

```
for j=1:n
   A(:,j) = A(:,j) + v;
end
```

This is equivalent to

```
A = A+v*ones(1,n)
```

Now suppose we are given two sets of 3-vectors $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ and that we want to translate the latter so that it is "as close as possible" to the former. Let v be the (unknown) translation vector and for $j = 1{:}n$ quantify the distance between a_j and $b_j + v$ with the 2-norm. This prompts us to minimize the function

$$\phi(v) = \sum_{j=1}^{n} \| a_j - (b_j + v) \|_2^2 = \sum_{j=1}^{n} (c_j - v)^T (v_j - v)$$

where we set $c_j = a_j - b_j$ and used the fact that $\| x \|_2^2 = x^T x$.

When trying to find a critical point (e.g., a minimum) of such a function we set its gradient to zero:

$$\nabla \phi(v) = \nabla \phi(v_1, v_2, v_3) = \begin{bmatrix} \partial\phi/\partial v_1 \\ \partial\phi/\partial v_2 \\ \partial\phi/\partial v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Now

$$\phi_j(v) \equiv (c_j - v)^T (c_j - v) = c_j^T c_j - 2v^T c_j + v^T v$$

has the property that

$$\nabla \phi_j(v) = -2c_j + 2v$$

and so

$$\nabla \phi(v) = \sum_{j=1}^{n} (-2c_j + 2v)\,.$$

Equating this to zero gives a recipe for the optimum v:

$$v_{opt} = \frac{1}{n} \sum_{j=1}^{n} c_j$$

This is just the centroid of the vectors $a_1 - b_1, \ldots, a_n - b_n$. Here is a MATLAB function that does this. The two sets of vectors are represented as a pair of 3-by-n vectors.

```
  function v = optTranslate(A,B)
%
% A and B are 3-by-n matrices.
% v is a 3-vector that minimizes
```

```
%
% || A(:,1) - (B(:,1)+v) ||^2 + ... + || A(:,n) - (B(:,n)+v) ||^2
%
  [m,n] = size(A);
  v = zeros(3,1);
  for i=1:n
     v = v + (A(:,i) - B(:,i));
  end
  v = v/n;
```

4. Orthogonal Matrices

A matrix $Q \in \mathbb{R}^{n \times n}$ is *orthogonal* if $Q^T Q = I_n$ where I_n is the n-by-n identity matrix. Here is an example:

$$Q = \frac{1}{9} \begin{bmatrix} 7 & -4 & -4 \\ -4 & 1 & -8 \\ -4 & -8 & 1 \end{bmatrix}$$

Some properties of orthogonal matrices:

(a) The transpose of an orthogonal matrix is its inverse.

(b) If $Q \in \mathbb{R}^{n \times n}$ is orthogonal then $QQ^T = I_n$.

(c) The determinant of an orthogonal matrix is ± 1 since

$$1 = \det(I_n) = \det(Q^T Q) = \det(Q^T)\det(Q) = \det(Q)^2$$

(d) Every column of an orthogonal matrix has unit 2-norm and is orthogonal to every other column, i.e., $Q(:,i)^T Q(:,j)$ is one if $i = j$ and 0 otherwise.

(e) Every row of an orthogonal matrix has unit 2-norm and is orthogonal to every other row, i.e., $Q(i,:)Q(j,:)^T$ is one if $i = j$ and 0 otherwise.

(f) The entries in an orthogonal matrix are in between -1 and 1.

