*Report for Home Work 9:*

PART I:

1. As it is pointed out in the slides, we have the following procedure for

**Classical Gram-Schmidt (CGS):**

(1)

Now let's take vectors *Vi* and project them orthogonally onto the line spanned by vectors *Ui*:

(2)

.

.

.

are the system of orthogonal vectors, and the normalized vectors are .

* proof of orthogonality of each pair of *u* and orthonormality of each pair of *e*:

using the relation (1) and substituting , one can confirm that ()=0:

(3)

The same is true for :

(4)

The third term is zero as we have concluded in equation (4).

If then we assume that **any one** statement in the infinite sequence of statements is true (i.e. ()), then so is the **next** one:

(5)

As mentioned in the last statement, as we know that terms three to end is zero, then we have () equal to zero. So by mathematical induction, we have concluded that () for all is true. The exact same procedure holds for (). We proceed by induction on *k*.

Suppose we know () = 0 for *1 ≤ i < j ≤ k − 1*. Then for *i =k* ,

(6)

Since all the terms are 0 except when *i = j* , and we have :

(7)

Therefore, each pair of () are mutually orthogonal. As are nonzero orthogonal vectors, and we let , then are orthonormal vectors. Orthonormal means that the vectors are orthogonal and that they each have length 1.

**Modified Gram-Schmidt (MGS):**

* proof of arithmetic equality of CGS and MGS:

(8)

It has to be shown that of MGS is exactly similar to of CGS:

(9)

then

(10)

the third term on RHS:

(11)

As we know that and are orthogonal to each other, therefore . So,

(12)

Again by mathematical induction, we presume that the equality holds for and we

consider :

(13)

(14)

the above last term is zero, as and are orthogonal. Substituting the and doing the same procedure until reaching :

(15)

So the exact equality of MGS and CGS has been proved.

1. As it is stated in slide 18:

(16)

(17)

(18)

(There would be three concepts involved here which needs to be proved.

first:

second:

The inverse matrix of is given by . We may prove by showing that :

We have

Hence is the inverse of . Since the inverse matrix is unique, therefore .

third:

Let then interchanging rows and columns gives transosing it back yields A again .

***Now the main proof:***

As ;

1. Based on the guidance in the class, there are two approaches for solving this example:

*first: Using the CGS for constructing the orthogonal basis followed by orthonormalizing*

*Second: directly relating sin(x) to :*

Now performing the inner product by , then we have three equations and three unknown:

Now using the relation

Therefore, if we write the short form of A.C=b where C=[C1;C2;C3] and A and b are marix and vector of calculated items:

Then,

Result: Using the second approach, I have got:



PART II:

1. Explanation of lines:

> n=32; m=1000; setting the n and m to be 32 and 1000, respectively.

> j=0:n-1; j is defined to be from 0 to n-1; n values.

>sigma = 2.^(-j); sigma is a vector function of j and has n components and by using "." we defined it to be 2(-j) component wise.

> X = randn(n); By definition, this function generates normally distributed pseudorandom numbers. randn(n) returns an n x n matrix containing pseudorandom values drawn from the standard normal distribution.

> [u,s,v]=svd(X); This function produces a diagonal matrix s, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices u and v so that X = u\*s\*v'. Where unitary matrix is a matrix which satisfies A.AT=I or AT.A=I .

> norm(X-u\*s\*v'); This is the second norm of a vector: norm(A)=(Sum(A(:)^2))^(1/2). For matrix, it is also the square root of all squared elements of that matrix. Now as we know that X = u\*s\*v', then norm of X-u\*s\*v' should be very small number in the order of matlab's precision (~10^-15).

> X=u\*diag(sigma)\*v'; Firstly, diag(sigma) , when sigma is a vector with n components, is a squared diagonal n by n matrix in which the components of sigma are the components of the main diagonal and all other components are zeros. u and v' are the matrices which we have obtained before by using svd(). Then, "\*" is used to perform matrix by matrix multiplication operations. So now, X is the product of three n by n matrices; namely u, diag(sigma) and v' .

> cond(X); This function returns the second norm condition number of matrix X. (i.e. the ratio of the largest singular value of X to the smallest). Large condition numbers indicate a nearly singular matrix. The general formulation for cond(X,p) which is condition number based on p-norm is:

In the case here, as we have diag(sigma) as singular value of X and then we have . I've got

> [q,r]=chol\_qr\_it(X);

function [Q,R] = chol\_qr\_it(A)

i=0;

cn = 200;

Q = A; (1)

G = Q'\*Q; (2)

n = size(A,2);

R = eye(n); (3)

**while** cn > 100,

i = i + 1

[u,s,v]=svd(G); (4)

[q,r]=qr(sqrt(s)\*v'); (5)

R = r \* R; (6)

cn = sqrt(cond(s)); (7)

Q = Q \* inv(r); (8)

**if** cn>100

G = Q'\*Q; (9)

end;

end;

**return**

This line is a call to function chol\_qr\_it() which we have defined before. The lines of the function are explained briefly here:

(1) is to set the matrix which has been passed in (A) to the function as Q

(2) is to define G = QT.Q

(3) is to define R = In where n is the number of columns of A.

Now a while statement starts which continues until cn>100. Inside the loop:

(4) setting u,s and v as singular value decomposition of G.

(5) based on the fact that sqrt(An x n) is the n by n matrix of square root of A elements, we construct (sqrt(s)\*v'). Then a n orthogonal-triangular decomposition is performed (QR) on the constructed matrix.

(6) Now R is updated as R = r\*R in which r is the triangular part of previous decomposition.

(7) cn now is updated as taking the 2-norm condition number of marix s, followed by a square root of the result.

(8) Q is updated as Q=Q\*inv(r).

(9) Now if cn>100, G=QT.Q

End of while loop and end of function.

running this command resulted in 2 iterations.

> norm(X-q\*r); As we have done a kind of QR factorization, so we expect that the norm of X-q\*r would be again a small number in order of 10-15. In this case we have got 1.6923e-15.

> norm(eye(n) – q'\*q) We also expect that QT.Q=I , so again the norm of eye(n) – q'\*q should be in order of 10-15. The result was 5.9449e-14.

> tic, [q,r]=chol\_qr\_it(X); toc

> tic, [Q,R]=qr(X,0); toc

These two lines are to compute the elapsed time between performing chol\_qr\_it() function and qr() function. The result was 0.003567 for former and 0.000565 for latter.

1. The code is attached to the folder I have sent.

please compile it with *gcc qr.c -llapack -lcblas -lm*

|  |  |  |
| --- | --- | --- |
| Dimension of the matrix | Norm(X-Q\*R) | Norm(G-I) |
| 1000\*32 | 9.623092e-16 | 1.904367e-14 |
|  |  |  |
| 2000\*32 | 1.823188e-15 | 2.107398e-14 |
|  |  |  |
| 3000\*32 | 1.513431e-15 | 2.725891e-14 |