Lab 2 (NLA 326/626)

1. Self-study the stability of least squares algorithm. Implements the all experiments and examples on the lecture 19. Paste the numerical results and matlab commands at each step of the experiments. Give comments on each numerical results (Use your own word to explain the results and avoid to verbatim the content on the book)

A=[];

for i=1:n

A=[A t.^(i-1)];

end

b=exp(sin(4\*t));

b=b/2006.787453080206;

**%The idea behind above is about least square fitting**

**%of the function exp(sin(4\*t)) on the interval [0,1]**

**%by the polynomial of degree 14.**

x=A\b;

y=A\*x;

kappa=cond(A)

theta=asin(norm(b-y)/norm(b))

eta=norm(A)\*norm(x)/norm(y)

**%below is the table**

b\_y=1/cos(theta)

A\_y=kappa/cos(theta)

b\_x=kappa/cos(theta)/eta

A\_x=kappa+kappa^2\*tan(theta)/eta

*kappa =*

*22.7177721511115e+009*

*theta =*

*3.74611140817419e-006*

*eta =*

*210.355963692202e+003*

*b\_y =*

*1.00000000000702e+000*

*A\_y =*

*22.7177721512709e+009*

*b\_x =*

*107.996805759746e+003*

*A\_x =*

*31.9086572964328e+009*

**%Householder Triangul**

[Q,R]=qr(A,0);

x=R\(Q'\*b);

x\_15\_1=x(15)

**%The x\_15 has a relative error which is mainly caused by epsilon(machine) and the order**

**%of condition number of x with respect to perturbations in A. Thus the**

**%inaccuracy of x\_15 can be entirely explained by ill-conditioning, no**

**%instability. The algorithm is backward stable.**

[Q,R]=qr([A,b],0);

Qb=R(1:n,n+1);

R=R(1:n,1:n);

x=R\Qb;

x\_15\_2=x(15)

**%The result is the same with the last one, i.e. we have x\_15\_1=x\_15\_2**

x=A\b;

x\_15\_3=x(15)

**%because of the Matlab's operator \ makes use of QR factorization with column pivoting.**

**%Thus result x\_15\_3 is better.**

**%below is Gram-Schimidt Orthogonalization**

[Q,R]=mgs(A);

x=R\(Q'\*b);

x\_15\_4=x(15)

**%This result x\_15\_4 sult is very bad, the relative error is very big, so this algorithm**

**%is not stable since it produces matrices Q whose columns are nor accurately orthonormal**

**%Since the algorithm above depends on that orthonormality, it suffers accordingly**

[Q,R]=mgs([A b]);

Qb=R(1:n,n+1);

R=R(1:n,1:n);

x=R\Qb;

x\_15\_5=x(15)

**%This algorithm is stable because it only needs Q to be well-conditioned**

**%below if the normal equation**

x=(A'\*A)\(A'\*b);

x\_15\_6=x(15)

**%the x\_15\_6 is very bad, and this algorithm is not stable for**

**% below is the SVD**

[U,S,V]=svd(A,0);

x=V\*(S\(U'\*b));

x\_15\_7=x(15)

**%The solution of the full rank least squares problem by the SVD is backward stable.**

1. **19.2**

**%the output matrix X is the Pseudoinverse of A with the dimensions of n-m**

**%when A's dimensions is m-n.**

**%because some singular values of A (by the SVD of A) might be small, which**

**%will make the computation unstable, so all the singular values**

**%less than tol are treated as zero**

1. **The NCM program censusgui involves several different linear models. The data are the total population of the United States, as determined by the U.S. Census, for the years 1900 to 2000. The units are millions of people**

**t y**

**1900 75.995**

**1910 91.972**

**1920 105.711**

**1930 123.203**

**1940 131.669**

**1950 150.697**

**1960 179.323**

**1970 203.212**

**1980 226.505**

**1990 249.633**

**2000 281.422**

**The task is to model the population growth and predict the population when t =2010. The default model in censusgui is a cubic polynomial in t.**

1. **Determine the cubic polynomial p(t)**

**point =**

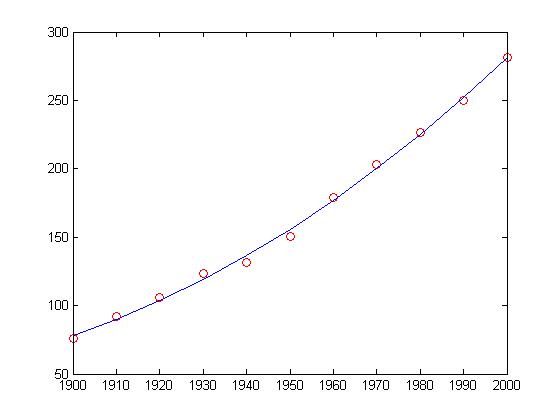
**Columns 1 through 9**

**78.2116 90.1096 103.8015 119.3134 136.6716 155.9022 177.0312 200.0849 225.0895**

**Columns 10 through 11**

**252.0710 281.0556**

1. **Plot the data and the fitting curve in the same graph**

****

1. **Compute the residue 2**

**residue =**

**10.1188**

1. **Predict the population when t =2010**

**predict\_t2010 =**

**312.0695**