(g) A vector does not change length when it is multiplied by an orthogonal matrix. This is because

$$\| Qx \|_2^2 = (Qx)^T (Qx) = (x^T Q^T)(Qx) = x^T (Q^T Q) x = x^T I_n x = x^T x = \| x \|_2^2$$

(h) Recall that if $x, y \in \mathbb{R}^n$ then $x^T y = \| x \|_2 \| y \|_2 \cos(\theta)$ where θ is the angle between x and y. The angle between two vectors does not change if they are each multiplied by an orthogonal matrix because their inner product does not change:

$$(Qx)^T (Qy) = (x^T Q^T)(Qy) = x^T y$$

(i) The Frobenius norm of a matrix $A \in \mathbb{R}^{n \times n}$ does not change if it is multiplied by an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ for if $C = QA$ then

$$\| C \|_F^2 = \sum_{j=1}^{n} \| C(:,j) \|_2^2 = \sum_{j=1}^{n} \| QA(:,j) \|_2^2 = \sum_{j=1}^{n} \| A(:,j) \|_2^2 = \| A \|_F^2$$

(j) If $Q_1, Q_2 \in \mathbb{R}^{n \times n}$ are each orthogonal, then so is their product:

$$(Q_1 Q_2)^T (Q_1 Q_2) = (Q_2^T Q_1^T)(Q_1 Q_2) = Q_2^T (Q_1^T Q_1) Q_2 = Q_2^T Q_2 = I_n$$

5. Rotations

All 2-by-2 orthogonal matrices have the form

$$Q_1 = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \text{or} \qquad Q_2 = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix}$$

for some angle θ. The example Q_1 is a *rotation* and the example Q_2 is a *reflection*. Rotations have determinant 1 and reflections have determinant -1.

In general, we will say that an orthogonal matrix Q is a *rotation* if $\det(Q) = 1$. Products of rotations are rotations.

We will be concerned with the rotation of coordinate systems in 3-space. In this context, 3-by-3 rotation matrices are important. It turns out that any such matrix is a product of the form

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_3) & \sin(\theta_3) \\ 0 & -\sin(\theta_3) & \cos(\theta_3) \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & \sin(\theta_2) & 0 \\ -\sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & \sin(\theta_1) \\ 0 & -\sin(\theta_1) & \cos(\theta_1) \end{bmatrix}.$$

You should regard the factors on the right as "simple" rotation matrices in that each leaves one coordinate "alone" when it is applied. Note that three angles characterize a 3-by-3 rotation. Here is a MATLAB function that generates a random 3-by-3 rotation matrix:

```
  function Q = randRot
% Generates a random 3-by-3 rotation matrix.
theta = 2*pi*randn(3,1);
c = cos(theta);
s = sin(theta);
Q1 = [ 1   0     0   ; ...
       0  c(1)  s(1) ; ...
       0 -s(1)  c(1) ];

Q2 = [c(2)  0   s(2)   ; ...
       0    1    0     ; ...
     -s(2)  0   c(2) ];

Q3 = [ 1    0    0    ; ...
       0   c(3) s(3) ; ...
       0  -s(3) c(3) ];

Q = Q3*Q2*Q1
```

We mention that there are other ways that a 3-by-3 rotation can be represented:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_3) & \sin(\theta_3) \\ 0 & -\sin(\theta_3) & \cos(\theta_3) \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

6. The Trace of a Matrix

The *trace* of a square matrix is the sum of its diagonal entries:

$$G \in \mathbb{R}^{n\times n} \qquad \Rightarrow \qquad \mathrm{tr}(G) = \sum_{i=1}^{n} g_{ii}$$

In MATLAB, the value of `sum(diag(G))` is the trace of the matrix `G`. Properties of the trace include

(a) If $G \in \mathbb{R}^{m\times n}$ then $\| G \|_F^2 = \text{tr}(G^TG)$:

$$\text{tr}(G^TG) = \sum_{k=1}^{n}(G^TG)_{kk} = \sum_{k=1}^{n}\left(\sum_{i=1}^{m} g_{ik}^2\right) = \| G \|_F^2.$$

(b) If $G \in \mathbb{R}^{n\times n}$, then $\text{tr}(G^T) = \text{tr}(G)$.

(c) If $F, G \in \mathbb{R}^{n\times n}$, then $\text{tr}(F+G) = \text{tr}(F) + \text{tr}(G)$.

