CSE 383C: Assignment 6

Brady Zhou

November 15, 2016

Exercise 1. Find perturbations that maximize the error in the solution for $Ax = b$.

(a) Find ΔA, with $\|\Delta A\|_2 = \tau\|A\|_2$, which maximizes $\|\Delta x\|_2$, where $(A+\Delta)(x+\Delta x) = b$.

Taking the SVD of A, we can see we have a decomposition of the form -

$$U \begin{bmatrix} 2 & 0 \\ 0 & 0.001 \\ 0 & 0 \end{bmatrix} V^T$$

We want to find $\|\Delta A\|_2$ such that $A + \Delta A$ spans a different space. This can be achieved by putting all the energy into the third row of Σ, since this U_3 is orthogonal to the range of A. U and V^T are orthonormal, they have a norm of 1, so we can reduce the problem to finding a $\Delta\Sigma$ with the same constraints.
Since we require $\|\Delta\Sigma\|_2 = \tau\|A\|_2$, and $\|A\|_2 = \sigma_1$, ΔA takes the form -

$$U \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -\tau\sigma_1 \end{bmatrix} V^T$$

Since A and ΔA are still full rank, we can solve for x and Δx using the corresponding pseudoinverses, and the relative error comes out to $\sim 17200\%$, a surprising result.

(b) Find Δb, with $\|\Delta b\|_2 = \tau\|b\|_2$, which maximizes $\|\Delta x\|_2$, where $A(x + \Delta x) = (b + \Delta b)$.

From the properties of SVD, we know that U_1 and U_2 span the range of A. We want to pick a direction U_i to "push" b further away from the range of A.
Since U_3 is not in the range of A, there is no point in perturbing b in that direction.
U_2 is "poorly spanned" by A, so this will be the best direction to perturb b in.
Δb will be a scalar multiple α times U_2.
The constraints on Δb makes $\alpha = \tau\|b\|_2$, so $\Delta b = \tau\|b\|_2 U_2$.

Solving for x and Δx using the pseudoinverses gives us that the relative error comes out to $\sim 20\%$.

Exercise 2. Let $A \in \mathbb{R}^{m\times n}$, $m < n$, $\text{rank}(A) = m$. A is underdetermined and full rank. Let $b \in \mathbb{R}^m$. Want to find $x \in \mathbb{R}^n$ such that $Ax = b$.

(a) Assume that $x = A^T y$.

Now we want to find the solution to $AA^T y = b$.
We can see $AA^T \in \mathbb{R}^{m\times m}$, and since $\text{rank}(A) = m$, $\text{rank}(AA^T) = m$, so AA^T is full rank and square and we can take the inverse of this.

$$y = (AA^T)^{-1}b$$
$$x = A^T(AA^T)^{-1}b$$

(b) Use the QR factorization of A^T.

For QR decomposition, we have the requirement that $m \geq n$, so A^T satisfies this. The fact that A is full rank allows us to use either Gram-Schmidt or Householder, but we have to make sure to used reduced QR.
With a reduced QR, we have $A^T = QR$, where $Q \in \mathbb{R}^{n\times m}$, and $R \in \mathbb{R}^{m\times m}$, since A is full rank.
So we have $A = R^T Q^T$, and want to solve $R^T Q^T x = b$. Since R^T is full rank and square, there exists an inverse R^{-T}.
So now, $Q^T x = R^{-T}b$, and since the solution x lives in the rowspace of A, and Q^T forms a basis for the range of A, Q forms a basis for the rowspace of A and we have $x = Qy$.
$Q^T Q y = R^{-T}b$, which gives us that $y = R^{-T}b$.

$$y = R^{-T}b$$
$$x = QR^{-T}b$$

(c) Relation to TSVD.

We can see finding a decomposition of the transpose and using that is equivalent to constructing the projector U_k from the TSVD.

Exercise 3. Least squares problems when A is nearly rank deficient.

(a) Plotting singular values.

See attached.

(b) Rank analysis. Numerical rank is calculated with MATLAB $\text{rank}(A, \sigma_1 * \tau)$.

M	N	Rank r	Numerical Rank k
1024	512	512	25
512	1024	512	21
1024	512	99	17
1024	512	512	44
1024	1024	1023	469

We can see for certain problems the matrix is clearly full rank. However, the numerical rank are quite low for many of the problems provided.

