CS 322 Homework 8 — Solutions

out: Thursday 26 April 2007
due: Fri 4 May 2007

In this homework we'll look at propagation of error in systems with $n \times n$ Hilbert matrices $\mathbf{H}_n$, which look like this:

$$\mathbf{H}_4 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

These matrices are ill-conditioned and get worse quickly as n increases.

Relevant Matlab functions for this homework include `hilb`, `cond`, `chi2pdf`, `chi2cdf`, and `chi2inv`.

Problem 1: Conditioning of linear systems.

1. One can study the propagation of error from the right-hand side to the solution in a linear system using a brute-force Monte Carlo experiment: add random errors artificially to the right hand side and see how much error they induce in the solution.

   Using the following procedure, estimate the condition numbers of $\mathbf{H}_2$ and $\mathbf{H}_5$.

   - Choose 10,000 random right-hand sides for the system $\mathbf{H}_n\mathbf{x} = \mathbf{b}$.
   - Perturb each right-hand side by a normally distributed random error with standard deviation 10^{-6}.
   - Solve the original and perturbed systems and compare the results.
   - Compare the relative errors in the right-hand sides and the solutions.

   Illustrate your work with a short Matlab transcript.

   For H_2 you should be able to estimate the true condition number pretty well. How good is your estimate for H_5 and why?

   Hint: You can do this experiment in about 5 Matlab statements.

   Answer::

```
k = 2;
H = hilb(k);
b = 2*rand(k, 10000) - 0.5;
e = 1e-6*randn(k, 10000);
max((sqrt(sum((H\(b+e)-H\b).^2,1))./sqrt(sum((H\b).^2,1)))...
        .* (sqrt(sum(b.^2,1)) ./ sqrt(sum(e.^2, 1))) )
cond(H)
```

This code returns around 19.196 for $k = 2$ and 1598.7 for $k = 5$. Compare this to the true condition numbers of 19.281 for $\mathbf{H}_2$ and 476607 for $\mathbf{H}_5$. We can see that our estimate for $\mathbf{H}_2$ is quite good; however, our estimate for $\mathbf{H}_5$ is off by several orders of magnitude. This can be attributed to two factors: one, we are randomly exploring a higher dimensional space (5 compared to 2), and two, the poor conditioning of $\mathbf{H}_5$ means that we need to get very close to the direction of the worst case RHS/perturbation in order to get close to the worst case error growth. As a result, it is relatively unlikely that we will randomly find the worst case error growth for $\mathbf{H}_5$ in only 10000 samples.

2. Use the SVD to find a right hand side and perturbation that demonstrate error growth that achieves the bound given by the condition number.

   Answer: Taking the SVD of $\mathbf{H}_2$, we get $\mathbf{U\Sigma V}^T\mathbf{x} = \mathbf{b} + \mathbf{e}$. We would like to choose $\mathbf{b}$ and $\mathbf{e}$ such that the relative growth of error is maximized. This can be done by choosing $\mathbf{b}$ to be a vector whose norm shrinks the most when multiplied by $\mathbf{H}_2^{-1}$ and $\mathbf{e}$ to be a vector whose norm increases the most when multiplied by $\mathbf{H}_2^{-1}$. Looking at the SVD, we see that $\mathbf{U}_1$ (the first singular vector in $\mathbf{U}$) will be multiplied by $\frac{1}{\sigma_1}$ and so it will shrink, while $\mathbf{U}_2$ will be multiplied by $\frac{1}{\sigma_2}$ and so it will grow. Thus, we choose $\mathbf{b} = \mathbf{U}_1$ and $\mathbf{e} = \mathbf{U}_2$. This is confirmed by testing in MATLAB:

```
H = hilb(2);
[U,S,V] = svd(H);
b = U(:, 1);
e = 1e-6*U(:, 2);
(norm(H\(b+e) - H\b) / norm(H\b)) * norm(b) / norm(e)
```

   which returns 19.2815 for the growth of error for $\mathbf{H}_2$. We can extend this to $\mathbf{H}_5$ by setting $\mathbf{e} = \mathbf{U}_5$, obtaining 476607 for the growth of error for $\mathbf{H}_5$

*3. Repeat question 1, but this time perturb the matrix instead of the right-hand side. Does the condition number still bound the growth of error when it propagates from the matrix to the solution?

   Answer: The code is extremely similar and not repeated here, but you should see that the condition number still bounds the growth of error when it propagates from the matrix to the solution.

*4. Repeat question 2, but this time work out the worst-case perturbation to the matrix.