(d) If $F, G \in \mathbb{R}^{n\times n}$, then $\text{tr}(FG) = \text{tr}(GF)$.

(e) If $G, Z \in \mathbb{R}^{n\times n}$ and Z is orthogonal, then $\text{tr}(Z^TGZ) = \text{tr}(G)$:

$$\begin{aligned}
\text{tr}(Z^TGZ) &= \sum_{k=1}^{n}(Z^TGZ)_{kk} = \sum_{k=1}^{n} Z(:,k)^TGZ(:,k) = \sum_{k=1}^{n}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} z_{ik}g_{ij}z_{jk}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\left(\sum_{k=1}^{n} z_{ik}z_{jk}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\delta_{ij} = \sum_{i=1}^{n} g_{ii}
\end{aligned}$$

where $\delta_{ij} = 1$ if $i = j$ and zero otherwise.

7. The Singular Value Decomposition (SVD)

If $A \in \mathbb{R}^{n\times n}$ then there exist orthogonal matrices $U, V \in \mathbb{R}^{n\times n}$ such that

$$U^TAV = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. This is called the singular value decomposition (SVD) of A. The σ_i's are the *singular values* and the columns of U and V are the left and right singular vectors respectively.

MATLAB has a built-in function that can be used to compute the singular value decomposition. The script

```
A = [4 -1 3; 2 7 -5 ; -6 2 1]
[U,Sigma,V] = svd(A)
```

computes the SVD $U^TAV = \Sigma$ where

$$A = \begin{bmatrix} 4 & -1 & 3 \\ 2 & 7 & -5 \\ -6 & 2 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} -0.2755 & 0.5212 & -0.8078 \\ 0.9576 & 0.2230 & -0.1826 \\ 0.0850 & -0.8238 & -0.5605 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 9.0422 & 0 & 0 \\ 0 & 7.5076 & 0 \\ 0 & 0 & 2.6221 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.0336 & 0.9955 & -0.0890 \\ 0.7905 & -0.0809 & -0.6070 \\ -0.6115 & -0.0500 & -0.7897 \end{bmatrix}$$