(c) Best k-rank approximation for range of A.

We can see that SVD performs the best, but takes the longest. Both QR schemes (pivoted and nonpivoted) have unbounded amount of error, but with pivoting, we can see the relative error is less overall. The randorth scheme also predicts fairly well.

(d) Figure descriptions.

The first five figures consist of plotting singular values. The next five figures consist of relative error for x_k as a function of k-rank approximations using the TSVD. The next 5 figures consist of l-curve analysis for problems 1-5.

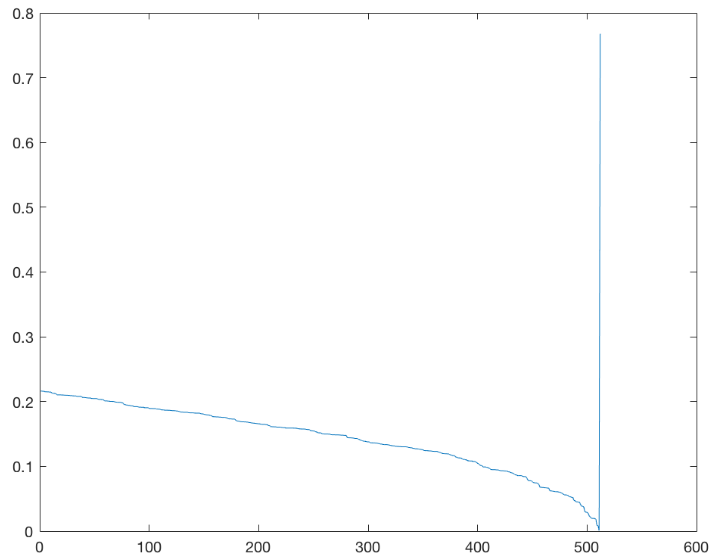
```
for i = 1:5
    fprintf('Problem %d:\n', i);
    [A, b, tau, xstar] = problem(i);
    [m, n] = size(A);

    % The numerical rank is defined as sigma_1 / sigma_k = tau.
    r = rank(A);
    k = rank(A, tau * rank(A));

    % Plot the singular values.
    [U, S, V] = svd(A);
    sigmas = diag(S);
    % figure;
    % semilogy(1:size(sigmas, 1), sigmas);

    % Get QR decomposition with and without pivoting.
    [Q, R] = qr(A);
    [Q_p, R_p, P_p] = qr(A);

    % Solving for Ax = b using SVD.
    k_values = n * [1/16, 1/8, 1/4, 1/2];
    for j = 1:size(k_values, 2)
        k_i = k_values(j);

        % Using TSVD.
        U_k = U(:,1:k_i);
        S_k = S(1:k_i,1:k_i);
        V_k = V(:,1:k_i);
        x_svd = V_k * inv(S_k) * U_k' * b;
        dx_svd_norm = norm(x_svd - xstar) / norm(xstar);

        % Using QR without pivoting.
        Q_k = Q(:,1:k_i);
        R_k = R(1:k_i,:);
        y_qr = Q_k' * b;
        x_qr = R_k \ y_qr;
        dx_qr_norm = norm(x_qr - xstar) / norm(xstar);

        % Using QR with pivoting.
        Q_p_k = Q_p(:,1:k_i);
        R_p_k = R_p(1:k_i,:);
        y_qr_p = Q_p_k' * b;
        x_qr_p = R_p_k \ y_qr_p;
        dx_qr_p_norm = norm(x_qr_p - xstar) / norm(xstar);

        % Using randorth.
        Z_k = randorth(A, sigmas(k_i));
        B = Z_k' * A;
        [W_k, T_k] = qr(B');
        y_randorth = T_k * Z_k' * b;
        x_randorth = W_k \ y_randorth;
        dx_randorth_norm = norm(x_randorth - xstar) / norm(xstar);

        % Print errors.
        fprintf('%d-Rank Error Analysis:\n', k_i);
        fprintf('TSVD Error: %f.\n', dx_svd_norm);
        fprintf('QR Error: %f.\n', dx_qr_norm);
        fprintf('QR Error (with pivoting): %f.\n', dx_qr_p_norm);
        fprintf('Randorth Error: %f.\n', dx_randorth_norm);
    end
```

