1. Householder reflections. 10 points.

(a)

\[
Q^TQ = (I - 2uu^T)(I - 2uu^T)
= (I - 2uu^T)(I - 2uu^T)
= I - 2uu^T - 2uu^T + 4uu^Tuu^T
= I - 2uu^T - 2uu^T + 4uu^T
\text{ using } u^Tu = 1
= I \text{ so } Q \text{ is orthogonal}
\]

(b)

\[
Qu = u - 2uu^Tu = u - 2u = -u \text{ using } u^Tu = 1
Qv = v - 2uu^Tv = v \text{ using } u^Tv = 0
\]

(c) We know \(\det(Q) = \prod_{i=1}^{n} \lambda_i\). Since \(Q\) is symmetric, all eigenvalues are real and we can construct an orthonormal eigenvector basis. From parts (a) and (b), \(u\) is an eigenvector with associated eigenvalue \(-1\), and any vector \(v\) orthogonal to \(u\) is an eigenvector with associated eigenvalue 1. The nullspace of \(u^T\) has dimension \(n - 1\), so we can construct an orthogonal eigenbasis with all eigenvalues 1 except for the \(-1\) eigenvalue with eigenvector \(u\). Thus the product of the eigenvalues is \(-1 = \det(Q)\).

(d) Since \(Q\) is orthogonal, \(Q^TQ = I\) has all eigenvalues 1, hence all singular values of \(Q\) are 1, so \(\kappa(Q) = 1\) (i.e., \(Q\) is as well-conditioned as can be.)

(e) We follow the hint and choose \(u = (x + \alpha e_1)/\|x + \alpha e_1\|\). Then

\[
Q = I - 2\frac{(x + \alpha e_1)(x + \alpha e_1)^T}{(x + \alpha e_1)^T(x + \alpha e_1)}
= I - 2\frac{x(x^T + \alpha e_1^T) + \alpha e_1(x^T + \alpha e_1^T)}{x^Tx + \alpha e_1^Tx + \alpha x^Te_1 + \alpha^2 e_1^Te_1}
Qx = x - 2\frac{x(\|x\|^2 + \alpha e_1^Tx) + e_1(\alpha\|x\|^2 + \alpha^2 x_1)}{\|x\|^2 + 2\alpha x_1 + \alpha^2}
\]

\[
= x - 2\frac{2\|x\|^2 + 2\alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2} x - 2\alpha \frac{\|x\|^2 + \alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2} e_1
\]

Need this zero.
We can achieve this by choosing $\alpha = \pm \|x\|$. This leads to $Qx = \mp \|x\|e_1$ (which makes sense ... $Q$ should always preserve norm).

Some people used a geometric argument here as well, and this can make the solution a lot neater if it’s well presented. The idea is to find a reflection plane that reflects the given vector onto the $e_1$ axis (there are two possibilities, for negative and positive parts of the $e_1$ axis), and $u$ is then a unit vector orthogonal to this plane.

2. Two representations of an ellipsoid. 10 points.

First we will show that

$$\mathcal{E}_2 = \{ y \mid y^T(AA^T)^{-1}y \leq 1 \} \tag{1}$$

$$\begin{align*}
\mathcal{E}_2 &= \{ y \mid y = Ax, \|x\| \leq 1 \} \\
&= \{ y \mid A^{-1}y = x, \|x\| \leq 1 \} \quad \text{since } A \text{ is invertible square matrix} \\
&= \{ y \mid \|A^{-1}y\| \leq 1 \} \\
&= \{ y \mid y^T A^{-T} A^{-1} y \leq 1 \} = \{ y \mid y^T (AA^T)^{-1} y \leq 1 \}
\end{align*}$$

(a) 5 points

Since $S$ is symmetric positive definite, the eigenvalues of $S$ are all positive and we can choose $n$ orthonormal eigenvectors. So $S = QAQ^T$ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) > 0$ and $Q$ is orthogonal. Let $\Lambda^{\frac{1}{2}} = \text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$.

