SF2520 - Laboratory 7

Goyens Florentin & Weicker David

26th November 2015

Introduction

This report answers the question given in the seventh laboratory for the course SF2520 - "Applied Numerical Methods". This laboratory is in the numerical algebra part. The first question concerns a stable QR-factorization for ill-posed systems and the second is about solving the least squares problem using a QR-factorization.

1 Stable QR-factorization

In this part of the homework we use the QR-factorization to solve a square ill-conditioned system. We use the given function `illposed.m` to generate an ill-conditioned system of size 8.

a) Estimate condition number of a linear system Given the matrix A, we can introduce a perturbation $b + \Delta b$ to estimate the condition number of A with

$$\frac{\| \Delta x \|_2}{\| x \|_2} \leq \kappa(A) \frac{\| \Delta b \|_2}{\| b \|_2}.$$

We took a large number of random perturbations and kept the one that gave the largest estimation of the condition number. The estimation of the condition number for a matrix A is in the table below. Taking another random matrix A give very similar results.

`cond(A)`	`estimateCond(A)`
1.67e+12	5.05e+11

Table 1: Estimated condition number of A compared with Matlab

It is clear that we find a lower bound of the exact condition number since we can only try a finite number of perturbations.

b) Approximate solution of an ill-posed system The script `qrFact.m` solves the task at hand. We solve the system for different values of the rank r. We define the residual $\text{res} = A\hat{x} - b$. Each time we estimate the condition number of R_{11} to compare it with the value of the Matlab function `cond`. We compute other values that appear in the table below.

rank r	$\| E \|$	$\| res \|$	cond(R11)	estimationCond(R11)	$\| \hat{x} \|$
8	/	$8.8818 * 10^{-16}$	$1.6882 * 10^{12}$	$0.8736 * 10^{12}$	1.701
7	$5.3835 * 10^{-12}$	$0.3267 * 10^{-11}$	$2.8435 * 10^{9}$	$1.2983 * 10^{9}$	2.202
6	$0.2064 * 10^{-8}$	$0.2878 * 10^{-9}$	10.8204	5.3686	3.212
5	0.5647	0.4753	6.8978	3.2608	2.676

Let's analyze the different numerical relations. The value $\| E \|$ increases as the rank decreases. This makes perfect sense since when decreasing the rank we set more elements to

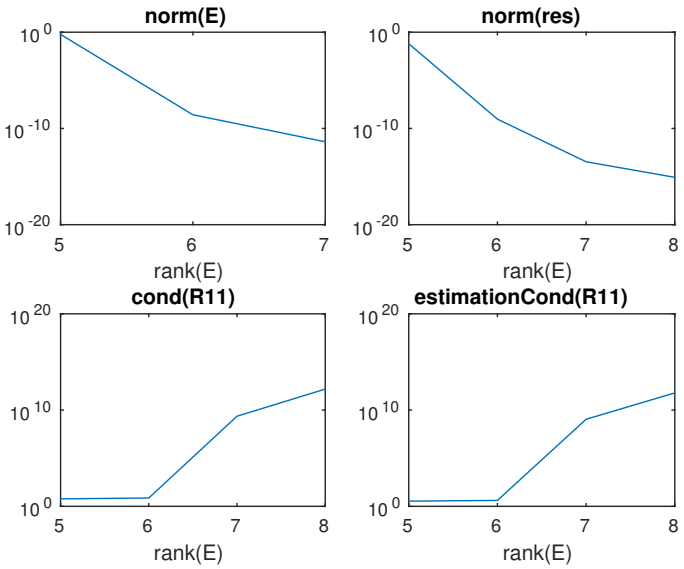

Figure 1: Numerical relations as a function of the rank r

zero and E has more entries. On the other hand, decreasing the rank reduces the quality of the approached solution and thus we get a larger residual. We commit a less accurate approximation by setting more elements to zero. We can note that the norm of the residual should be zero for $r = 8$ (as we solve the system exactly). The value obtained (around 10^{-16}) is due to floating point errors and should be considered zero.