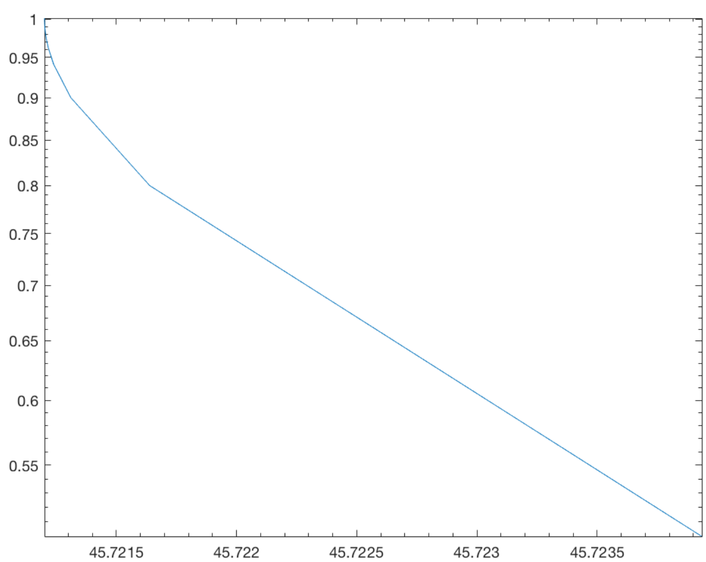
```
    % Plotting relative error from solution using TSVD.
    [U, S, V] = svd(A);
    points = [];
    for k = 1:rank(A)
        U_k = U(:,1:k);
        S_k = S(1:k,1:k);
        V_k = V(:,1:k);

        x_svd = V_k * inv(S_k) * U_k' * b;
        dx_svd_norm = norm(x_svd - xstar) / norm(xstar);
        points = [points dx_svd_norm];
    end
    figure;
    plot(1:rank(A), points);

    % Different beta finding methods.
    beta1 = discrepancy_analysis(A, b, tau, xstar);
    l_curve(A, b, tau, xstar);
    % beta2 = cross_validation(A, b, tau, xstar);

    % Test out betas.
    [U, S, V] = svd(A);
    [m, n] = size(A);
    sigma = diag(S);

    b_hat = U' * b;
    x_hat = zeros(n, 1);
    k = min(m, n);
    x_hat(1:k) = (sigma(1:k).^2)./(sigma(1:k).^2 + beta1).*b_hat(1:k)./sigma(1:k);
    x_prime = V * x_hat;
    fprintf('Discrepancy analysis found beta = %f, with error %f.\n', norm(x_prime - xstar));

    % x_hat = zeros(n, 1);
    % k = min(m, n);
    % x_hat(1:k) = (sigma(1:k).^2)./(sigma(1:k).^2 + beta2).*b_hat(1:k)./sigma(1:k);
    % x_prime = V * x_hat;
    % fprintf('Cross validation analysis found beta = %f, with error %f.\n', norm(x_prime - xstar));
end
```

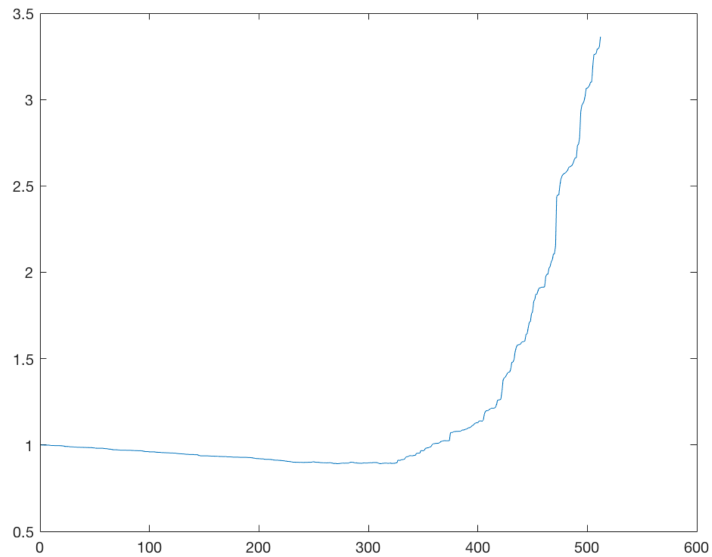
```
Problem 1:
32-Rank Error Analysis:
TSVD Error: 0.208794.
QR Error: 5.859683.
QR Error (with pivoting): 4.997082.
Randorth Error: 0.532031.
64-Rank Error Analysis:
TSVD Error: 0.200362.
QR Error: 4.236537.
QR Error (with pivoting): 4.249700.
Randorth Error: 0.810536.
128-Rank Error Analysis:
TSVD Error: 0.184480.
QR Error: 4.432005.
QR Error (with pivoting): 2.708385.
Randorth Error: 0.967829.
256-Rank Error Analysis:
TSVD Error: 0.151705.
QR Error: 3.450810.
QR Error (with pivoting): 1.991164.
```