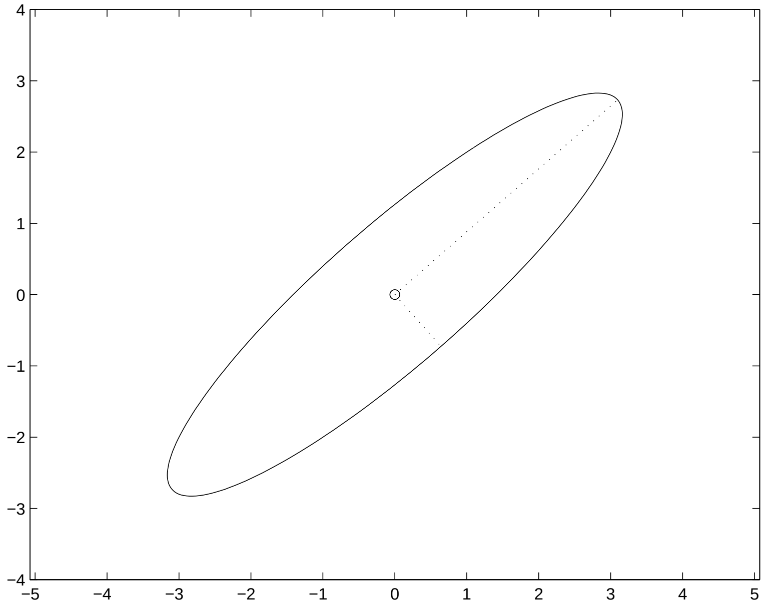

FIGURE 1 SVD Geometry

There is a nice geometry associated with the SVD which is most simply discussed when $n = 2$. The map $x \rightarrow Ax$ maps the unit circle onto an ellipse: The singular values are the lengths of the semiaxes. Moreover, the dotted vectors in the figure are $AV(:,1) = \sigma_1 U(:,1)$ and $AV(:,2) = \sigma_2 U(:,2)$ respectively. Here is the script that produces the figure:

```
% Script File ShowSVD
% Show what a 2-by-2 matrix does to the unit sphere.

A = [1 3; 2 2];

% Generate 200 unit vectors
t = linspace(0,2*pi,200);
P = [cos(t);sin(t)];

% The columns of B are the images of these unit vectors.

B = A*P;

% Depict the ellipse and its semiaxes

plot(B(1,:),B(2,:))
axis([-4 4 -4 4])
axis('equal')
hold on
plot(0,0,'o')
[U,S,V] = svd(A);
plot([0 S(1,1)*U(1,1)],[0 S(1,1)*U(2,1)],':r')
plot([0 S(2,2)*U(1,2)],[0 S(2,2)*U(2,2)],':r')
hold off
```

8. Finding Best Rotations with the SVD

Consider the problem of finding a 3-by-3 orthogonal matrix that minimizes $\| A - QB \|_F$ where $A, B \in \mathbb{R}^{3\times n}$ are given. Using properties of the trace we have

$$\begin{aligned}
\| A - QB \|_F^2 &= \text{tr}\left((A-QB)^T(A-QB)\right) \\
&= \text{tr}\left((A^T - B^TQ^T)(A-QB)\right) \\
&= \text{tr}\left(A^TA - B^TQ^TA - A^TQB + B^TQ^TQB\right) \\
&= \text{tr}(A^TA) - \text{tr}(B^TQ^TA) - \text{tr}(A^TQB) + \text{tr}(B^TQ^TQB) \\
&= \text{tr}(A^TA) - \text{tr}(A^TQB) - \text{tr}(A^TQB) + \text{tr}(B^TB) \\
&= \| A \|_F^2 + \| B \|_F^2 - 2\text{tr}(A^T(QB)) \\
&= \| A \|_F^2 + \| B \|_F^2 - 2\text{tr}(QBA^T)
\end{aligned}$$

Thus, the problem of choosing a 3-by-3 rotation Q that minimizes $\| A - QB \|_F$ is equivalent to the problem of *maximizing* $\text{tr}(QBA^T)$ over all 3-by-3 orthogonal Q.

Suppose

$$U^TCV = \Sigma = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}$$

is the SVD of the 3-by-3 matrix $C = BA^T$. It follows that

$$\begin{aligned}
\text{tr}(QC) &= \text{tr}((V^TQ(UU^T)CV)) \\
&= \text{tr}((V^TQU)(U^TCV)) \\
&= \text{tr}(\tilde{(Q)}\Sigma) = \tilde{q}_{11}\sigma_1 + \tilde{q}_{22}\sigma_2 + \tilde{q}_{33}\sigma_3
\end{aligned}$$

where $\tilde{Q} = V^TQU$. Note that $\tilde{Q}$ is orthogonal. The entries in an orthogonal matrix are always between -1 and 1 and so the above three-term sum is maximized if $\tilde{q}_{11} = \tilde{q}_{22} = \tilde{q}_{33} = 1$. Since $\tilde{Q}$'s columns have unit 2-norm, it follows that $\tilde{Q} = I_3$, i.e., VU^T is the optimizing Q.

To sum up, $\| A - QB \|_F$ *is minimized by setting* $Q = Q_{opt} = VU^T$ *where* $U^T(BA^T)V = \Sigma$ *is the SVD of* BA^T.