If we let $A = Q(\Lambda^{\frac{1}{2}})^{-1} = QA^{-\frac{1}{2}}$,

$$(AA^T)^{-1} = (QA^{-\frac{1}{2}} \Lambda^{-\frac{1}{2}} Q^T)^{-1}$$

$$= (Q \Lambda^{-1} Q^T)^{-1} = QAQ^T$$

Therefore, by (1) $A = QA^{-\frac{1}{2}}$ yields $\mathcal{E}_1 = \mathcal{E}_2$.

(b) 5 points

By (1), $S = (AA^T)^{-1}$ yields $\mathcal{E}_1 = \mathcal{E}_2$.

Uniqueness: We show that

$$\mathcal{E}_S = \mathcal{E}_T \Leftrightarrow S = T \tag{2}$$

where $\mathcal{E}_S = \{ x \mid x^T S x \leq 1 \}$, $\mathcal{E}_T = \{ x \mid x^T T x \leq 1 \}$, $S^T = S > 0$ and $T^T = T > 0$.

It is clear that if $S = T$, then $\mathcal{E}_S = \mathcal{E}_T$. Now we show that $\mathcal{E}_S = \mathcal{E}_T \Rightarrow x^T S x = x^T T x$, $\forall x \in \mathbb{R}^n$. Without loss of generality let’s assume $\exists x_0 \in \mathbb{R}^n$ such that $x_0^T S x_0 > x_0^T T x_0 = \alpha \neq 0$. If we let $x_1 = x_0 / \sqrt{\alpha}$, then $x_1^T T x_1 = 1$, but $x_1^T S x_1 > 1$, thus $x_1 \in \mathcal{E}_T$ but $x_1 \notin \mathcal{E}_S$, and therefore $\mathcal{E}_S \neq \mathcal{E}_T$. Finally, $\mathcal{E}_S = \mathcal{E}_T \Rightarrow x^T S x = x^T T x$, $\forall x \in \mathbb{R}^n \Rightarrow
\( S = T \) by the uniqueness of the symmetric part in a quadratic form (lecture notes 12-7, and from the homework.) Hence \( S \) is unique.

Given \( S \), find all \( A \) that yield \( \mathcal{E}_1 = \mathcal{E}_2 \).
The answer is
\[
A = QA^{-\frac{1}{2}}V^T
\]
where \( V \in \mathbb{R}^{n \times n} \) is any orthogonal matrix and
\[
S^T = S = QAQ^T > 0
\]
where \( Q \) is orthogonal and \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) > 0 \).
Let the singular value decomposition of \( A \) be
\[
A = U\Sigma V^T
\]
where \( U, V \in \mathbb{R}^{n \times n} \) are orthogonal and \( \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n) > 0 \) (since \( \det(A) \neq 0 \).)
By (1),
\[
\mathcal{E}_2 = \{ y \mid y^T(AA^T)^{-1}y \leq 1 \}
\]
\[
= \{ y \mid y^T(U\Sigma^2U^T)^{-1}y \leq 1 \}
\]
\[
= \{ y \mid y^TU\Sigma^{-2}U^Ty \leq 1 \}
\]
Thus, if \( \mathcal{E}_1 = \mathcal{E}_2 \), then \( S = U\Sigma^{-2}U^T \) by (2). Therefore \( U = Q \) and \( \Lambda = \Sigma^{-\frac{1}{2}} \), and \( V \) can be any orthogonal matrix.

You can also see why \( A \)'s are different only by right-side multiplication by an orthogonal matrix by the following argument. By (2) and (1), \( AA^T = S^{-1} \). Let
\[
A = \begin{bmatrix} \tilde{a}_1 & \tilde{a}_2 & \ldots & \tilde{a}_n \end{bmatrix}^T
\]
Then we have,
\[
\|\tilde{a}_i\|^2 = (S^{-1})_{ii},
\]
\[
\tilde{a}_i^T\tilde{a}_j = (S^{-1})_{ij},
\]
and
\[
\cos \theta_{ij} = \frac{(S^{-1})_{ij}}{\sqrt{(S^{-1})_{ii}(S^{-1})_{jj}}}.
\]
This means that the row vectors of any \( A \) satisfying \( AA^T = S^{-1} \) have the same length and the same angle between any two of them. So the rows of \( A \) can vary only by the application of an identical rotation or reflection to all of them. These are the transformations preserving length and angle, and correspond to orthogonal matrices. Since we are considering row vectors, the orthogonal matrix should be multiplied on the right.
3. Determining the number of signal sources. 15 points.