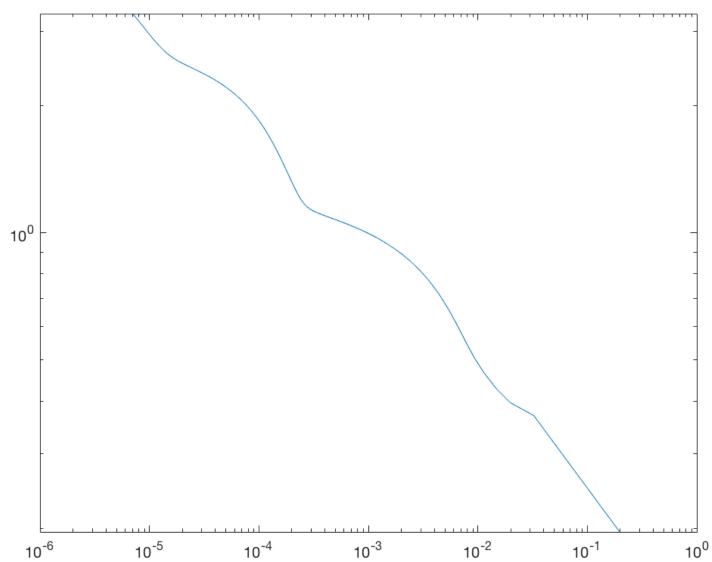
```
Randorth Error: 1.299280.
Discrepancy analysis found beta = 0.546869, with error Problem 2:
64-Rank Error Analysis:
TSVD Error: 0.975740.
QR Error: 2.077609.
QR Error (with pivoting): 1.897389.
Randorth Error: 1.000101.
128-Rank Error Analysis:
TSVD Error: 0.949669.
QR Error: 2.689937.
QR Error (with pivoting): 1.962711.
Randorth Error: 1.000016.
256-Rank Error Analysis:
TSVD Error: 0.897836.
QR Error: 3.072351.
QR Error (with pivoting): 1.918374.
Randorth Error: 1.000084.
512-Rank Error Analysis:
TSVD Error: 3.363632.
QR Error: 23.032910.
QR Error (with pivoting): 22.979055.
Randorth Error: 1.000008.
Discrepancy analysis found beta = 32.427183, with error Problem 3:
32-Rank Error Analysis:
TSVD Error: 0.962808.
QR Error: 1.637452.
QR Error (with pivoting): 1.582697.
Randorth Error: 0.998930.
64-Rank Error Analysis:
TSVD Error: 0.924262.
QR Error: 2.165237.
QR Error (with pivoting): 1.843967.
Randorth Error: 0.998159.
128-Rank Error Analysis:
TSVD Error: 781163346968162.875000.
QR Error: 3.486776.
QR Error (with pivoting): 3.587672.
Randorth Error: 1.026395.
256-Rank Error Analysis:
TSVD Error: 1756184301608475.750000.
QR Error: 3.608916.
QR Error (with pivoting): 3.587672.
Randorth Error: 1.001435.
Discrepancy analysis found beta = 22.654055, with error Problem 4:
32-Rank Error Analysis:
TSVD Error: 0.963583.
QR Error: 2.025398.
QR Error (with pivoting): 1.575018.
Randorth Error: 0.999638.
64-Rank Error Analysis:
TSVD Error: 0.932301.
QR Error: 199.439315.
QR Error (with pivoting): 15.528565.
Randorth Error: 1.000976.
128-Rank Error Analysis:
TSVD Error: 113868.358680.
QR Error: 605987.956600.
QR Error (with pivoting): 300793.307285.
Randorth Error: 0.999731.
256-Rank Error Analysis:
TSVD Error: 114018.649975.
QR Error: 437638.913963.
```

