Sample questions for a final. The final includes 3 questions only.

1. The $C_{\alpha}$ coordinates of four protein structures are in the vectors $X_1, X_2, X_3, X_4$. The lengths of all four proteins are the same. All the geometric centers are at (0,0,0). The task is to find a single rotation matrix $U$ that will minimize the sum of the six distances $D_{ij}^2 = \text{norm}(X_i - UX_j)^2$ minimize $\sum_{i<j} D_{ij}^2(U)$ as a function of the rotation matrix $U$.

a. Write the target function to be minimized, and its derivatives with respect to $u_{ij}$ -- the matrix element of the rotation matrix.

SOLUTION

$$F = \sum_{p<q} \sum_{n} (U_{n}^{p} - r_{n}^{q})^2 + \sum_{i,j} (\sum_{k} u_{ik} u_{kj} - \delta_{ij})$$

And after a page of entertaining algebra, we find:

$$\frac{1}{2} \frac{\partial F}{\partial u_{ab}} = \sum_{i} u_{ij} \left( \lambda_{ij} + \sum_{p<q} \sum_{n} r_{nj}^{p} r_{nb}^{p} - \sum_{p<q} \sum_{n} r_{nb}^{q} r_{nb}^{p} \right)$$

b. Discuss the similarities and differences of the present problem compared to the problem of overlapping only two structures. Based on the similarities, suggests a solution to the rotation matrix.

SOLUTION This problem is very similar to the standard two-structure overlap problem studied in Lecture Notes 4 (Lagrange multipliers and optimal rotations). Instead of one pair of structures, we have six. The formula above for $\frac{dF}{dt_{ab}}$ shows that we once again have a matrix equation of the form $U(S + \Lambda) = R$. But here the matrices $S$ and $R$ are in fact sums of matrices $S_{p}$ and $R_{pq}$, where $p$ and $q$ index the four proteins. The equation can be solved using the same techniques: multiply it by its transpose to get rid of $U$, compute the svd of $R$, then expand the sum of distances and reduce it, using the unitarity of $U$, to a sum of coordinates (squared) and singular values.

2. Let $T$ be an $n \times n$ symmetric matrix. We are interested in the minimum of the function $F(x) = x^T T x$ subject to the constraint that $(x^T x) = 1$, where $x$ is a vector of length $n$. The elements of the vector are the unknowns. A possible algorithm is the following: (a) perform one (small) minimization step to obtain $x'$ and “re-normalize” $x = \frac{x'}{\sqrt{(x'^T x')}}$ for the next optimization step. Write a steepest descent MATLAB program that minimizes $F(x)$ with the above constraint. Why is it necessary to have the constraint? What kind of a smoothing algorithm will you use on this function?
The constraint is necessary because the matrix $T$ is not known to be positive definite and hence $F$ may not have an unconstrained minimum. We can use a steepest-descent-like algorithm by taking small steps (i.e., steps whose size is much smaller than 1) off the unit sphere $\|x^2\|=1$ in the direction of the negative gradient, then renormalizing to hop back on the unit sphere. Because $T$ is symmetric, the gradient takes the simple form $g(x) = 2Tx$. Here is the algorithm:

```matlab
function xnext = steep_constraint(T, x0)
    if norm(x0) ~= 1
        error('x0 must have unit norm');
    end
    xprev = 2 * x0;
    xnext = x0;
    while norm(xnext - xprev) > 1/4096
        xprev = xnext;
        g = 2 * T * xprev; % gradient (factor of 2 unnecessary)
        g = g / (32 * norm(g)); make step size small (1/32)
        xnext = xprev - g; % step off unit sphere against gradient
        xnext = xnext / norm(xnext); % normalize to hop back on
    end

You can try running this algorithm on the matrix $T = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$. What minima do you expect from this matrix? Does the algorithm find them?

3. Starting with random numbers sampled uniformly between 0 and 1 generate random numbers from the probability density $p(x) = \sqrt{x}$ (the probability of observing $x$ between $x$ and $x+dx$ is $p(x)dx$).

The following Matlab function produces random numbers from the desired distribution. It uses the reject/accept method. Call it repeatedly to generate a sequence from the distribution.