Answer: The matrix perturbation is $\mathbf{E} = \mathbf{U}_k \sigma_k \mathbf{V}_k$, where $k = 2$ or 5 depending on the size of $\mathbf{H}$, and the right hand side is $\mathbf{U}_k$, i.e. the last singular vector

Problem 2: The χ^2 distribution.

1. Use Monte Carlo to estimate the probability that the norm of a point $\mathbf{x}$ drawn from a 3D Gaussian distribution with unit standard deviation has a norm greater than 2. How many samples do you need to get 3 significant figures of accuracy?

   Answer:

   ```
   x = randn(3, 1000000);
   nx = sqrt(sum(x.^2, 1));
   ind = find(nx > 2);
   size(ind,2)/1000000
   ```

   This returns a probability of about 0.2618. In testing, it seems as if about a million samples are needed in order to reasonably guarantee 3 significant figures of accuracy.

2. Use Matlab's built in χ^2 functions to compute the same number.

   Answer: Using `1 - chi2cdf(4, 3)`, we obtain 0.2615. Note that we are using $4 = 2^2$ instead of 2, because the χ^2 distribution is measuring the 2-norm squared (i.e. the square of length).

3. Let $\mathbf{y} = \mathbf{H}_3\mathbf{x}$. Give a 90% confidence region for $\mathbf{y}$ (specified as the axes of the ellipse) and for its first component, y_1.

   Answer:

   ```
   [U,S,V] = svd(hilb(3));
   >> rad = sqrt(chi2inv(.9, 3));
   >> Ellipse = rad*U*S

   Ellipse =

     -2.9122   0.1674   0.00086
     -1.6193  -0.1616  -0.00480
     -1.1384  -0.1985   0.00463
   ```

```
>> Individual = sqrt(chi2inv(.9, 1))*U*S;
>> y1 = norm(Individual(1,:))

y1 =

   1.9190
```

Problem 3: Confidence in least squares fits.

In this problem, no fair using Monte Carlo except where specified. Specify all confidence ellipses as the axes of the ellipse.

1. The file `hw08data1.txt` contains 20 (x, y) points in which all the ys have independent Gaussian noise with $\sigma = 0.05$.

   (a) Fit the model $y = a_1 x + a_2$ to the data using linear least squares.
   Answer:$a_1 = 0.3029$ and $a_2 = 0.5036$ using standard least squares fitting.

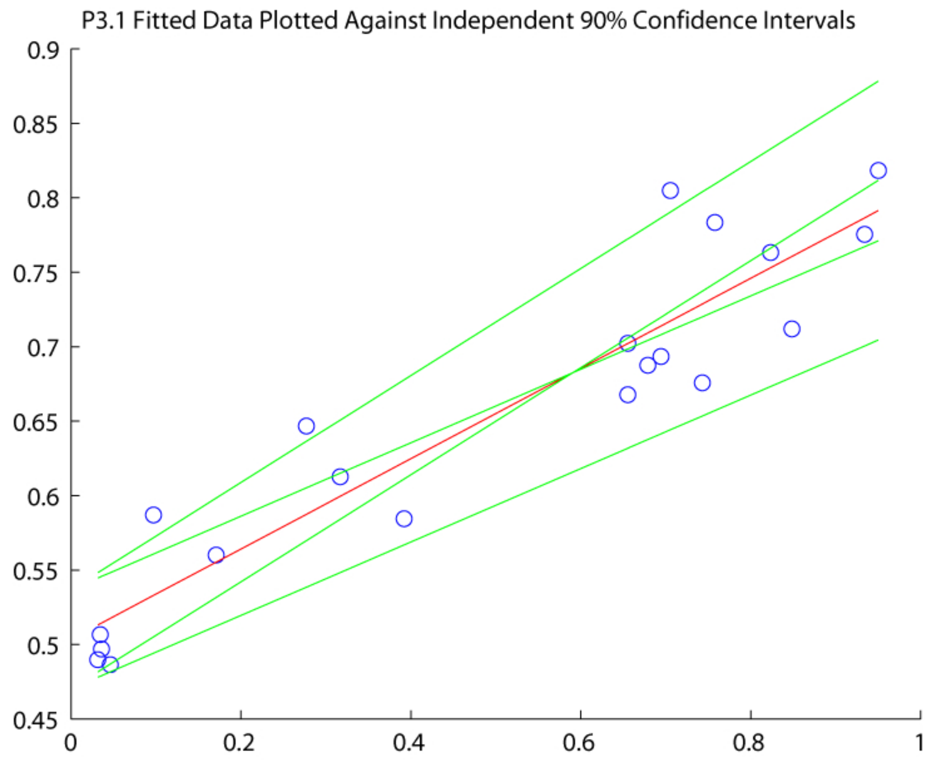
   (b) Give 90% confidence intervals for a_1 and a_2. Plot the lines corresponding to the four extreme combinations of parameters against the data points.
   Answer:

   ```
   >> M = [xs, ones(20, 1)];
   >> [U,S,V] = svd(M,0);
   >> rad = 0.05*sqrt(chi2inv(0.9, 1));
   >> Individual = rad*V*S^-1;
   >> ar = sqrt(sum(Individual .^2, 2))

   ar =

      0.05641
      0.03332
   ```

   These are the radii for the individual confidence intervals, so the actual 90% confidence intervals are $a_1 \in [0.2465, 0.3593]$ and $a_2 \in [0.4702, 0.5369]$. Note that there was some confusion in office hours about whether the distribution used should have 1 or 2 degrees of freedom. The correct answer uses 1 degree of freedom, but we accepted answers that computed the values using 2 degrees of freedom.
   We note that we can see $\mathbf{V}\mathbf{\Sigma}^{-1}$ are the important axes by multiplying both sides of $\mathbf{M}\mathbf{a} = \mathbf{y}_s$ by $\mathbf{M}^+$, ending up with $\mathbf{a} = \mathbf{V}\mathbf{\Sigma}^{-1}\sqrt{0.05^2\text{chi2inv}(0.9, 1)}$ (since $\mathbf{U}$ does not change the distribution of noise at all).



Looking at Figure (1b), we can see the original fit in red and the four green lines corresponding to the four extreme choices of parameters from our 90% confidence intervals.

(c) Confirm your intervals using Monte Carlo.

Answer:

```
k = 10000;
ybig = repmat(av(1)*xs + av(2), 1, k) + 0.05*randn(20, k);
parambig = A\ybig;
size(find(abs(parambig(1,:) - a1) <= ar(1)),2) / k
size(find(abs(parambig(2,:) - a2) <= ar(2)),2) / k

ans =

   0.8973


ans =
   0.9011
```

From this we can see that our parameters computed using Monte Carlo fall within our 90% confidence interval just about 90% of the time, as expected

(d) Give a 90% confidence ellipse for (a_1, a_2) together. Plot the lines corresponding to the ends of the principal axes of the ellipse against the data points.

Answer:

```
>> rad = 0.05*sqrt(chi2inv(0.9, 2));
>> Ellipse = rad*V*Sˆ-1

Ellipse =

  -0.01010   0.07290
  -0.01879  -0.03920
```

We can check these using Monte Carlo (not shown, but similar to the above code), observing that it matches closely with experimental results.

We can plot this using

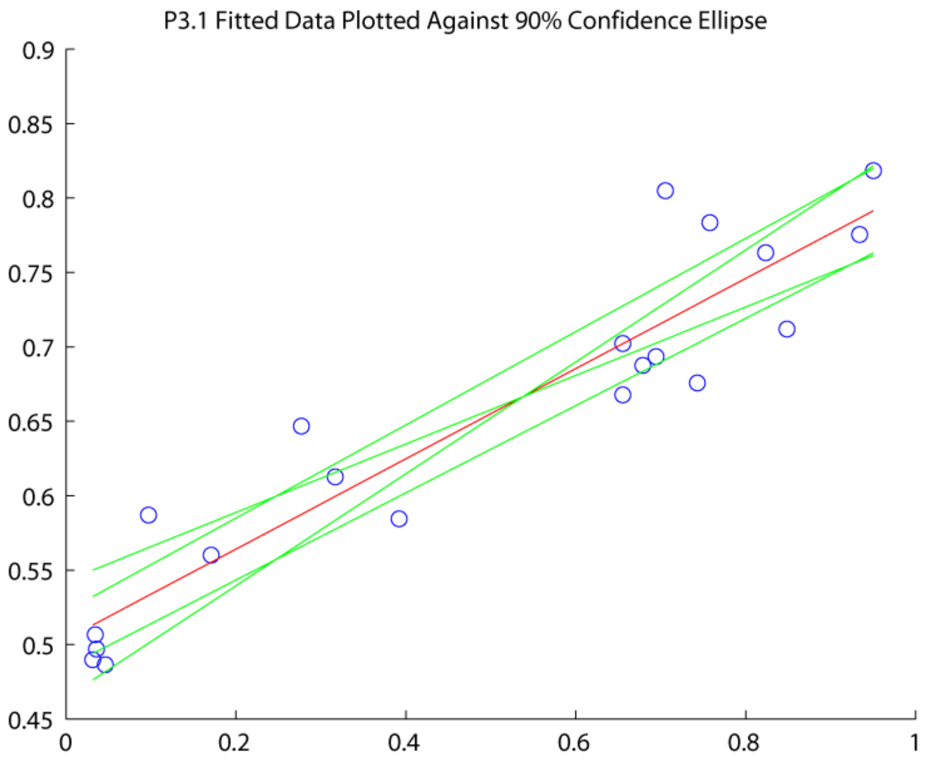
```
>> plot(xr, xr*(a1+Ellipse(1,1)) + a2+Ellipse(2,1), 'g-')
>> plot(xr, xr*(a1-Ellipse(1,1)) + a2-Ellipse(2,1), 'g-')
>> plot(xr, xr*(a1-Ellipse(1,2)) + a2-Ellipse(2,2), 'g-')
>> plot(xr, xr*(a1+Ellipse(1,2)) + a2+Ellipse(2,2), 'g-')
```