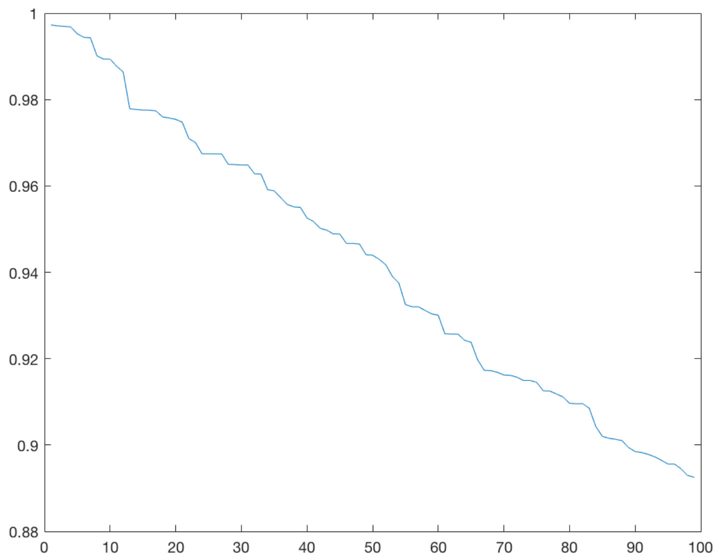
```
QR Error (with pivoting): 207238.414627.
Randorth Error: 0.999269.
Discrepancy analysis found beta = 22.189749, with error Problem 5:
64-Rank Error Analysis:
TSVD Error: 0.976008.
QR Error: 1.222400.
QR Error (with pivoting): 1.053778.
Randorth Error: 22.380281.
128-Rank Error Analysis:
TSVD Error: 0.935030.
QR Error: 1.411550.
QR Error (with pivoting): 1.104557.
Randorth Error: 22.337605.
256-Rank Error Analysis:
TSVD Error: 0.851487.
QR Error: 1.145675.
QR Error (with pivoting): 1.140214.
Randorth Error: 22.324762.
512-Rank Error Analysis:
TSVD Error: 0.710144.
QR Error: 1.114940.
QR Error (with pivoting): 1.244186.
Randorth Error: 22.349314.
Discrepancy analysis found beta = 31.649878, with error
```

