University of Helsinki / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 08 / Solutions

1. (a) Show that for a square matrix a, if λ is an eigenvalue of a (i.e. for some $x \neq 0, ax = \lambda x$), then $\lambda + c$ is an eigenvalue of $a + c * I$ where I is the identity matrix of the same size as a .

(b) Let a_1 be a random $n \times n$ matrix, $a = a_1 + a_1^T$, and define t to be the least eigenvalue of a if it is < 0 and $= 0$ otherwise. Show that $a - tI$ has nonnegative eigenvalues and that it is symmetric.

(c) Generate with (b) symmetric square matrices a with nonnegative eigenvalues and show that if $[u, s, v] = svd(a)$ and $ss = sqrt(s), c = u * ss$, then $a = c * c^T$.

Solution:

```
% FILE d081.m begins.
fprintf('  min. eig(a)       norm(a- c*(c^T))\n')
for m=3:13
    a1=2*(rand(m,m)-0.5);
    a=a1+(a1');
    t=min(eig(a))-0.01;
    if (t>0)
       t=0;
    end
    a=a-t*eye(m,m);
    fprintf('%12.4e  ',min(eig(a)))
    [u,s,v]=svd(a);
    ss=sqrt(s);
    c=u*ss;
    cT=transpose(c);
    fprintf('  %12.4e\n',norm(a-c*cT))
end
% FILE d081.m ends.
```

Output:

```
  min. eig(a)       norm(a- c*(c^T))
  1.0000e-02      2.8277e-15
  1.0000e-02      1.7759e-15
  1.0000e-02      2.7153e-15
  1.0000e-02      7.3617e-15
  1.0000e-02      7.4746e-15
  1.0000e-02      6.5471e-15
  1.0000e-02      8.1564e-15
  1.0000e-02      1.4054e-14
  1.0000e-02      1.8071e-14
  1.0000e-02      1.4443e-14
  1.0000e-02      3.8112e-14
```

2. The vertices of a quadrilateral are $(0, 0), (p_1, p_2), (1, 0), (q_1, q_2)$, where $p_2 < 0, q_2 > 0$. Generate such quadrilaterals and compute their areas. Show that Bretschneider's formula for the area holds:

$$K = \sqrt{(s-a)(s-b)(s-c)(s-d) - abcd\cos^2\alpha}$$

where α is the mean value of the angles at p and q and a, b, c, d are the sides, $2s = a + b + c + d$. Hint: Use polyarea to check the formula.

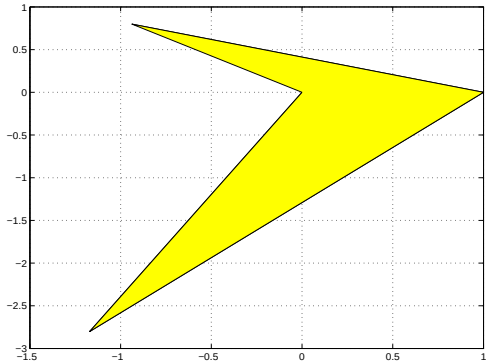
Solution:

```
% FILE d082.m begins.
for ii=1:10
% Generate p, q
%  0  p  1  q
p = 3*[rand-0.5  -rand];
q = 3*[rand-0.5  rand];
a=norm(p); b=norm(p-[1,0]);
c=norm(q); d=norm(q-[1,0]);
s=0.5*(a+b+c+d);
al1=acos(dot(-p, [1,0]-p)/(a*b));
al2=acos(dot(-q, [1,0]-q)/(c*d));
alpha=0.5*(al1+al2);
area1=polyarea([0 p(1) 1 q(1)], [0 p(2) 0 q(2)]);
myc=cos(alpha)^2;
K= sqrt((s-a)*(s-b)*(s-c)*(s-d)-a*b*c*d*myc);
figure(1)
fprintf('%12.4e \n', K-area1)
plot([0 p(1) 1 q(1)], [0 p(2) 0 q(2)]);
patch([0 p(1) 1 q(1)], [0 p(2) 0 q(2)],'y');
grid on
pause(0.1)
end
% FILE d082.m ends.
```

Output:

```
  5.5511e-16
```

```
 0.0000e+00
-1.3323e-15
-3.3307e-15
 1.6653e-16
 5.5511e-17
 1.7764e-15
-2.4425e-15
 3.1086e-15
-2.2204e-16
```



3. Let B and C be $m \times n$ matrices with real entries and let $A = B + iC$. Relate the singular values of A to those of

$$\begin{bmatrix} B & -C \\ C & B \end{bmatrix}.$$

Observe this gives an idea to reduce the computation of the SVD of complex $m \times n$ matrices to the case of real $2m \times 2n$ matrices.