obtaining figure (1d), we can see the original fit in red and the four green lines corresponding to the four choices of parameters from each axis of the ellipse. We note that the values in general are closer to the computed fit than the lines in part (b).

*2. The file `hw08data1.txt` contains 20 (x, y) points in which all the ys have independent Gaussian noise with $\sigma = 0.01$.

(a) Fit the model $y = a_1 e^{a_2 x}$ to the data using linear least squares.

Answer: We note that $\ln y = \ln a_1 + a_2 x$, allowing us to solve for the terms a_2 and $\ln a_1$. We then take $e^{\ln a_1}$ in order to obtain the parameter a_1. One other issue is the fact that the uncertainties ($\sigma = 0.01$) in the y values are not the same in log space — the logs of small values will have their uncertainties magnified. For small errors like these, we assume that the error is magnified by the derivative of $\ln y$, i.e. $\frac{1}{y}$; as a result, we have to weight each data point by its y value.


P3.1 Fitted Data Plotted Against 90% Confidence Ellipse
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0
0.2
0.4
0.6
0.8
1

```
>> lny = log(ys);
>> M = [ones(20, 1), xs];
>> sigmas = 0.01 ./ ys;
>> A = diag(1./ sigmas) * M;
>> b = diag(1./ sigmas) * lny;
>> ar = A \ b;
>> a1 = exp(ar(1));
>> a2 = ar(2);
```

This gives us $a_1 = 14.167$ and $a_2 = -2.956$

(b) Give 90% confidence intervals for a_1 and a_2. Plot the curves corresponding to the four extreme combinations of parameters against the data points.

Answer:

```
>> [U,S,V] = svd(A,0);
>> rad = sqrt(chi2inv(0.9, 1));
>> Individual = rad*V*S^-1;
>> arad = sqrt(sum(Individual .^2, 2))

arad =

   0.2289
   0.1831
```

Note that this gives us the interval of $\ln a_1$ and a_2, not a_1 and a_2. To compute the interval for a_1, we note that $(\ln a_1) + r$ is equal to $(\ln a_1) + \ln e^r = \ln(a_1 e^r)$, and similarly $(\ln a_1) - r$ is $\ln\left(\frac{a_1}{e^r}\right)$. Thus, our confidence intervals are $a_1 \in [11.268, 17.809]$ and $a_2 \in [-3.139, -2.773]$