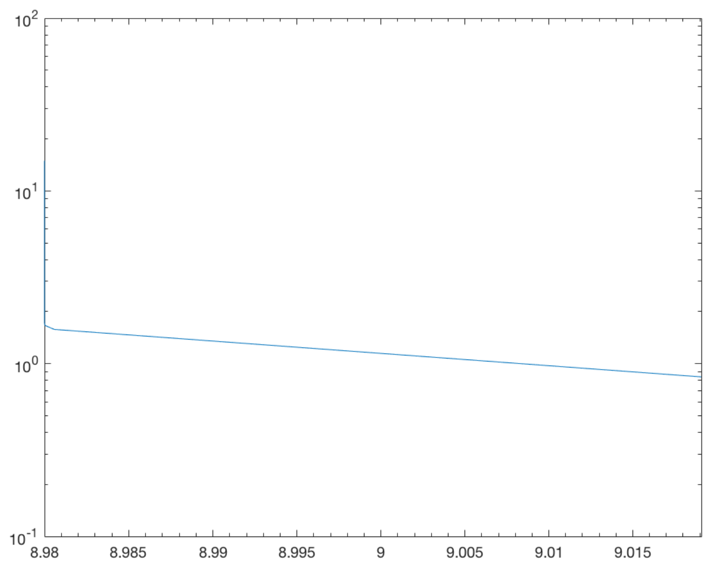

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600

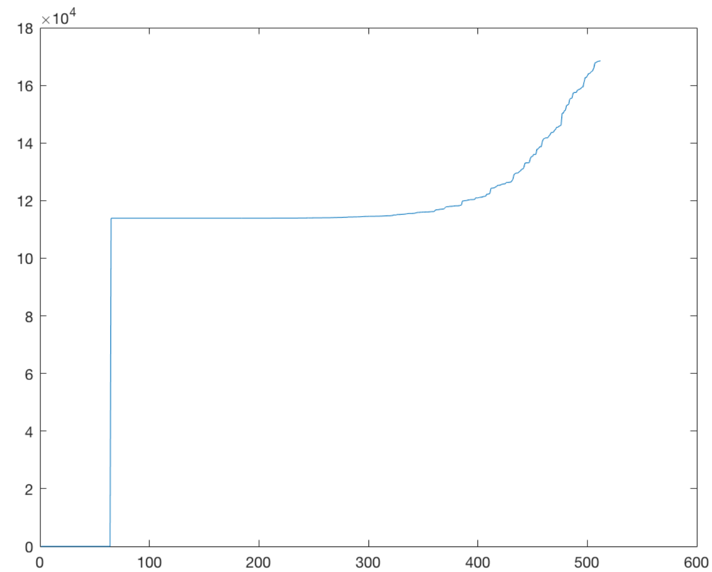

1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
45.7215
45.722
45.7225
45.723
45.7235