The quick answer to this problem is: count the number of large singular values of $Y$. Since the singular values of $Y$ are the square-root of the eigenvalues of $YY^T$, and $YY^T$ is a much smaller matrix, the most efficient way to solve this problem is to find the eigenvalues of $YY^T$ (the SVD of $YY^T$ works as well.) With the Matlab commands:

```matlab
z = eig(Y*Y');
stem(sqrt(z))
```

we see there are 7 large eigenvalues, which correspond to signal sources, and 13 small ones, due to noise (see figure). We conclude there are 7 sources present.

Let’s justify our method. There are many ways to think about this problem, we’ll discuss one of them. First, collect the source signal vectors $x(k)$ in the matrix $X \in \mathbb{R}^{n \times p}$,

$$X = [x(1) \ x(2) \ \cdots \ x(p)]$$

and the noise vectors in the matrix $V \in \mathbb{R}^{m \times p}$,

$$V = [v(1) \ v(2) \ \cdots \ v(p)].$$

Now the equations can be summarized as $Y = AX + V$.

We start by noting that the range of $A$ in an $n$-dimensional subspace of $\mathbb{R}^m$. Also, our assumptions mean that the matrix $X$ is full-rank and well-conditioned: Since the $x(k)$ point equally likely in all directions, we have that

$$\frac{1}{p} \sum_{k=1}^{p} x(k)^T v$$
has about the same value for any unit length vector $v$ (and for large enough $p$.) Hence, the matrix $X^T$ has about the same gain in all directions, which implies that all the singular values of $X$ have about the same value (and $\kappa(X)$ is on the order of 1.)

Since $X$ can map a vector into any point in $\mathbb{R}^n$, the matrix $AX$ has the same rank and range as $A$, i.e., can map into any point in the range of $A$. Also, the ratio between the maximum singular value of $AX$ and the smallest non-zero singular value is not large (it’s bounded by $\kappa(A)\kappa(X)$.)

A geometrical interpretation is that both $A$ and $AX$ map a ball into an ellipsoid that’s flat in $m - n$ directions, and has a moderate aspect ration in the other directions (because of well-conditioning in the case of $A$, and because of the aspect ratio of the non-zero axes in the case of $AX$.)

Another way to think about it is to view the columns of $X$ as “filling up” a ball in $\mathbb{R}^n$. Then, the columns of $AX$ “fill up” a flat ellipsoid in $\mathbb{R}^m$ (with $n$ non-zero axes.)

The columns of $Y = AX + V$ will no longer be in the range of $A$ because of the noise (in fact, $Y$ can be expected to be full-rank.) The ellipsoid is no longer flat, but not by much – if the noise is small, the semi-axes that do not correspond to the range of $A$ are small.

Another way to see this is as follows. Since $AX$ has $n$ large singular values, the gain of $Y$ in the direction of the input singular vectors of $AX$ is also large (greater or equal than $\|AXv_i\| - \|Vv_i\|$.) Also, there are $p - n$ orthogonal directions for which $AX$ has zero gain (the nullspace of $AX$), and the gain of $Y$ in those directions is small (equal to the gain of $V$.) In summary, $Y$ must have $n$ orthogonal directions with large gain, and $p - n$ orthogonal directions with small gain.


**Part I**

(a) The system is controllable iff its controllability matrix $C = [B \ AB \ \cdots \ A^{19}B]$ is full-rank. This is straightforward to check in Matlab, and it turns out that the system is controllable when the actuator is placed on mass 1, 2, 4, 5, 8 or 10.