For the estimation of the condition number, we have again a lower bound of the exact value (let's assume that the value given by Matlab is close enough to the exact value). We are not so far away from the value of Matlab. At least the tendency of the condition number with r is correct as we also see on the two lower graphs of figure 1. This number is decreasing as the rank decreases. This again makes sense because the whole point of setting entries to zero is to solve a problem that is less sensitive. Finally the norm of the solution $\hat{x}$ is not related to the rank r. We just note that the steps applied lead to the solution with the smallest norm (since the problem with modified rank does not have an unique solution).

On figure 1, we can see that the "best" choice for r is 6. We can see that for this value, the condition number is quite small (around 10^1) and the approximation is quite good because the norm of the residual is around 10^{-10}. This seems to be the best trade-off between the quality of the approximation and the sensitivity of the solution.

2 Least Squares with QR-factorization

This section tries to fit the data given with the least squares method. The function used to fit the data is :

$$y(t) = c_1 + c_2 e^{\alpha t} + \sum_{k=1}^{n} (a_k cos(\frac{2\pi kt}{12}) + b_k sin(\frac{2\pi kt}{12}))$$

The problem consists thus of finding c_1, c_2, a_k and b_k that minimize $||Ax - b||_2$ where:

$$A = \begin{pmatrix} 1 & e^{\alpha t_1} & cos(\frac{2\pi k_1 t_1}{12}) & \dots & cos(\frac{2\pi k_n t_1}{12}) & sin(\frac{2\pi k_1 t_1}{12}) & \dots & sin(\frac{2\pi k_n t_1}{12}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & e^{\alpha t_m} & cos(\frac{2\pi k_1 t_m}{12}) & \dots & cos(\frac{2\pi k_n t_m}{12}) & sin(\frac{2\pi k_1 t_m}{12}) & \dots & sin(\frac{2\pi k_n t_m}{12}) \end{pmatrix}$$

$$x = (c_1\, c_2\, a_1\, \dots\, a_n\, b_1\, \dots b_n)^T$$

And $b \in \mathbb{R}^{m\times 1}$ is the vector containing the data at times t_i with $i = 1...m$.

This problem can be solved using the least squares method. To do this, we are going to use the QR-factorization, i.e. the fact that:

$$A = QR$$

With Q an orthogonal matrix and R upper triangular. Indeed, to minimize the norm of the residual, we know that we have to solve the normal equations :

$$A^T Ax = A^T b$$

But using QR-factorization allows us to say (the proof can be found in the course notes) :

$$Rx = Q^T b$$

The Matlab code used to solve the problem can be found at the end of this report. We define the residual as res $= A\mathbf{x} - b$ where $\mathbf{x}$ is the solution of the normal equations.

For $n = 1$, here are the values obtained for the parameters :

c_1	2.5581
c_2	334.9964
a_1	-1.7186
b_1	2.3581

And the computed norm of the residual is :

$$||\text{res } (n = 1)||_2 = 15.2036$$

For $n = 2$, the parameters are the following :

c_1	2.7340
c_2	334.8298
a_1	-1.7185
a_2	0.8382
b_1	2.3579
b_2	-0.0345

And the norm of the residual is :

$$||\text{res } (n = 2)||_2 = 11.3936$$

And finally, for $n = 3$, these are the parameters :

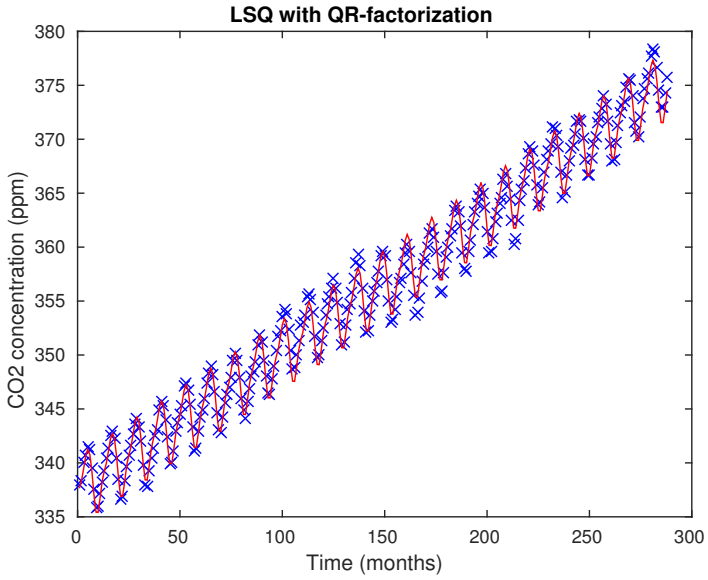

Figure 2: Fitting of the data for $n = 3$

c_1	2.7665
c_2	334.7990
a_1	-1.7185
a_2	0.8382
a_3	0.1094
b_1	2.3579
b_2	-0.0345
b_3	-0.0568

