# 18.409 Problem Set 3

Jonathan Kelner

December 1, 2009

## How this Problem Set Works

Just like the first problem set, I'll put some hints online a few days after I post the problem set. Use as many of the hints as you like; just note in your solution whether you are doing so. Also, recall that you need not do all of the problems on the problem set. Only do as many as is helpful for you to learn the material.

As was the case last time, many of these problems have solutions that can easily be found by web searching. Please don't do that. It's not very nice.

Note: Problems 3 and 4 relate to material that will be covered in lecture this coming Thursday.

## Iterative Methods for Linear Algebra

### Problem 1: Computing Eigenvalues with the Power Method

In this question, we'll examine the power method, which is a common technique used to approximately compute the biggest eigenvalue of a matrix and its corresponding eigenvector.

Let $M$ be a positive definite symmetric matrix with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $v_1, \ldots, v_n$ (normalized to be on the unit sphere), and suppose that $\lambda_{n-1} \neq \lambda_n$. Let $x_0 \in \mathbb{R}^n$ be a uniformly random vector of norm 1. Inductively define

$$x_{i+1} = \frac{Ax_i}{||Ax_i||}.$$

**(a)** Show that the $x_i$ converge to $v_n$ with high probability over the choice of $x_0$. Are there any starting vectors $x_0$ for which this method fails? What happens if $\lambda_{n-k} = \cdots = \lambda_n$?

**(b)** How quickly does the power method converge? That is, approximately how many steps does it take before $||v_n - x_i||_2 \leq \epsilon$ for some given $\epsilon > 0$ (and on which of the given parameters and variables does this rate depend)?

**(c)** Suppose that you just want to know the largest eigenvalue of $M$ and don't care about the corresponding eigenvector. Describe how to use a method like this to find it, and give a bound on how many iterations you need to get an estimate within a factor of $1 \pm \epsilon$ of the correct answer with high probability.

**(d)** What (if anything) changes about the above analysis if $M$ is allowed to have negative eigenvalues? What about if $M$ is not required to be symmetric?

**(e)** How might you use a procedure like this to compute the *smallest* eigenvalue of $M$ (in absolute value)? There are two answers to this (that I see). One of them only works when I promise you that all of the eigenvalues of $M$ are positive, whereas the other works for any symmetric $M$ but is computationally more difficult.

**(f)** Design a procedure like the method from part (d) that computes the eigenvalue of $M$ that is closest to some given value $\lambda$. How quickly does it converge? You may assume that you can quickly solve linear systems of the form $Mx = b$.

## Problem 2: Speeding up Eigenvalue Computation and the Lanczos Method

In this problem, we'll see how to do better than the power method under certain circumstances. Everything below will use the notation set forth in problem 1.

**(a)** The eigenvector of $A$ corresponding to the smallest eigenvalue is the unit vector that minimizes the quadratic form $x^T Ax$ (with an analogous statement for the biggest eigenvalue too). We showed in class how to minimize a quadratic form using the conjugate gradient method. Why can't we just use this exact algorithm to find the biggest/smallest eigenvalue?

**(b)** Suppose that I promised you in advance that $\lambda_n = 1$ and that all of the other eigenvalues lie in some interval $[\lambda_{\min}, \lambda_{\max}]$, with $\lambda_{\min} > 0$. Describe an iterative procedure to obtain a good approximation of $v$ that converges significantly faster than the power method and uses the word "Chebyshev."[1] How quickly can you make this converge?

**(c)** In this and the following parts, we will develop the Lanczos method, which is, to some extent, the proper analogue of conjugate gradient for computing eigenvalues. As with conjugate gradient, Lanczos will perform as well as the Chebyshev construction, but without requiring you to know in advance what $\lambda_{\min}$ and $\lambda_{\max}$ are.

For a vector $x$, let $K(A, x, k) = \text{span}(x, Ax, A^2x, \ldots, A^{k-1}x)$. Show how to use a procedure like the conjugate gradient method to produce an orthonormal basis for $K(A, x, k)$. Note that I am asking for an actual orthonormal basis, not an $A$-orthonormal basis. You should do this iteratively, producing a new vector $q_i$ in each step that is orthogonal to those produced in previous steps. As in conjugate gradient, you should be able to express $q_{i+1}$ using a recurrence that involves only the vectors $q_i$, $q_{i-1}$, and $Aq_i$.

**(d)** Show how to use the recurrence and the vectors from the previous part to find a tridiagonal matrix $T$ (i.e., a matrix with nonzero terms only on the diagonal and the off-diagonals directly above and below) and an orthogonal matrix $Q$ such that $Q^T AQ = T$. Deduce that we can find the eigenvalues and eigenvectors of $A$ by finding the eigenvalues and eigenvectors of $T$. Why should it be any faster to find approximate eigenvalues of $T$ than it is to find those of $A$?