(b) What should we minimize or maximize here? The problem statement says any state should be reachable with relatively small input forces. This suggests that we should minimize the maximum (over all $x$) energy required to reach $x$ with an infinite horizon. So we really want to attain the “minimum maximum minimum energy”! (No, that’s not a typo!)

The energy required to reach a state $x$ with an infinite horizon is $x^TPx$ where $P$ is the inverse of the controllability Gramian $W$. We recall the bound $x^TPx \leq \lambda_1(P)x^Tx$, in which equality is attained for $x = v_1$ (the eigenvector corresponding to the largest eigenvalue $\lambda_1$). So our goal is equivalent to minimizing $\lambda_1(P)$. Note that $\lambda_1(P) = 1/\lambda_n(W)$ and so we want to find the actuator placement that results in the largest $\lambda_n(W)$. The accompanying Matlab code does this, and tells us to place the actuator on mass 4.
Some people pointed out that since $P$ is symmetric and positive semi-definite, its eigenvalues are the same as its singular values, so minimizing the max singular value is equivalent, and therefore minimizing the matrix norm $\|P\|$ is also equivalent. This is all OK.

A common error was to minimize the *minimum* minimum energy! That is, many people chose the $P$ with the smallest $\lambda_{\text{min}}$ or $\sigma_{\text{min}}$. This finds the actuator placement whose cheapest-to-reach unit-norm state is cheaper than that of the other actuator placements, i.e., the absolute cheapest thing that could happen! That’s not what we asked for.

**Part II**

(a) This is checked the same way as in part I (a), and the answer is no, the system isn’t controllable ($\text{Rank } (C) = 6$).

(b) Here we follow the hint and discretize, with $N = 100$ steps and thus a stepsize of $h = t_f/N = 0.2$. The discretized state equation is $x_d(k+1) = A_d x_d(k) + B_d u_d(k)$ with $A_d = e^{Ah}$ and $B_d = \int_0^h e^{At} B dt = A^{-1}(e^{Ah} - I) B$.

Define $C_d(N) = [B_d \ A_d B_d \ \cdots \ \ A_d^{N-1} B_d]$. Then

$$x(t_f) \simeq x_d(N) = A_d^N x(0) + C_d(N) \begin{bmatrix} u_d(N-1) \\ u_d(N-2) \\ \vdots \u_d(0) \end{bmatrix} = A_d^N x(0) + C_d(N) U_d$$

We want to find $U_d$ so that $x(t_f)$ is minimized. In other words, we seek a least-squares-approximate solution $U_d$ to

$$-A_d^N x(0) = C_d(N) U_d$$

and not just any such solution – we seek the minimum-norm one. (Since $C_d(N)$ is a fat but non-full-rank matrix, there are in general many solutions that achieve the optimal LS error).

This just screams out “Moore-Penrose pseudo-inverse!” (Many people went ahead and used regularized least-squares here, but that doesn’t match what we asked for in the problem statement. Reg-LS finds a tradeoff solution, but we asked for the absolute minimum LS error, and the smallest input that achieves it. That’s a job for the general pseudo-inverse)

So we compute $U_d = C_d(N)^\dagger (-A_d^N x(0))$ and that’s our approximation to the continuous-time optimal force input $u(t)$. The result is plotted in figure 2, and Matlab code follows. The input energy is $E_{u,\text{min}} = 0.1643$ and the final state error $\|x(t_f)\| = 0.2719$.