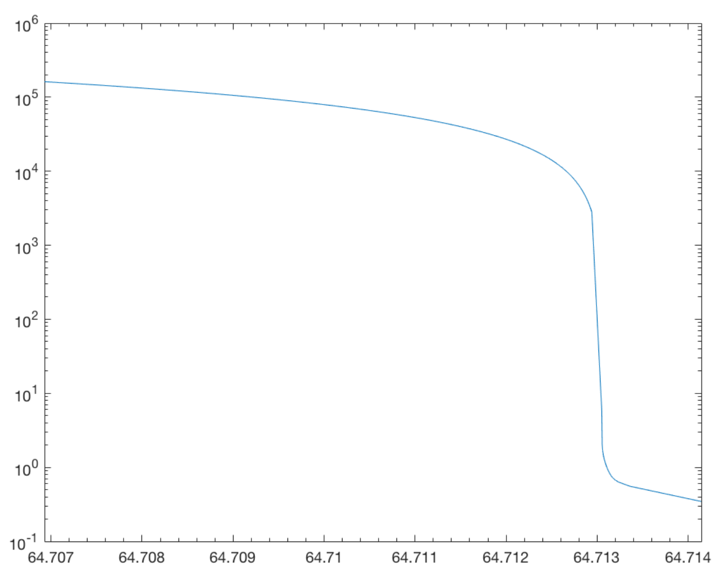

3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600

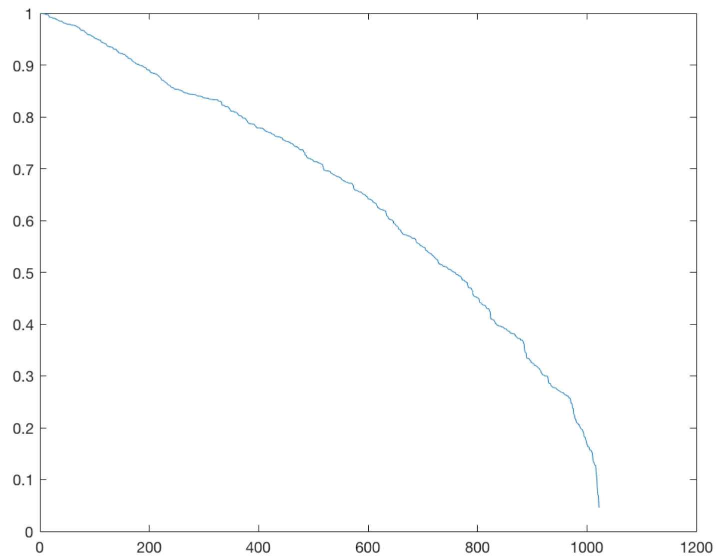

10^0
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0

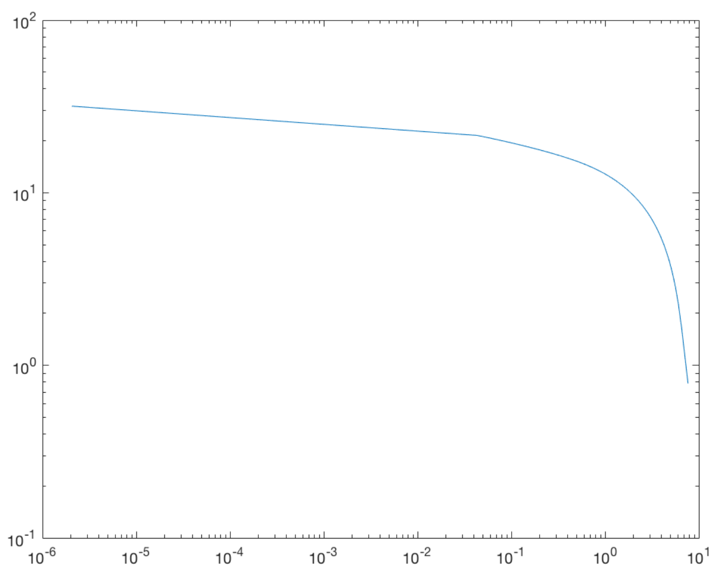

1
0.98
0.96
0.94
0.92
0.9
0.88
0 10 20 30 40 50 60 70 80 90 100


10^2
10^1
10^0
10^{-1}
8.98 8.985 8.99 8.995 9 9.005 9.01 9.015


×10^4
18
16
14
12
10
8
6
4
2
0
0
100
200
300
400
500
600

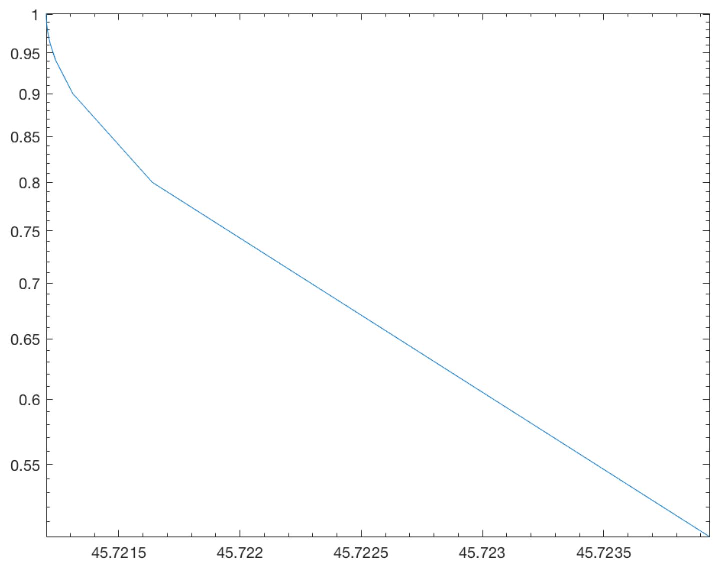

10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^-1
64.707
64.708
64.709
64.71
64.711
64.712
64.713
64.714