**(e)** Suppose that we just want to approximately find the largest and smallest eigenvalues of $A$, and that we stop the Lanczos method after $t < n$ iterations. Show how to get a good estimate by finding the maximum/minimum eigenvalues of a $t \times t$ submatrix of the tridiagonal matrix from part (c) (or slight variant thereof, if you like) that we have already computed. Why should this be a good estimate? If we apply it to the problem in part (b), how will it compare to the answer you obtained using variants of Chebyshev polynomials?

## Problem 3: Reducing Diagonally Dominant Linear Systems to Laplacians

Let $M$ be any $n \times n$ symmetric diagonally dominant matrix with all positive diagonal entries. Our goal is to solve $Mx = b$. We noted in class that, if all of the off-diagonal entries of $M$ are nonpositive, $M$ can be thought of as the Laplacian of some weighted graph (possibly with self-loops), and we can use the preconditioning techniques that we've been discussing.

[1] Note: The following is not the sort of solution that I have in mind: "Compute the eigenvalues using an exact method in one iteration. Chebyshev."

Suppose now that I give you a black box for solving linear systems of the form $Lx = b$, where $L$ is the Laplacian of a weighted graph. Show how to use this to solve $Mx = b$ for any symmetric, diagonally dominant $M$, even if $M$ has positive off-diagonals. (Hint: you'll need to build a $2n \times 2n$ matrix from $M$.)

## Problem 4: Building Augmented Spanning Tree Preconditioners

In this problem, we'll fill in some of the details about the construction of preconditioners from augmented spanning trees. Assume throughout the problem we've already sparsified our graph, so the degree is at most polylogarithmic.

**(a)** Suppose that we have an unweighted graph $G$ and a subgraph $H$. Suppose that every edge $(i, j)$ of $G$ is routed along some path $\pi(i, j)$ in $H$. Let $\text{st}(i, j)$ be the length of the path $\pi(i, j)$, and let

$$\Gamma = \max_{e \in H} \sum_{\substack{(i,j) \in E_G \text{ s.t.} \\ e \in \pi(i,j)}} \text{st}(i, j).$$

Call the inner summation the "total stretch over $e$." Show that

$$H \preceq G \preceq \Gamma H.$$

**(b)** Suppose that we take $H$ to be a low-stretch spanning tree, so that

$$\sum_{(i,j) \in E_G} \text{st}(i, j) \leq m\text{polylog}(n).$$

Call this sum the "total stretch" of $G$. Show that

$$H \preceq G \preceq m\text{polylog}(n)H.$$

**(c)** Suppose that we cut $G$ into pieces $G_1, \ldots, G_t$ and use this to induce a similar partition of the low-stretch spanning tree $H$ into $H_1, \ldots, H_t$. Suppose further that the total stretch of each $G_i$ is about the same, i.e., it is, up to polylogarithmic factors, the total stretch of $G$ divided by $t$, and that all of the $H_i$ are connected. Show that edges internal to $G_i$ (i.e., that don't cross this partition) can be routed along edges in $H_i$ so that the average stretch of such edges in $H_i$ is at most polylogarithmic in $n$. (Don't be alarmed if this strikes you as essentially immediate from the definitions.)

**(d)** Now add at most $t(t-1)/2$ edges to $H$ as follows. For every pair $(G_i, G_j)$ such that there is at least one edge in $G$ connecting a vertex in one set to the other, add the one of these edges with largest total stretch. Let the resulting graph be called $H'$. Show that

$$H' \preceq G \preceq (m/t) \cdot \text{polylog}(n)H'.$$

This has therefore produced a preconditioner with $n + O(t^2)$ edges that yields a condition number (of the preconditioned system) of $\widetilde{O}(n/t)$.

## Non-Problem: Chebyshev Polynomials

I was going to put a problem on the problem set that developed the properties of Chebyshev polynomials, but they are quite nicely laid out in Shewchuk's survey, and it seems silly to assign a problem that is answered in the course readings. Instead, I will just encourage you to read about them therein.

# Lattices

## Problem 1: General Lattice Problems

**(a)** Let $\Lambda \subseteq \mathbb{Z}^n$ be a full-rank lattice. Prove that $\Lambda \supseteq \det(\Lambda) \cdot \mathbb{Z}^n$.

**(b)** In class, we showed that LLL results eventually in a pretty good basis consisting of short vectors. Show, however, that just running the reduction step once can actually make some of the vectors in a lattice basis much longer. More precisely, construct a set of basis vectors $b_1, \ldots, b_n$ such that the result of running the LLL reduction step once contains a basis vector that is $\Omega(\sqrt{n})$ times longer than the longest of the $b_i$s.

**(c)** Let $b_1, \ldots, b_n$ be a reduced basis for some lattice $\Lambda$. In class, we showed that the length $b_1$ is within a factor of $2^{O(n)}$ of the length of the shortest vector in $\Lambda$. Show that this is tight by producing a reduced basis that meets this bound.