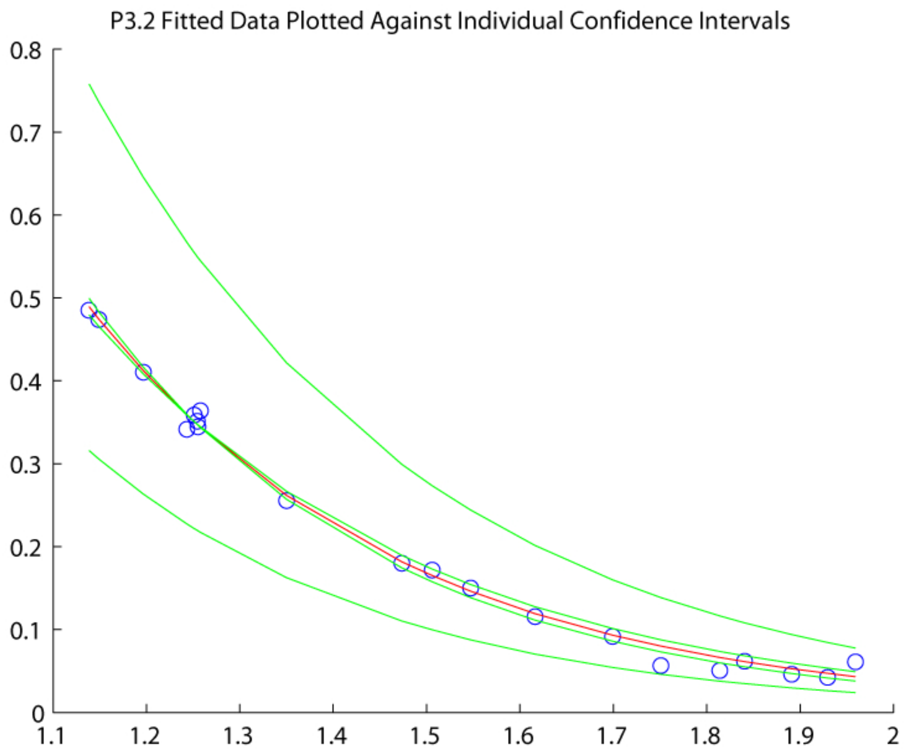
(c) Give a 90% confidence ellipse for (a_1, a_2) together. Plot the curves corresponding to the ends of the principal axes of the ellipse against the data points.

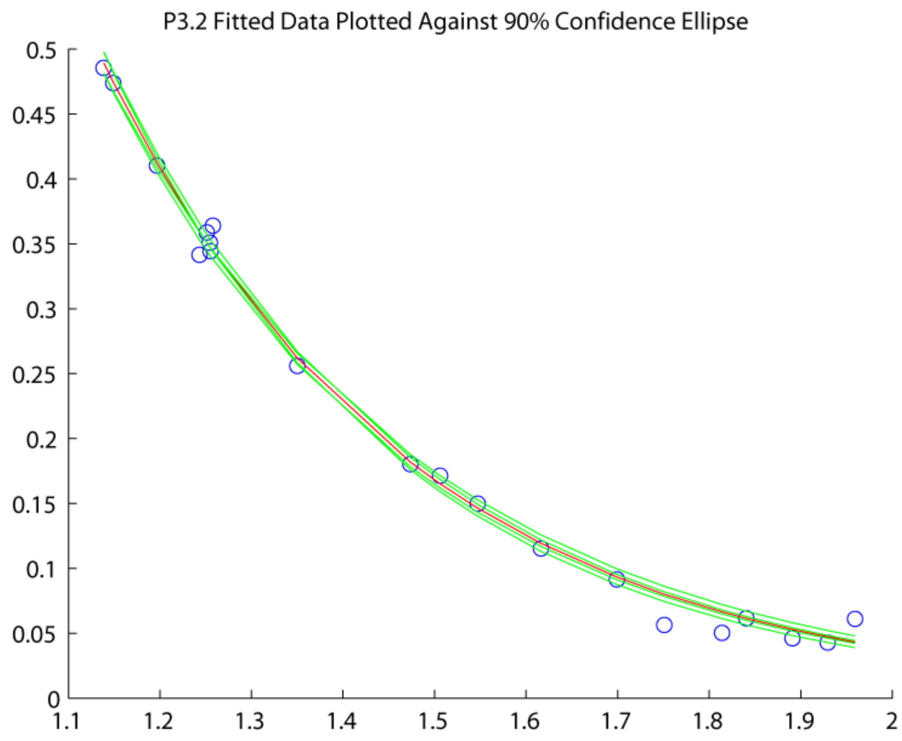
Answer:

```
>> rad = sqrt(chi2inv(0.9, 2));
>> Ellipse = rad*V*S^-1

Ellipse =

  -0.0070  -0.1814
  -0.0087   0.1450
```


P3.2 Fitted Data Plotted Against Individual Confidence Intervals
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2



```
>> Ellipse(1, :) = exp(Ellipse(1,:))

Ellipse =

   0.9930   0.8341
  -0.0087   0.1450
```

Looking at Figure (1b), we can see the original fit in red and the four green lines corresponding to the four extreme choices of parameters from our 90% confidence intervals.

(d) Find the 90% confidence ellipse using a Monte Carlo simulation. Why might the result be different from the previous part?

The code to do this is similar to the previous Monte Carlo simulation. However, in this case the result might be different because there is a nonlinear operation (natural logarithm) applied as part of the fit, while the computations above assume that only linear operations are applied to the Gaussian noise.