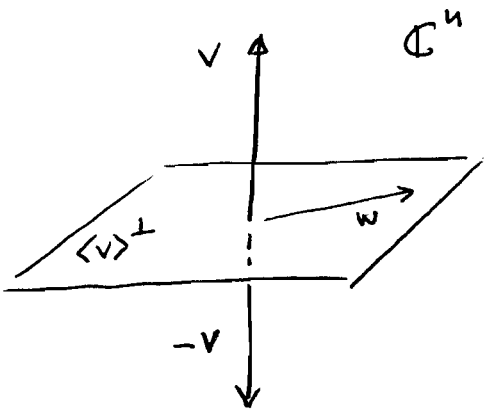
Problem Set IV



(10.1) $H_v = I - 2vv^*$; $\|v\|_2 = 1$

(a) $Hv = v - 2v(v^*v) = v - 2v = -v$

let $w \in \langle v \rangle^\perp$ ($\dim \langle v \rangle^\perp = n-1$, where $v \in \mathbb{C}^n$):

Then $v^*w = 0$, and $H_v w = w$

i.e. $\begin{cases} v \text{ ; eigenvalue } (-1) \quad (\text{multiplicity } 1) \\ w \in \langle v \rangle^\perp \text{ ; eigenvalue } (+1) \text{; geometric multiplicity } (n-1) \end{cases}$

As a reflector about a $(n-1)$-dimensional hyperplane, H reverses vector normal to that plane, while it leaves vectors on that plane unchanged.

(b) $\det H = (\text{product of eigenvalues}) = (-1)\cdot(+1)^{n-1} = -1$

(c) Since H is unitary ($H^* = I - 2(vv^*)^* = I - 2vv^* = H$ and $H^2 = I$), its singular values $\sigma_i = \sqrt{1^2}$, i.e.