```matlab
function x = sqrtprob()
    x = 0;
    y = 1;
    while y > sqrt(x)
        x = rand;
        y = rand;
    end
```
4. Starting with random numbers sampled uniformly between 0 and 1 suggest an algorithm that produces random numbers sampled from the probability density

\[ p(x) = \left( \frac{2}{b^2} \right) x, \quad x \in [0,b] \]

**SOLUTION** The following Matlab code solves the problem:

```matlab
function x = linprob(b)
    x = -1;
    y = 0;
    while y > x
        x = rand;
        y = rand;
    end
    x = b * x;
end
```

To drive it, run:

```matlab
n = 10000;
b = 100;
prob = zeros(1,b);
for i=1:n
    k = ceil(linprob(b));
    prob(k) = prob(k) + 1;
end
prob = prob/n;
```

If we then enter `plot(prob)`, we see a (noisy) linear probability distribution whose maximum value is 0.02, i.e., \(2/b\).

5. Outline how can you use the accept/reject sampling procedure to estimate the number \(\pi\).

**SOLUTION** Use Matlab’s `rand` to sprinkle the square \([-1,1] \times [-1,1]\) with \(N\) points. Count the number of points \(M\) that lie within the unit circle. The ratio \(M/N\) approximately equals the ratio of the area of the square to the circle, \(M/N = \pi/4\). So \(\pi = 4M/N\). The result becomes exact in the limit of large \(N\).

6. \(A\) is a \(n \times n\) symmetric and positive definite matrix (all the eigenvalues are real and larger than zero). It is also highly sparse (most of the elements are zero). Outline an efficient minimization algorithm to find the lowest eigenvalue of \(A\). Explain your choices.

**SOLUTION** Expressing \(x\) as an expansion in terms of eigenvectors shows that the smallest eigenvalue is also the minimum of the function \(f(x) = (x'Ax)/(x'x)\). The following code uses this observation to solve the problem. The actual function \(f\) is given by:
function z = least_eig_func(x)
global a;
    z = (x'*a*x) / (x'*x);

To minimize, run:

global a;
a = [4 2 0; 2 5 3; 0 3 6]; % sample spd matrix
vmin = fminunc('least_eig_func', [1 1 1]);

The actual eigenvalue is got by evaluating f:

least_eig_func(vmin) \rightarrow 1.45163408330922

This value compares very well against the eigenvalue obtained using \texttt{eig(a)}.

7. Derive a lower bound to the number of possible alignments of the sequence $A$ length $n$ and the sequence $B$ length $m$.

\textbf{Solution:} This is following the material in the lecture notes. The number of alignments of two sequences of lengths $m$ and $n$, $N(n,m)$, follows the recursion (we include redundant alignments):

$$N(n,m) = N(n-1,m) + N(n,m-1) + N(n-1,m-1)$$

A lower bound is provided by the expression

$$N(n,m) > \bar{N}(n,m) = N(n-1,m) + N(n,m-1), \text{ and } \bar{N}(n,m) = \binom{n+m}{m}$$

8. We wish to find a zero of the scalar product:

$$A' \cdot X = 0 \quad X = (x_1, x_2, ..., x_n) \quad A = (a_1, a_2, ..., a_n).$$

Formulate a procedure to find a solution using the conjugate gradient algorithm.

\textbf{Solution:} Conjugate gradient is formulated to find minima and not zeroes of a function. However, if we square the scalar product to have $F(X) = X'AA'X$, which is a function that has a minimum for $F(X) = 0$ which is also the solution that we seek; when $F(X_{\text{min}}) = 0$ also $A'X = 0$. We can therefore minimize the function $F(X)$ using the conjugate gradient algorithm.

9. In the class we plotted the protein chain using a cubic spline interpolation that kept the second derivative continuous. Suppose we care only about the first derivatives. How would you change the interpolation?

\textbf{Solution:} This is taken directly from the lecture notes. Below I copied the relevant notes

Consider the edges of the interval that we wish to fit ($x_L$ and $x_R$ - two points only of the complete data points). We write the polynomial $q(x)$ interpolating between the two points as:
\[
q(x) = a + b(x - x_L) + c(x - x_L)^2 + d(x - x_L)^2(x - x_R)
\]

The parameters \(a - d\) are determined from the conditions that the function and the first derivative will be continuous at the edges of the interval (when we patch additional polynomials in between). There are four conditions that are sufficient to determine the four parameters (function and first derivative of the function at the two end points).

The set of linear equations that we need to solve is

\[
\begin{align*}
a &= y_L \\
b &= y'_L \\
a + b \cdot \Delta x + c \cdot \Delta x^2 &= y_R \\
b + 2 \cdot c \cdot \Delta x + d \Delta x^2 &= y'_R \\
(\Delta x &= x_R - x_L)
\end{align*}
\]