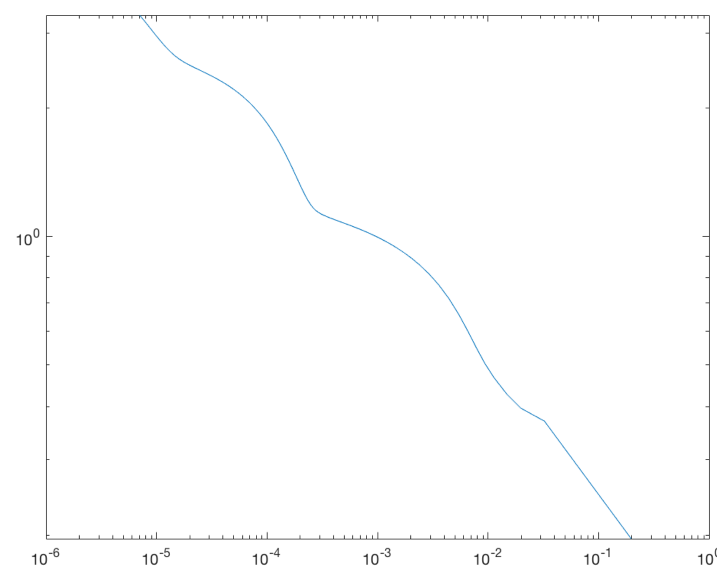


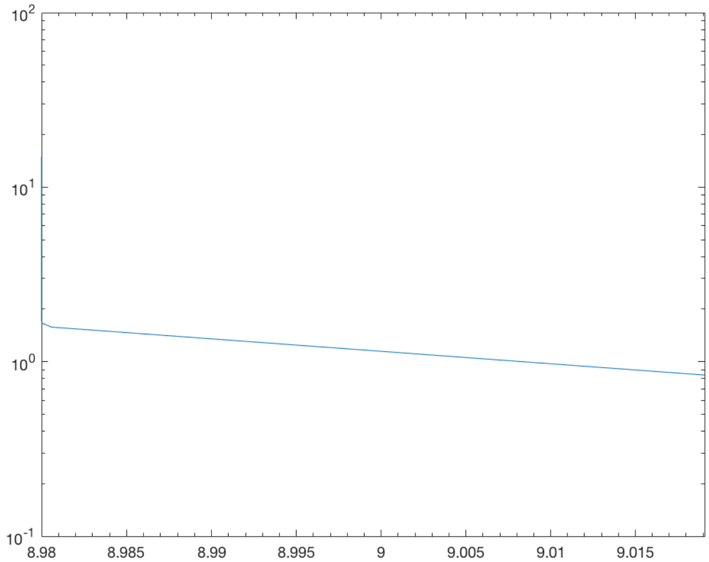


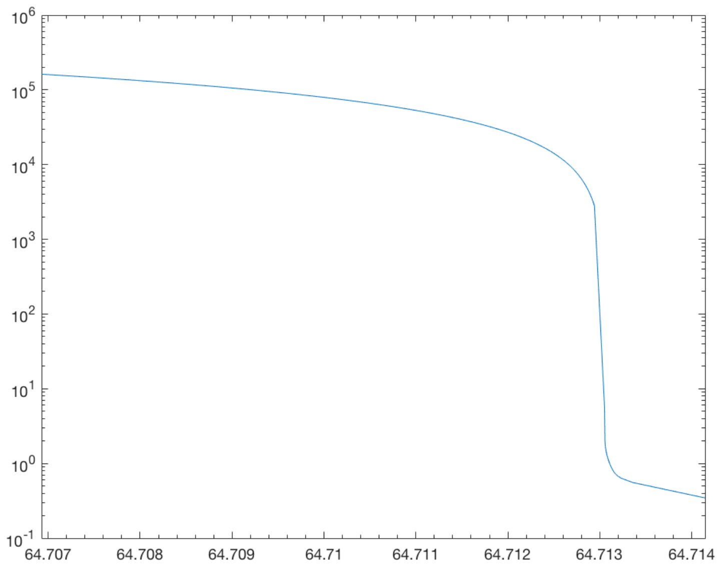
Published with MATLAB® R2016b

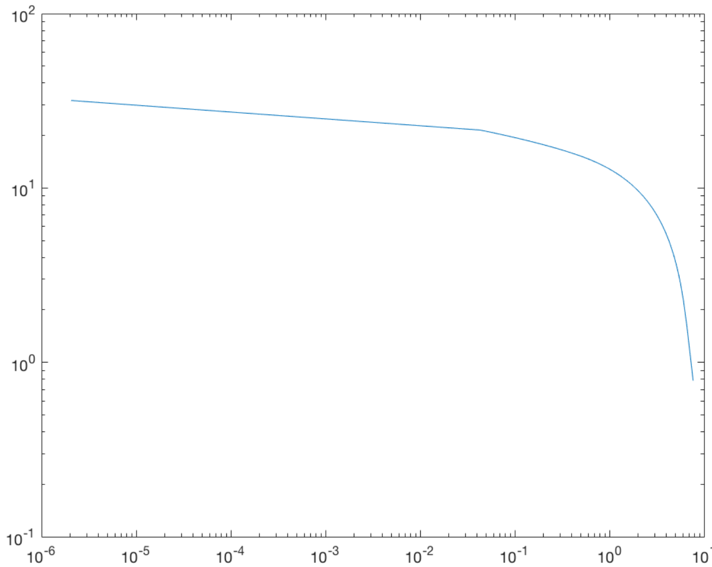
```
for i = 1:5
    [A, b, tau, xstar] = problem(i);
    [m, n] = size(A);

    l_curve(A, b, tau, xstar);
end
```


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
45.7215
45.722
45.7225
45.723
45.7235


10^0
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0


10^2
10^1
10^0
10^-1
8.98
8.985
8.99
8.995
9
9.005
9.01
9.015


10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^-1
64.707
64.708
64.709
64.71
64.711
64.712
64.713
64.714



Published with MATLAB® R2016b