*Matlab code for Problem 4*

```matlab
disp('Part I');```

---

6
Figure 2: The optimal control force for Problem 4, Part II (b)

disp('=======');
disp(' ');
disp('(a) Controllability');
disp(' ');

% I commented out the k,d,p,act lines in spring_series.m
% so that we can set them outside it!
k=1;
d=0.01;
p=10;
for act=1:10,
    spring_series; % my modified version with k,d,p,act lines commented out
    C=ctrb(A,B);
    if(rank(C)==20),
        disp(['Controllable with actuator on ' num2str(act)]);
    else
        disp(['Not controllable with actuator on ' num2str(act)]);
    end;
end;

disp(' ');
disp('(b) Optimal actuator placement');
disp(' ');

7
for act=[1 2 4 5 8 10],
    spring_series;
    W=gram(A,B);
    l_min_W = min(eig(W));
    disp(['With actuator on ' num2str(act) ', maxmin = ' num2str(1/l_min_W)])
end;

disp('Minimum over all actuator placements, of maximum over all unit x, of');
disp(['minimum energy required to reach x, is ' num2str(1/l_min_W)]);

disp(' ');
disp('Part II');
disp('==');
disp(' ');
disp('(a) Controllability');
disp(' ');

p=4;
act=3;
spring_series;
C=ctrb(A,B);
disp(['Rank of C is ' num2str(rank(C)) ' < 8,']); % rank is 6
    disp(['hence system is not controllable.']);

disp(' ');
disp('(b) Least-norm least-squares input');
disp(' ');
disp('See graph');

x0 = zeros(2*p,1); x0(2*p)=1;
tf=20;
N=100;
h=tf/N;
Ad=expm(A*h);
Bd=inv(A)*(Ad-eye(size(Ad)))*B;
Cd=Bd;
for i=1:N-1,
    Cd=[Cd Ad^i*Bd];
end;
Ud=pinv(Cd)*(-Ad^N*x0);
plot(0:h:tf-h,flipud(Ud)); % flip Ud around because it's in reverse order!

*****************
* MATLAB OUTPUT *
***************

>> springs_gulli
Part I
=====

(a) Controllability

Controllable with actuator on 1
Controllable with actuator on 2
Not controllable with actuator on 3
Controllable with actuator on 4
Controllable with actuator on 5
Not controllable with actuator on 6
Not controllable with actuator on 7
Controllable with actuator on 8
Not controllable with actuator on 9
Controllable with actuator on 10

(b) Optimal actuator placement

With actuator on 1, maxmin=26.8511
With actuator on 2, maxmin=22.6665
With actuator on 4, maxmin=7.3694
With actuator on 5, maxmin=93.9486
With actuator on 8, maxmin=38.2856
With actuator on 10, maxmin=104.7655
Minimum over all actuator placements, of maximum over all unit x, of
minimum energy required to reach x, is 104.7655

Part II
=====

(a) Controllability

Rank of C is 6 < 8,
hence system is not controllable.

(b) Least-norm least-squares input

See graph

5. Output feedback and maximum damping. 10 points.
(a) 3 points.
From the equations given in the problem,

\[
x(t + 1) = Ax(t) + Bu(t) \\
= Ax(t) + BK_y(t) \\
= Ax(t) + BK_Cx(t) \\
= (A + BK_C)x(t)
\]

Therefore, \( \tilde{A} = A + BK_C \).

(b) 7 points.
Since \( x(t) = (\tilde{A})^t x(0) \), this is a discrete-time autonomous system. Thus, the system is stable iff all the eigenvalues of \( \tilde{A} \) are less than 1 in magnitude, that is, \( |\lambda(\tilde{A})| < 1 \). Since the largest eigenvalue in magnitude corresponds to the slowest mode, it determines asymptotic decay rate. Therefore, this feedback system is maximally damped when this largest eigenvalue is minimized over \( K \). So

\[
K_{opt} = \arg\min_K \max |\lambda_i(\tilde{A})|
\]

There is no easy analytic solution for \( K_{opt} \); actually \( \max_i |\lambda_i(\tilde{A})| \) is a non-differentiable function of \( K \) and, in general, can have local minima.

So we just compute the \( \max_i |\lambda_i(\tilde{A})| \) as function of \( K \) (at small intervals) and find the minimum point. \( K_{opt} \) is found to be \( 1.93 \pm 0.005 \) and \( \min_K \max_i |\lambda_i(\tilde{A})| \) is \( 0.74895 \).

*Note:* \( \max_i |\lambda_i(\tilde{A})| \) is called the spectral radius of the matrix \( \tilde{A} \) and denoted by \( \rho(\tilde{A}) \). When \( K \) is a matrix instead of a scalar, this simple approach won’t work. (Until recently this was thought to be a very hard problem! You can learn the tools you need to solve this problem very efficiently for any size \( K \) in EE364.)

Matlab code for Problem 5

