Homework 2

6.s955 Applied Numerical Analysis

Pitipat Wongsittikan

September 29^{th}, 2023

Problem 1

a. Given a random vector v, we can write v in terms of null space of A^T (x_0), A, and some vector y

$$v = x_0 + Ay \tag{1}$$

multiply by P_0 we will have,

$$\begin{aligned} P_0 v &= (I - QQ^T)v && (2)\\ &= (I - QQ^T)(x_0 + Ay) && (3)\\ &= (I - QQ^T)(x_0 + QRy) && (4)\\ &= (x_0 + QQ^T x_0 + QRy - QQ^T QRy) && (5)\\ &= (x_0 + QQ^T x_0 + QRy - QRy) && (6)\\ &= (x_0 + QQ^T x_0) && (7) \end{aligned}$$

Since x_0 is a null space of A^T, we know that

$$\begin{aligned} A^T x_0 &= 0 && (8)\\ (QR)^T x_0 &= 0 && (9)\\ R^T Q^T x_0 &= 0 && (10) \end{aligned}$$

Since R is invertible, $Q^T x_0$ will have to be 0. Therefore,

$$\begin{aligned} x_0 + QQ^T x_0 &= x_0 + 0 && (11)\\ &= x_0 && (12) \end{aligned}$$

Since the multiplication of P_0 to a random vector v is the null space x_0, we have proved that P_0 is the projection matrix to the null space of A^T

b. First, we go from Cholesky's to QR.
From Cholesky's, we know that $A^TA = LL^T$. Therefore,

$$AL^{-T} = A^{-T}L \tag{13}$$

We take a dot product of AL^{-T}

$$(AL^{-T})^T(AL^{-T}) = L^{-1}A^TAL^{-T} \tag{14}$$
$$= L^{-1}(LL^T)L^{-T} \tag{15}$$
$$= I \tag{16}$$

Therefore, AL^{-T} is an orthogonal matrix. We can define $Q = AL^{-T}$ in QR factorization, and will will have

$$A = QR \tag{17}$$
$$A = AL^{-T}R \tag{18}$$
$$R = L^T \tag{19}$$

Therefore, if we want to go from Cholesky's to QR, we can define $Q = AL^{-T}$ and an upper triangular matrix $R = L^T$

From QR factorization to Cholesky.
We know that $A = QR$. Therefore, A^TA will take the form of

$$A^TA = (QR)^T(QR) \tag{20}$$
$$= R^TQ^TQR \tag{21}$$
$$= R^T(I)R \tag{22}$$
$$= R^TR \tag{23}$$

Since R is an upper triangular matrix, R^T is a lower triangular matrix.
Therefore,

$$A^TA = R^TR \tag{24}$$
$$= LL^T \tag{25}$$

Which is the form of Cholesky's factorization.
Therefore, to go from QR factorization to Chlesky's we can define L in Cholesky's as R^T

c. Since $A = uv^T$ where $u^Tv = 1$, we found that $A^2 = (uv^T)(uv^T) = u(u^Tv)^Tv^T = u(1)v^T = uv^T = A$.Therefore, $A^k = A$.

This means, in the power iteration process, the left-hand-side, where we multiply a random vector v with A's, it wont change from Av. Therefore, after iteration 2, we will found that the value does not change from iteration 1 and we can stop (criteria satisfied). Therefore, we will do 2 iterations. [Note: We can actually got the answer from iteration 1, but the second iteration is to make sure the value does not change]

Problem 2

A distance preserving function means the distance between 2 points before and after applying the function will be the same. Therefore,

$$||f(p) - f(q)||_2^2 = ||p - q||_2^2 \tag{26}$$

Let's try q to be the origin, $q = (0, 0, 0)$

$$||f(p) - f(q)||_2^2 = ||p||_2^2 \tag{27}$$
$$||f(p)||^2 - 2f(p)f(q) + ||f(q)||^2 = ||p||^2 \tag{28}$$

We can see that, if $f(x) = Rx$ for some orthogonal R we the last equation will holds true.

$$||f(p)||^2 - 2f(p)f(q) + ||f(q)||^2 = ||p||^2 \tag{29}$$
$$||Rp||^2 - 0 + 0 = ||p||^2 \tag{30}$$
$$||p||^2 = ||p||^2 \tag{31}$$

Now, we want to show that $f(x) = Rx$ preserves dot product.... (not done) Note, I have a version where I substitute f(x) = Rx + b, but it seems that that's a wrong way to do this. I can go so far as getting f(x) = Rx , but not the b term.