$\sigma_i = 1$, $i = 1, \ldots, n$.

```
function [Q,R] = mgs(A)
% performs the modified Gram-Schmidt QR factorization of a full rank
% matrix A.
p=size(A);  %determine the dimensions of A
m=p(1);n=p(2);
V=A;         % This version does not destroy the matrix
R=zeros(p(2),p(2)); Q=zeros(p);  %ensure zeros in unaffected places
% if space is a problem, can store in one array.
%tic;flops(0);  % start flop counter and clock
for i = 1:n
   R(i,i) = norm(V(:,i),2);      % normalization of next projector direction
   Q(:,i) = V(:,i)/R(i,i);       % this is the stage of multiplying by the
                                 % diagonal matrix, prob. 8.3  (1)
   for j = i+1:n
      R(i,j) = dot(Q(:,i),V(:,j)); %this computes the entries of the upper
                                 % triangular matrix
      V(:,j) = V(:,j) - R(i,j)*Q(:,i); %(2)mult. by upper triang. matrix
   end
end
%toc; flops;
%  /   |   |   |   \ /1/r11          \ /1 -r12...-r1n \    /   |   |   |   \
%  |   |   |   |   | |  1            | | 1             |    |   |   |   |   |
%  |   |   |   |   | |   1    /\     | |  1            |    |   |   |   |   |
%  |   |   |   |   | |    1   \/     | |   1    /\     |    |   |   |   |   |
%  |   |   |   |   | |     1         | |    1   \/     |    |   |   |   |   |
%  | v1| v2| ...| vn| *|      1      |*|      .        | =  | q1| V2| ...| Vn|
%  |   |   |   |   | |        .      | |       .       |    |   |   |   |   |
%  |   |   |   |   | |  /\     .     | |  /\    .      |    |   |   |   |   |
%  |   |   |   |   | |  \/      .    | |  \/     .     |    |   |   |   |   |
%  |   |   |   |   | |           .   | |           .   |    |   |   |   |   |
%  \   |   |   |   / \           1/ \ |             1 /    \   |   |   |   /
%                    (1)              (2)
%
```

```
-------------------------------------------------------------------------
Path for all scripts:
~vageli/homew/504/HW4_5
-------------------------------------------------------------------------
                        Problem 10.2
------------ Scripts USED ( as defined in problem 10.2)
%-------------------------------------------------------------------------
%              ------< script test_house >------
%-------------------------------------------------------------------------
% script test_house
% tests the performance (flops, timing) of the Householder orthogonal
% triangularization routines (house.m, formQ.m).
% and the modified Gram-Schmidt function (mgs.m)
disp('   n           f1            f2            f3          t1          t2      t3')
disp('-------------------------------------------------------------------------')
format long e
for n = [10, 20, 40, 80, 160, 320, 640]
   A=randn(n,n);
   flops(0);tic;
   [W,R]=house(A);
   f1 = flops; t1 = toc;
   flops(0);tic;
   Q = formQ(W);
   f2 = flops; t2 = toc;
   E = A-Q*R;
   t1 = norm(E,inf);
   flops(0);tic;
   [Q1,R1] = mgs(A);
   f3 = flops; t3 = toc;
   E1 = A-Q1*R1;
   t2 = norm(E1,inf);
disp(sprintf('%4.0f  %10.0f  %10.0f  %10.0f  %15.10f %15.10f',n,f1,f2,f3,t1,t2))
end
%-------------------------------------------------------------------------
%              ------< function [W,R] = house(A) >------
%-------------------------------------------------------------------------
function [W,R] = house(A)
% performs the Householder orthogonal triangularization QR factorization of A
% in operational form. The matrix Q is available through the rank-1 (vector)
% modifiers of the identity, stored in the columns of W upon exit. A is saved
p=size(A);
m=p(1);n=p(2);
R=A;
W=zeros(m,m);
%tic;flops(0);
for k=1:n
   x=R(k:m,k);
   W(k:m,k) = x;
   a=sign(x(1));
   if ( a == 0 ) a=1; end
   a = a*norm(x,2);
   W(k,k)=a+W(k,k);
   W(k:m,k) = W(k:m,k)/norm(W(k:m,k),2);
   R(k,k) = -a;
   if (k < m) R(k+1:m,k)=0; end
   R(k:m,k+1:n) = R(k:m,k+1:n) - 2*W(k:m,k)*(W(k:m,k)'*R(k:m,k+1:n));
end
%toc; flops;
```

```
%-------------------------------------------------------------------------
%               ------< function Q = formQ(W,n) >------
%-------------------------------------------------------------------------
function Q = formQ(W,n)
% forms the Q matrix from the successive reflections in a Householder
% orthogonal triangularization. Two versions are given. They both take
% advantage of the character of a Householder reflections as a rank-1
% modification of the identity. The first version takes advantage
% of the successively decreasing number of nonzero entries in the
% reflection vectors W(:,k), while the second version does not.
% The second argument is optional, given if a reduced factorization
% is desired (defaults to m, the row dimension of A for a full factorization).
p=size(W);
m=p(1);
if nargin == 1
% full factorization is desired
   n = m;
end
%flops(0);tic;
Q=diag(ones(m,1));
for k = n:-1:1
    Q(k:m,k:m) = Q(k:m,k:m) - 2*W(k:m,k) * (W(k:m,k)' * Q(k:m,k:m));
end
%flops; toc;
%-----------------------------------------------------------------
%this version uses full column vectors (not accounting for zeros)
%flops(0);tic;
%Q=diag(ones(m,1));
%for k = n:-1:1
%   Q = Q - 2*W(1:m,k)*(W(1:m,k)'*Q);
%end
%flops
%toc
%-------------------------------------------------------------------------
%               ------< function [Q,R] = mgs(A) >------
%-------------------------------------------------------------------------
function [Q,R] = mgs(A)
% performs the modified Gram-Schmidt QR factorization of a full rank
% matrix A.
p=size(A);  %determine the dimensions of A
m=p(1);n=p(2);
V=A;        % This version does not destroy the matrix
R=zeros(p(2),p(2)); Q=zeros(p);  %ensure zeros in unaffected places
% if space is a problem, can store in one array.
%tic;flops(0);  % start flop counter and clock
for i = 1:n
   R(i,i) = norm(V(:,i),2);     % normalization of next projector direction
   Q(:,i) = V(:,i)/R(i,i);      % this is the stage of multiplying by the
                                % diagonal matrix, prob. 8.3  (1)
   for j = i+1:n
      R(i,j) = dot(Q(:,i),V(:,j)); %this computes the entries of the upper
                                % triangular matrix
      V(:,j) = V(:,j) - R(i,j)*Q(:,i); %(2)mult. by upper triang. matrix
   end
end
%toc; flops;
```

```
%-------------------------------------------------------------------------
% OUTPUT of script test_house: flop and timing comparisons for arbitrary A
%-------------------------------------------------------------------------
santana (ultra sparc20)
>> test_house
   n           f1           f2           f3         t1         t2         t3
-------------------------------------------------------------------------
  10         2257         1978         2310      0.057      0.025      0.068
  20        15517        14558        17220      0.019      0.009      0.111
  40       115037       111518       132840      0.050      0.026      0.450
  80       886077       872638      1043280      0.157      0.108      1.858
 160      6956157      6903678      8268960      0.813      0.728      7.735
 320     55128317     54920958     65843520      7.070      6.821     34.204