```Matlab
A=[0.5 1.0 0.1; -0.1 0.5 -0.1; 0.2 0.0 0.9];
B=[1;0;0];
C=[0 1 0];
Klist = -2:1e-3:3;
spectral_radius = [];

for K = Klist
    spectral_radius = [ spectral_radius max(abs(eig(A+K*B*C))) ];
end

[sr_opt,i_opt] = min( spectral_radius );
```
Kopt = Klist(i_opt);

figure(1); plot(Klist, spectral_radius);
title( ['Spectral Radius; K_{opt} = ', num2str(Kopt), ', \rho_{opt} = ', num2str(sr_opt) ]);
xlabel('K'); ylabel('\rho');
grid on

6. Quadratic least-squares extrapolation of a time series. 10 points.

\[ f(n + k) = a_2(n + k)^2 + a_1(n + k) + a_0 \]
\[ = a_2k^2 + (2na_2 + a_1)k + (n^2a_2 + na_1 + a_0) \]
\[ = u_2(n)k^2 + u_1(n)k + u_0(n) \]

by letting

\[ u_2(n) = a_2 \]
\[ u_1(n) = 2na_2 + a_1 \]
\[ u_0(n) = n^2a_2 + na_1 + a_0 \]
Then,

\[
\begin{bmatrix}
  f(n) \\
f(n-1) \\
  \vdots \\
f(n-9)
\end{bmatrix} =
\begin{bmatrix}
  0^2 & 0 & 1 \\
  (-1)^2 & -1 & 1 \\
  \vdots & \vdots & \vdots \\
  (-9)^2 & -9 & 1
\end{bmatrix}
\begin{bmatrix}
  u_2 \\
u_1 \\
u_0
\end{bmatrix}
\]

Let \( u(n) = u = [u_2 \ u_1 \ u_0]^T \). Then,

\[
\min \sum_{k=-9}^{0} (z(n+k) - f(n+k))^2 = \min \| x - Au \|^2
\]

This is a least-squares problem since \( A \) is skinny and full rank. The optimal solution is \( u_{opt} = (A^T A)^{-1} A^T x \). Therefore,

\[
y = f(n+1) = a_{2, opt}(n+1)^2 + a_{1, opt}(n+1) + a_{0, opt}
\]

\[
= a_{2, opt} + (2n a_{2, opt} + a_{1, opt}) + (n^2 a_{2, opt} + n a_{1, opt} + a_{0, opt})
\]

\[
= u_{2, opt}(n) + u_{1, opt}(n) + u_{0, opt}(n)
\]

\[
= [1 \ 1 \ 1] u_{opt} = [1 \ 1 \ 1] (A^T A)^{-1} A^T x.
\]

Therefore \( c = A(A^T A)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \).

So \( c \) is indeed independent of \( n \) and is equal to:

\[
c = [0.900 \ 0.500 \ 0.183 \ -0.050 \ -0.200 \ -0.267 \ -0.250 \ -0.150 \ 0.033 \ 0.300]^T
\]

Matlab code for Problem 6

\[
a = (0:-1:-9)';
A = [ a.^2 a.^1 a.^0 ];
c = A*inv(A'*A)*[1 1 1]'
\]

Grading criteria
2 points for noting that \( c \) is time-invariant (i.e. independent of \( n \)).
2 points for showing that \( c \) is time-invariant.
6 points for finding \( c \).

7. The EE263 search engine. 15 points.

The singular values of the normalized matrix are shown in the figure. The full-rank search yields the documents:
The corresponding weights are $c = [0.60 \ 0.55 \ 0.50 \ 0.49 \ 0.41]^T$.

Given the SVD of $\tilde{A} = U\Sigma V^T$, the nearest low-rank approximation is easily computed:

$$\hat{A}_r = \min_{A: \text{rank}(A) \leq r} \| \tilde{A} - \hat{A} \| = \sum_{k=1}^{r} \sigma_k u_k v_k^T = U_1 W_1^T,$$