## Problem 2: Small Solutions to Modular Polynomial Equations and Attacking Low-Exponent RSA Encryption

Let $p(x) = c_0 + c_1 x + \cdots + c_{d-1}x^{d-1} + x^d$ be a monic polynomial with integer coefficients $c_0, \ldots, c_{d-1} \in \mathbb{Z}$, let $c_d = 1$, and let $M \in \mathbb{Z}$ be a positive integer. In this problem, we will try to find an integer $m$ such that:

- $p(m) \equiv 0 \mod M$, and
- $|m| \leq \sqrt[d]{M}$.

We will use this to attack the RSA cryptosystem when it is used with a small public exponent.

Finding the roots of a (reasonable) univariate polynomial over $\mathbb{R}$ is not very difficult, and it can be accomplished using standard techniques. The difficulty comes from the fact that we need to solve the equation mod $M$, where $M$ is an arbitrary (not necessarily prime) integer that we don't know how to factor. (If we knew how to factor $M$, there would be better solutions to this problem. However, we don't know how to factor large numbers in polynomial time, and RSA is based on this problem being difficult.) Our basic plan will be to reduce solving the modular equation to finding the roots of a small number of polynomials over $\mathbb{R}$.

**(a)** Suppose that, for some $X \in \mathbb{R}_{>0}$ and $h \in \mathbb{Z}_{>0}$, we have

$$\sum_{i=0}^{d} \left|c_i X^i\right| < M^{h-1}.$$

Show that any integer solution $x$ to $p(x) \equiv 0 \mod M^{h-1}$ with $|x| \leq X$ is also a solution to $p(x) = 0$ over $\mathbb{R}$.

**(b)** Let $m$ be any solution to $p(m) \equiv 0 \mod M$, and let

$$q_{ij}(x) = M^{h-i-1}x^j p(x)^i, \qquad i = 0, \ldots, h-1, \;\; j = 0, \ldots, d-1.$$

Show that $\sum_{i,j} \alpha_{ij} q_{ij}(m) \equiv 0 \mod M^{h-1}$ for any coefficients $\alpha_{ij} \in \mathbb{Z}$.

**(c)** For each polynomial $q_{ij}$, define a vector $b_{ij}$ whose $k^{\text{th}}$ coefficient is $a_k X^k$, where $a_k$ is the coefficient of $x^k$ in $q_{ij}$. [Note the extra factor of $X^k$ in the definition.] Collect these vectors into a matrix $B$, whose columns are the $b_{ij}$. Argue that the matrix $B$ is triangular, and compute its determinant.

**(d)** Let $\Lambda$ be the lattice generated by the $b_{ij}$. Note that its dimension is $dh$. Show that LLL finds a vector in $\Lambda$ whose $\ell_1$ norm is at most $2^{dh} \cdot dh \cdot M^{(h-1)/2} X^{(dh-1)/2}$. [Note that we are looking at $\ell_1$, while LLL gives you a bound on $\ell_2$.]

*Remark* 1. The input size for the problem is $O(d \log M)$, so we can construct the above lattice and run LLL when $h = \text{poly}(d, \log M)$. It will suffice for the rest of the question to take $h = \log(M)$.

**(e)** Use the above to find all $m$ such that $p(m) \equiv 0 \mod M$ and $|m| \leq C \sqrt[d]{M}$ for some constant $C$. Then, show how to achieve this for any fixed constant $C$ by running your algorithm a constant number of times (where the constant may depend on $C$).

**Parts (f) and (g) are for people familiar with the RSA cryptosystem.**

**(f)** Suppose Alice fixes her public exponent to be 3. (Our attack will work for any small integer. People have seriously considered such possibilities to improve the efficiency of encryption.) Suppose that Bob, Charlie, and David have pairwise relatively prime public keys $N_B$, $N_C$, and $N_D$, and that Alice sends the same message $m$ to each. This gives the ciphertexts $c_i = m^3 \mod N_i$ for $i \in \{B, C, D\}$. Show that an eavesdropper who received $c_B$, $c_C$, and $c_D$ (and who knows everyone's public keys) can recover $m$. You may use without proof that fact that the Chinese Remainder Theorem can be realized as an efficient algorithm (i.e., the number it guarantees to exist can be found efficiently). This part of the problem shouldn't require any lattices. [Note: remember that RSA requires the message $m$ to be less than each of the $N_i$, or else the receiver wouldn't be able to uniquely decrypt. You'll need to use this.]

**(g)** A common solution to the above problem is to concatenate the recipient's name to the message before encrypting it so that you send different messages to $B$, $C$, and $D$. More precisely, you replace the message $m$ with $M + 2^k R$, where $k$ is the length of the message and $R$ is some identification number assigned to the recipient. Use the algorithm derived earlier in this problem to show that an eavesdropper can still recover $m$ from $c_B$, $c_C$, and $c_D$.

18.409 Topics in Theoretical Computer Science: An Algorithmist's Toolkit
Fall 2009