 640    438960637    438136318    525517440     63.220     61.891    171.440
-------------------------------------------------------------------------
hekla (sparc 2)
>> test_house
   n           f1           f2           f3         t1         t2         t3
-------------------------------------------------------------------------
  10         2257         1978         2310      0.075      0.038      0.248
  20        15517        14558        17220      0.202      0.083      1.158
  40       115037       111518       132840      0.436      0.254      4.778
  80       886077       872638      1043280      1.566      1.131     19.801
 160      6956157      6903678      8268960      8.437      7.407     87.646
 320     55128317     54920958     65843520     67.587     55.456    404.754
 640    438960637    438136318    525517440    430.386    401.582   2276.707
-------------------------------------------------------------------------


-------------------------------------------------------------------------
HOMEWORK PROBLEM 10.3,
Comparison of 3 different QR factorization routines:
-------------------------------------------------------
% test script: hw5_10_3
disp('        ops      CPU time        Absolute error (inf-norm)      ')
disp('-----------------------------------------------------------')
format long e
   A=[1 2 3;4 5 6;7 8 7; 4 2 3;4 2 2];
% QR by Householder routines "house" and "formQ"
   flops(0);tic;
   [W,R1]=house(A);
   Q1 = formQ(W);
   f1 = flops; t1 = toc;
   E1 = A-Q1*R1;
   e1 = norm(E1,inf);
disp('Householder factorization')
disp(sprintf('  %10.0f   %10.6f   %22.16f   ',f1,t1,e1))
% QR by modified GS routine "mgs"
   flops(0);tic;
   [Q2,R2] = mgs(A);
   f2 = flops; t2 = toc;
   E2 = A-Q2*R2;
   e2 = norm(E2,inf);
disp('Gram-Schmidt factorization')
disp(sprintf('  %10.0f   %10.6f   %22.16f   ',f2,t2,e2))
% QR by matlab routine "qr"
   flops(0);tic;
   [Q3,R3] = qr(A,0);
   f3 = flops;
```

```
   t3 = toc;
   E3 = A-Q3*R3;
   e3 = norm(E3,inf);
disp('Matlab intrinsic qr-factorization')
disp(sprintf('  %10.0f   %10.6f   %22.16f   ',f3,t3,e3))
disp('-------------------------------------------------------------------')
>> diary
>> hw5_10_3
        ops      CPU time         Absolute error (inf-norm)
-------------------------------------------------------------
Householder factorization
         492     0.005307       0.0000000000000036  ( FULL)
   (     466     0.004351       0.0000000000000036  REDUCED)
Gram-Schmidt factorization
         138     0.002582       0.0000000000000009
Matlab intrinsic qr-factorization
         272     0.000416       0.0000000000000094
-------------------------------------------------------------------
The R matrices, as computed by the three algorithms:
R1 =
    -9.899494936611665e+00    -9.495433918790781e+00    -9.697464427701224e+00
                         0    -3.291919606229404e+00    -3.012943368413352e+00
                         0                         0     1.970115715434853e+00
                         0                         0                         0
                         0                         0                         0
-------------------------------------------------------------
R2 =
     9.899494936611665e+00     9.495433918790779e+00     9.697464427701222e+00
                         0     3.291919606229403e+00     3.012943368413352e+00
                         0                         0     1.970115715434855e+00
-------------------------------------------------------------
R3 =
    -9.899494936611665e+00    -9.495433918790779e+00    -9.697464427701220e+00
                         0    -3.291919606229403e+00    -3.012943368413350e+00
                         0                         0     1.970115715434854e+00
-------------------------------------------------------------
The Q matrices as computed by the three algorithms (notice sign differences)
 (Householder algorithm actually gave full factorization)
Q1(:,1:3) =
    -1.010152544552210e-01    -3.161730695248585e-01     5.419968998794573e-01
    -4.040610178208843e-01    -3.533699012336644e-01     5.161875236947212e-01
    -7.071067811865475e-01    -3.905667329424716e-01    -5.247906490896336e-01
    -4.040610178208843e-01     5.579524756321020e-01     3.871406427710416e-01
    -4.040610178208843e-01     5.579524756321020e-01    -1.204437555287684e-01
-------------------------------------------------------------
Q2 =
     1.010152544552211e-01     3.161730695248581e-01     5.419968998794579e-01
     4.040610178208843e-01     3.533699012336651e-01     5.161875236947217e-01
     7.071067811865476e-01     3.905667329424719e-01    -5.247906490896340e-01
     4.040610178208843e-01    -5.579524756321020e-01     3.871406427710413e-01
     4.040610178208843e-01    -5.579524756321020e-01    -1.204437555287686e-01
-------------------------------------------------------------
Q3 =
    -1.010152544552210e-01    -3.161730695248572e-01     5.419968998794579e-01
    -4.040610178208843e-01    -3.533699012336649e-01     5.161875236947218e-01
    -7.071067811865475e-01    -3.905667329424717e-01    -5.247906490896337e-01
    -4.040610178208843e-01     5.579524756321023e-01     3.871406427710413e-01
    -4.040610178208843e-01     5.579524756321023e-01    -1.204437555287687e-01
-------------------------------------------------------------
```

```
------------------------------------------------------------------------
                        Problem 11.3
------------------------------------------------------------------------
% script: hw5_11_3
m = 50; n = 12;
t = linspace(0,1,m)';
B = vander(t);
BB = fliplr(B);
A  = BB(:,1:n);
b = cos(4*t);
%-----------------------------------------------
% solution by normal equations an \
A1 = A'*A;
b1 = A'*b;
x1 = A1\b1;
%-----------------------------------------------
% solution by QR via mgs
[Q1,R1] = mgs(A);
b2 = Q1'*b;
x2 = R1\b2;
%-----------------------------------------------
% solution by QR via house and formQ
[W3,R3] = house(A);
Q3 = formQ(W3,n);
b3 = Q3'*b;
x3 = R3\b3;
%-----------------------------------------------
% solution via MATLAB qr (householder)
[Q4,R4] = qr(A,0);
b4 = Q4'*b;
x4 = R4\b4
%-----------------------------------------------
% solution via MATLAB \  (qr)
x5 = A\b
%-----------------------------------------------
% solution by MATLAB svd
[U6,S6,V6] = svd(A,0);
b6 = U6'*b;
b61= S6\b6;
x6 = V6*b61;
disp('------------------------------------------------------------------------')
disp(' coeff        Normal Equs.            mgs QR            house/formQ QR  ')
disp('------------------------------------------------------------------------')
 for i = 1:n
disp(sprintf('%2.0f  %22.16f   %22.16f   %22.16f   ',i,x1(i),x2(i),x3(i)))
 end
disp('------------------------------------------------------------------------')
disp(' coeff        MATLAB qr.              MATLAB \              MATLAB SVD    ')
disp('------------------------------------------------------------------------')
 for i = 1:n
disp(sprintf('%2.0f  %22.16f   %22.16f   %22.16f   ',i,x4(i),x5(i),x6(i)))
 end
disp('------------------------------------------------------------------------')
```