And the norm of the residual is :

$$||\text{res } (n = 3)||_2 = 11.2972$$

It is important to note that the norm of the residual decreases as we increase n. That is to be expected since when we increase n by one, we allow two new degrees of freedom. Thus, it is not possible to obtain a "worse" residual with a higher n. In the worst case scenario, the two new basis functions do not increase the quality of the fitting and then the corresponding coefficients would be zero. The residuals would then be equal.

Figure 2 shows the fitting for $n = 3$. We can see that it is relatively good. The $'\times'$ are the data points and the red curve is the fitting.

3 Code Listing

```
function [table, condA, estimCondA] = qrFact()
% qrFact Solve the task one for the lab7 that is the analysis of the
% solution for an ill-posed system with qr-factorization
%
% table(:,1) is norm(E) for rank = [8,7,6,5]
% table(:,2) is norm(res) for rank = [8,7,6,5]
% table(:,3) is cond(R11) for rank = [8,7,6,5]
% table(:,4) is estimationCond(R11) for rank = [8,7,6,5]
% table(:,5) is norm(xhat) for rank = [8,7,6,5]
%
% Authors : Weicker David & Goyens Florentin
close all;
format long
[A,b] = illposed(8);

table = zeros(4,5);

[q,r,p] = qr(A);
d = q'*b;
condA = cond(A);
estimCondA = estimationCond(A,b,8);
rank = 8:-1:5;
for i=1:4
    % compute norm(E)
    table(i,1) = norm(r(rank(i)+1:8,rank(i)+1:8));
    % compute cond(R11)
    r11 = r(1:rank(i),1:rank(i));
    table(i,3) = cond(r11);
    % compute estimationCond(R11)
    dhat = d(1:rank(i));
    table(i,4) = estimationCond(r11,dhat,rank(i));
    % compute norm(xhat)
    yhat = r11\dhat;
    xhat = p*[yhat;zeros(8-rank(i),1)];
    table(i,5) = norm(xhat);
    % compute norm(res)
    table(i,2) = norm(A*xhat-b);
end

titre = {'norm(E)','norm(res)','cond(R11)', 'estimationCond(R11)'};
%Plot the results
for i=1:4
    subplot(2,2,i);
    semilogy(rank,table(:,i));title(titre{i});
    xlabel('rank(E)');
end
end

function [condA] = estimationCond(A,b,n)
% Estimates the condition number of matrix A with variation on b
% n is the size of A

condA = 0;
for j=1:1000
    deltaB = 1e-8*(rand(n,1)+1);

    x = A\b;
    y = A\(b+deltaB);

    condA = max(condA,norm(b)*norm(x-y)/(norm(deltaB)*norm(x)));
end
end
```

```
function [res,param] = lsp()
%LSP Solves the least squares problem with the basis functions given in
% the assignment.
% res = norm of the residual for n=1:3
% param = contains the values of the parameters a_i,b_i,c_i
%
% Authors : Weicker David & Goyens Florentin
close all;

mauna = load('mauna.dat');
Q = mauna(:,2:13)';
b = Q(:);
m = length(b);
t = (1:m)';

alpha = 0.00037;
e = ones(m,1);
res = zeros(3,1);
param = zeros(8,3);

for n=1:3
    A = [e exp(alpha*t) cos(2*pi*t*(1:n)/12) sin(2*pi*t*(1:n)/12)];
    [q,r] = qr(A);
    x = r\(q'*b);
    res(n) = norm(A*x-b);
    param(1:2+2*n,n) = x;
end

%Plot for n=3
plot(t,b,'bx',t,A*x,'r-');title('LSQ with QR-factorization');xlabel('Time (months)');
ylabel('CO2 concentration (ppm)');

end
```