Problem 3

a. From problem 2, we know that $f(x) = Rx + b$ for $R^T R = I$
Therefore,

$$f(q_i) - f(p_i) = Rq_i + b - Rp_i - b \tag{32}$$
$$= Rq_i - Rp_i \tag{33}$$
$$= R(q_i - p_i) \tag{34}$$

b. Define $e_j = q_j - p$ and $e'_j = q'_j - p'$, we will have,

$$\sum_j w_{ij}||Re_j - e'_j||_2^2 = \sum_j w_{ij}(||Re_j||^2 - 2Re_j.e'_j + ||e'_j||_2^2) \tag{35}$$
$$= \sum_j w_{ij}(||e_j||^2 - 2Re_j.e'_j + ||e'_j||_2^2) \tag{36}$$

Since this is a minimization problem with R, we can only focus on terms with R.
The problem becomes,

$$\min \sum_j w_{ij}(-2Re_j.e'_j) \tag{37}$$

Which is

$$\max \sum_j w_{ij} R e_j . e'_j \tag{38}$$

$$\tag{39}$$

We will find the maximum using Trace property, therefore, we will rearrange the term into a form of Tr()

$$\sum_j w_{ij} 2 R e_j . e'_j = Tr(R \sum w_{ij} e_j e_j'^T) \tag{40}$$

we can do a singular value decomposition of the term $w_i j e_j e'_j = U \Sigma V^T$. We will have

$$Tr(R \sum w_{ij} e_j e_j'^T) = Tr(R U^T) \tag{41}$$

$$= Tr(V^T R U \Sigma) \tag{42}$$

Since Σ is a psd matrix, we know that $Tr(\Sigma) \geq Tr(M\Sigma)$ for any orthogonal M.
Therefore, we must make the front term become I,

$$V^T R U = I \tag{43}$$

$$R = V^{-T} U^{-1} \tag{44}$$

Since U and V are unitary matrices,

$$R = V U^T \tag{45}$$

With this formulation, we can solve for R by doing a singular value decomposition to the $w_{ij} e_j e_j'^T$ term to get U and V, then $R = V U^T$.

c. see the attached code.

Problem 4

a. pi and p'i are vectors of the vectices of interest before and after the deformation.
pj and p'j are other vectors (not the current sum) before and after of the deformation.
wij is weight of each pair vertices.
Ri is rotational matrix
S is subset of the vertices that were constrained.
Pbar i is a given moved positions (constraints)

The objective function is to minimize the difference in vertices position after deformation.
We could use this solver in 3D problems when we want to displace vertices around with simple to implement and inexpensive calculations.

b. We let $w_i = 1$. The derivative will only concerns terms with i and j that's connects to i. Therefore, we can break the given term into 2 parts, parts that vertex i is in the front part (connected to many j's) and vertex i is the back part (connected to a vertex j).

$$= \frac{\partial}{\partial p'_i}(\sum w_{ij}||p'_i - p'_j - R_i(p_i - p_j)||^2 + \sum w_{ij}||p'_j - p'_i - R_j(p_j - p_i)||^2) \tag{46}$$

$$= (\sum 2w_{ij}(p'_i - p'_j - R_i(p_i - p_j)) + \sum -2w_{ij}(p'_j - p'_i - R_j(p_j - p_i))) \tag{47}$$

$$= \sum 4w_{ij}((p'_i - p'_j) - \frac{1}{2}(R_i + R_j)(p_i - p_j)) \tag{48}$$

c. To minimize the terms, we set the derivative to 0 and rearrange the terms.

$$\sum w_{ij}(p'_i - p'_j) = \sum \frac{1}{2}(R_i + R_j)(p_i - p_j)) \tag{49}$$

d. We can define a matrix A and B to solve the equation as follows.

$$A_{ii} = \sum_{j=1}^{nv} w_{ij} \tag{50}$$

And when $i \neq j$

$$A_{ij} = -w_{ij}, \tag{51}$$

$$\tag{52}$$

Also for B

$$B_i = \sum_{j=1}^{nv} \frac{w_{ij}}{2}(R_i + R_j)(p_i - p_j) \tag{53}$$

With this formulation, we will have the final equation as

$$\sum_j w_{ij}(p'_i - p'_j) = \sum_j \frac{w_{ij}}{2}(R_i + R_j)(p_i - p_j) \tag{54}$$

$$AP' = B \tag{55}$$

where P; is a nv by 3 matrix containing rows of pi

e. see the attached code.

Problem 5

see the attached code.

Problem 6

a. We can set stopping criteria such as the change in norm of the deformation vector between successive iterations. If it falls under some small number, we stop. We can also do the same thing with R, but that involve many 2d matrices, which is harder to find the norm compared to the deformation, which is also intuitive in terms of the convergence of our result.

b. We could vectorize every loop. This would the scipt much faster.

c. We can multiply the final result by A^T, now, our factorization will be more efficient since we can do Cholesky factorization.

d. -