OUTPUT of script hw5_11_3

coeff	Normal Equs.	mgs QR	house/formQ QR
1	1.0000000081896643	0.9999999995910052	1.0000000009965966
2	-0.0000025133881074	-0.0000000266021513	-0.0000004227427950
3	-7.9999031216709344	-7.9999958240780131	-7.9999812356890079
4	-0.0014656292723255	-0.0001070567192450	-0.0003187632034550
5	10.6781868410937530	10.6678303775985182	10.6694307957369716
6	-0.0532492396411685	-0.0066749531103873	-0.0138202873756417
7	-5.5354817305943529	-5.6671478069769634	-5.6470756289145303
8	-0.2793833651375133	-0.0388525969664806	-0.0753160216803314
9	1.9344232453730554	1.6508290053331509	1.6936069605831718
10	-0.1710462207551495	0.0373231844275405	0.0060321107435869
11	-0.3004584042583804	-0.3872184166229742	-0.3742417042153702
12	0.0747365059086554	0.0903704952888315	0.0880405761990397

coeff	MATLAB qr.	MATLAB \	MATLAB SVD
1	1.0000000009966143	1.0000000009966148	1.0000000009966135
2	-0.0000004227434609	-0.0000004227436707	-0.0000004227434894
3	-7.9999812356755324	-7.9999812356689253	-7.9999812356746949
4	-0.0003187633473716	-0.0003187634318361	-0.0003187633588128
5	10.6694307966383040	10.6694307972147264	10.6694307967217892
6	-0.0138202909116742	-0.0138202932809089	-0.0138202912726537
7	-5.6470756198966434	-5.6470756136733851	-5.6470756189150544
8	-0.0753160368502198	-0.0753160475498423	-0.0753160385714864
9	1.6936069772795401	1.6936069892709593	1.6936069792197830
10	0.0060320991795924	0.0060320907393184	0.0060320978241377
11	-0.3742416996429023	-0.3742416962555675	-0.3742416991098951
12	0.0880405754119923	0.0880405748207566	0.0880405753220038

etc.

SVD does the best,
Matlab qr,
house/formQ close
(as is also Matlab \)
mgs does worse (as expected)
Normal equations perform
the worst
(ill-conditioning)