where $\sigma_k$ are the singular values, $u_k$ are the columns of $U$, $v_k$ are the columns of $V$. We have also written the low rank approximation as the product of two smaller full-rank matrices, $U_1 = [u_1 \cdots u_r]$, and $W_1 = [\sigma_1 v_1 \cdots \sigma_r v_r]$.

The search results are as follows (the table includes the document numbers only, i.e., the last part of the URL):

<table>
<thead>
<tr>
<th>Rank</th>
<th>1st (Document)</th>
<th>2nd (Document)</th>
<th>3rd (Document)</th>
<th>4th (Document)</th>
<th>5th (Document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>103562</td>
<td>103562</td>
<td>101252</td>
<td>101252</td>
<td>100107</td>
</tr>
<tr>
<td>16</td>
<td>103561</td>
<td>103563</td>
<td>106106</td>
<td>103563</td>
<td>104173</td>
</tr>
<tr>
<td>8</td>
<td>103561</td>
<td>103563</td>
<td>106106</td>
<td>103563</td>
<td>104173</td>
</tr>
<tr>
<td>4</td>
<td>101252</td>
<td>101252</td>
<td>103563</td>
<td>104173</td>
<td>100651</td>
</tr>
<tr>
<td>4</td>
<td>100107</td>
<td>100107</td>
<td>100107</td>
<td>100107</td>
<td>100651</td>
</tr>
</tbody>
</table>
And the corresponding weights are:

<table>
<thead>
<tr>
<th>rank</th>
<th>rank 32</th>
<th>rank 16</th>
<th>rank 8</th>
<th>rank 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>0.6445</td>
<td>0.5474</td>
<td>0.3751</td>
<td>0.2212</td>
</tr>
<tr>
<td>2nd</td>
<td>0.4952</td>
<td>0.4519</td>
<td>0.3666</td>
<td>0.1961</td>
</tr>
<tr>
<td>3rd</td>
<td>0.4726</td>
<td>0.4312</td>
<td>0.3462</td>
<td>0.1912</td>
</tr>
<tr>
<td>4th</td>
<td>0.4299</td>
<td>0.3910</td>
<td>0.3322</td>
<td>0.1894</td>
</tr>
<tr>
<td>5th</td>
<td>0.3846</td>
<td>0.3745</td>
<td>0.2998</td>
<td>0.1824</td>
</tr>
</tbody>
</table>

Note the “loss of resolution” for the lower rank searches, as the weights become less differentiated.

Down to rank 16 the low-rank search is essentially equivalent to the full-rank search. For rank 8, the top 5 results are still very similar, although their ordering is somewhat changed. For rank 4 the results start differing significantly, although there are still some top 5 documents in common with the full-rank search.

However, if you look at the content of the documents (something that almost no-one seems to have done), you’ll see that the ones returned by the low-rank searches appear to be more relevant! Of course, this is a subjective appreciation... But we’ll get back to this in a while, when discussing the advantages of the low-rank search.

The Matlab code for this problem is as follows:

```matlab
term_by_doc  % run the script that defines matrix A

t=53;  % index of term(s) to search for (t=[53 64] also works)
q=zeros(m,1);  % query vector
q(t)=1;  % put 1 in the entries indexed by the vector term
disp('query:')
for k=1:length(t), disp(term{t(k)}), end  % print words in query
for j=1:n, An(:,j)=A(:,j)/norm(A(:,j)); end  % normalize columns
q=q/norm(q);  % normalize query

%%% full-rank search %%%

c0=An'*q;
[w0,i0]=sort(c0);  % sort in ascending order
w0=flipud(w0);  % flip to get descending order
i0=flipud(i0);
disp('full-rank top 5:')  % print top 5 docs
for k=1:5, disp(document{i0(k)}), end
disp(w0(1:5))  % print weights of the top 5 docs
```
%% low-rank searches %

[U,S,V]=svd(A);
stem(diag(S)) % plot of singular values
grid on
for p=1:4
    r=2^(6-p); % r=2.^((5:-1:2)=[32 16 8 4])
    % submatrices for low-rank approximation
    U1=U(:,1:r);
    S1=S(1:r,1:r);
    V1=V(:,1:r);
    % perform query (no need to rebuild the matrix!)
    c1=(V1*S1)*(U1'*q);
    [w1,i1]=sort(c1);
    w1=flipud(w1);
    i1=flipud(i1);
    disp(' ')
    disp('low-rank top 5:') % print top 5 docs
    for k=1:5, disp(document{i1(k)}) end
    disp(w1(1:5)') % print weights of the top 5 docs
end

Now, for the advantages of low-rank search.

The most obvious one is query speed, which can be achieved by computing the query using the factored form of $\hat{A}_r$:

$$c = W_1(U_1^T q)$$

(with $U_1$ and $W_1$ as defined above.) The full-rank query (and the low-rank without using the factored form) requires about $nm$ operations. The factored low-rank query requires about $r(m+n)$ operations. Of course, computing the SVD for a large matrix requires a large numbers of operations, so the number of searches performed between updates of the matrix must be large for this to pay off. (Note that there are techiques for doing an approximate update of the low-rank approximation when a new document is added without recomputing the whole SVD.)

Also, note that the SVD can be done off-line. Even if it’s overall more expensive (i.e., requires more operations) to do the SVD plus the low-rank searches than to do all searches on the full-rank matrix, the user will be happy to have faster searches. Meanwhile another computer can be used to work in parallel on the next updating of the low-rank approximation.

The only part of this problem that most people didn’t get was the “latent semantic indexing” property, which can be interpreted as a form of “de-noising.” More than query speed, this is actually the driving motivation for the use of low rank approximations in database searches.
The best way to explain this is by example, so here’s a good one (from a student’s exam solution!) Consider the following term-by-document matrix:

<table>
<thead>
<tr>
<th></th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>30</td>
<td>15</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>automobile</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>penguin</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>coffee</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>14</td>
</tr>
</tbody>
</table>

The SVD of the normalized $\tilde{A}$ yields the singular values:

$$
\sigma = [1.33, 1.08, 0.88, 0.54].
$$

Note that the fourth is smaller, but not by that much. Also from the SVD, we get:

$$
U = \begin{bmatrix}
0.89 & -0.23 & 0.09 & -0.38 \\
0.37 & -0.11 & -0.01 & 0.92 \\
0.13 & 0.71 & 0.69 & 0.03 \\
0.23 & 0.66 & -0.72 & -0.02
\end{bmatrix}.
$$

We can give a “semantic” interpretation of each dyad. The first vector weights the terms car and automobile. The second vector weights the terms penguin and coffee. The third vector puts differential weights on penguin and coffee. The fourth vector puts differential weights on car and automobile.

If we remove the fourth dyad, the result is that the differentiation between car and automobile is removed! Here’s the resulting low-rank approximation of $\tilde{A}$:

$$
\hat{A}_3 = \begin{bmatrix}
0.85 & 0.85 & 0.05 & 0.14 \\
0.35 & 0.35 & -0.01 & 0.08 \\
0.01 & -0.01 & 0.99 & 0.07 \\
-0.01 & 0.01 & 0.15 & 0.98
\end{bmatrix}.
$$

We see that car and automobile are now weighted equally in documents 1 and 2. A search for automobile (i.e., $q = [0 1 0 0]^T$) on the original $\tilde{A}$ yields zero relevance for the first document:

$$
c = \tilde{A}^T q = [0.00, 0.71, 0.00, 0.07]^T.
$$

The same search on the low-rank approximation $\hat{A}_3$ produces the much more sensible result:

$$
c = \hat{A}_3^T q = [0.35, 0.35, 0.01, 0.08]^T.
$$

Documents 1 and 2 are now considered to have equal relevance for the word automobile! The conclusion is that even though the search results of a low-rank approximation may be “less accurate” (in the sense that they differ from those of a search with the original matrix), they may actually be better from the user’s point of view.