However, we are interested in minimizing $\| A - QB \|_F$ over all 3-by-3 rotations. If $\det(VU^T) = \det(U)\det(V)$ $= 1$, then the procedure outlined above renders the optimum rotation. But if $\det(U)$ and $\det(V)$ are opposite in sign, then $\det(VU^T)$ is negative and we have a problem. To rectify this we return to

$$\| A - QB \|_F^2 = \| A \|_F^2 + \| B \|_F^2 - 2\text{tr}(QC) \qquad C = BA^T$$

and ask how we can maximize $\text{tr}(QC)$ given that Q is a *rotation*. As we have said, we cannot simply compute the SVD $U^TCV = \Sigma$ and set $Q_{opt} = VU^T$ since this matrix has a negative determinant. Note however, that if

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}^T C \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

is the SVD, then

$$\begin{bmatrix} u_{11} & u_{12} & -u_{13} \\ u_{21} & u_{22} & -u_{23} \\ u_{31} & u_{32} & -u_{33} \end{bmatrix}^T C \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & -\sigma_3 \end{bmatrix}.$$

Set

$$U_- = \begin{bmatrix} u_{11} & u_{12} & -u_{13} \\ u_{21} & u_{22} & -u_{23} \\ u_{31} & u_{32} & -u_{33} \end{bmatrix}$$

and

$$\Sigma_- = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & -\sigma_3 \end{bmatrix}$$

and note that $U_-^T C V = \Sigma_-$. Thus,

$$\mathrm{tr}(QC) = \mathrm{tr}(VQU_-U_-^TCV) = \mathrm{tr}(\tilde{Q}\Sigma_-) = \tilde{q}_{11}\sigma_1 + \tilde{q}_{22}\sigma_2 - \tilde{q}_{33}\sigma_3$$

If we parameterize $\tilde{Q}$ as follows

$$\tilde{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_3) & \sin(\theta_3) \\ 0 & -\sin(\theta_3) & \cos(\theta_3) \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & \sin(\theta_2) & 0 \\ -\sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & \sin(\theta_1) \\ 0 & -\sin(\theta_1) & \cos(\theta_1) \end{bmatrix}.$$

then

$$\begin{aligned} \mathrm{tr}(\tilde{Q}\Sigma_-) &= \tilde{q}_{11}\sigma_1 + \tilde{q}_{22}\sigma_2 - \tilde{q}_{33}\sigma_3 \\ &= c_2\sigma_1 + (c_1c_2c_3 - s_1s_3)\sigma_2 + (-s_1c_2s_3 + c_1c_3)\sigma_3 \\ &= f(\theta_1, \theta_2, \theta_3), \end{aligned}$$

a function of three variables. Using elementary calculus and the fact that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ it is possible to show that the optimum $\tilde{Q}$ is I_3. It follows that

$$Q_{opt} = VU_-^T.$$

9. The Overall Method

Given $A, B \in \mathbb{R}^{3\times n}$ our aim is to translate and rotate B so that it matches A as much as possible. In particular, we are looking for a 3-vector v and a rotation Q so that

$$\| A - Q(B + ve^T) \|_F = \min$$

Here, $e \in \mathbb{R}^n$ is the vector of all ones. Note that

$$\| A - Q(B + ve^T) \|_F = \| (Q^TA - B) - ve^T \|_F$$

From §3 we know that for any Q the optimum v is given by $(Q^TA - B)e/n$. Thus, our goal is to choose a rotation Q so that

$$\begin{aligned} \| (Q^TA - B) - ve^T \|_F &= \| (Q^TA - B) - (Q^TA - B)ee^T/n \|_F \\ &= \| (A - QB) - (A - QB)ee^T/n) \|_F \\ &= \| \tilde{A} - Q\tilde{B} \|_F \end{aligned}$$

is minimized where

$$\tilde{A} = A(I - ee^T/n) \qquad \text{and} \qquad \tilde{B} = B(I - ee^T/n.$$

Note that the columns of these two matrices have centroids at 0, i.e., $\tilde{A}e/n = \tilde{B}e/n = 0$.

So the overall process involves applying the ideas of §8 to $\tilde{A}$ and $\tilde{B}$ to get the optimum rotation Q_{opt} and then setting $v_{opt} = (Q_{opt}^TA - B)e/n$