Solution:

```
% FILE d083.m begins.
% The singular values of B+ i*C are twice repeated
% among the singular values of [ B -C; C B].
m= 4; n=2;
for pp=1:10
b= rand(m,n)-0.5;
c= rand(m,n)-0.5;
a=b+i*c;
[u,s,v]=svd(a);
```



```
a2=[b -c; c b];
[u2,s2,v2]=svd(a2);
for j=1:min(m,n)
fprintf('s(%d)-s2(2*%d-1)= %12.3e\n',j,j,s(j)-s2(2*j-1))
end
end
disp('s =')
disp(s)
disp('s2 =')
disp(s2)
% FILE d083.m ends.
```

Output: If we inspect the output (see below) of the program, then we see that each singular values of A occurs twice among the singular values of the new matrix.

```
s(1)-s2(2*1-1)=   -3.331e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=    2.220e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=   -2.220e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=   -1.110e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=   -1.110e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=    0.000e+00
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=   -3.331e-16
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=    0.000e+00
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=    0.000e+00
s(2)-s2(2*2-1)=    0.000e+00
s(1)-s2(2*1-1)=   -2.220e-16
s(2)-s2(2*2-1)=    0.000e+00
s =
    1.0353         0
         0    0.5554
         0         0
```

```
         0         0

s2 =
    1.0353         0         0         0
         0    1.0353         0         0
         0         0    0.5554         0
         0         0         0    0.5554
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
```

4. Recall from linear algebra that if $\Delta \equiv ad - bc \neq 0$, then

$$\Delta \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

Consider the Newton method $x_{n+1} = x_n - J_f(x_n)^{-1} f(x_n)$ to solve the system

$$\begin{cases} x_1^2 + x_2^2 - 1 &= 0 \\ x_1^2 - x_2^2 &= 0 \end{cases}$$

with the initial point $(0.5, 0.5)$. Use standard methods from linear algebra and multidimensional calculus, such as the above formula for matrix inverse, to compute the following expressions analytically and to print the numerical values $J_f(x_n), J_f(x_n)^{-1}, T(x_n) \equiv J_f(x_n)^{-1} f(x_n)$ for the first three iteration steps.

Solution: We have $f(x) = (f_1(x), f_2(x))^T$ with $f_1(x) = x_1^2 + x_2^2 - 1$, $f_2(x) = x_1^2 - x_2^2$. The definition of the Jacobian matrix gives

$$J_f(x) = \begin{bmatrix} \partial_1 f_1(x) & \partial_2 f_1(x) \\ \partial_1 f_2(x) & \partial_2 f_2(x) \end{bmatrix} = \begin{bmatrix} 2x_1 & 2x_2 \\ 2x_1 & -2x_2 \end{bmatrix}$$

and also

$$J_f(x)^{-1} = -\frac{1}{4x_1x_2} \begin{bmatrix} -2x_2 & -2x_2 \\ -2x_1 & 2x_1 \end{bmatrix}; \quad T(x) = \begin{bmatrix} (-1 + 2x_1^2)/(4x_1) \\ (-1 + 2x_2^2)/(4x_2) \end{bmatrix}$$

```
function z=d084_tst(x, nriter)
  for j=1:nriter
     disp('J_f(x) =')
```



```
     a=J(x); b=invJ(x); c= b*F(x);
      disp(a)
     disp('J_f(x)^{-1} =')
     disp(b)
     disp('T(x) = J_f(x)^{-1} *F(x) =')
     disp(c)
     disp('J(x)*invJ(x) =')
     disp(a*b)
     disp('J_f(x)^{-1} *F(x)-T(x) =')
     disp(c-T(x))
     x= x-c;
     disp('x =')
     disp(x)
     pause(1)
  end

function w= F(x)
  w=[x(1)*x(1)+ x(2)*x(2)-1; x(1)*x(1)- x(2)*x(2)];

function w= J(x)
  w=2*[x(1)  x(2); x(1) -x(2)];

function w=invJ(x)
  w=-(1/(4*x(1)*x(2)))*[-x(2) -x(2); -x(1) x(1)];

function w=T(x)   % T= invJ(x)*F(x)
  w=[ (-1+2*x(1)*x(1))/(4*x(1)) ; (-1+2*x(2)*x(2))/(4*x(2))];
```

Output: With the commands:

```
format rational
d085([1/2; 1/2], 3)
```

we get:

```
J_f(x) =
       1              1
       1             -1

J_f(x)^{-1} =
       1/2            1/2
       1/2           -1/2
```

```
T(x) = J_f(x)^{-1} *F(x) =
      -1/4
      -1/4

J(x)*invJ(x) =
       1              0
       0              1

J_f(x)^{-1} *F(x)-T(x) =
       0
       0

x =
       3/4
       3/4

J_f(x) =
       3/2            3/2
       3/2           -3/2

J_f(x)^{-1} =
       1/3            1/3
       1/3           -1/3

T(x) = J_f(x)^{-1} *F(x) =
       1/24
       1/24

J(x)*invJ(x) =
       1              0
       0              1

J_f(x)^{-1} *F(x)-T(x) =
       0
       0

x =
      17/24
      17/24

J_f(x) =
      17/12          17/12
      17/12         -17/12
```



```
J_f(x)^{-1} =
       6/17           6/17
       6/17          -6/17

T(x) = J_f(x)^{-1} *F(x) =
       1/816
       1/816

J(x)*invJ(x) =
       1              0
       0              1

J_f(x)^{-1} *F(x)-T(x) =
       1/4611686018427387904
       1/4611686018427387904

x =
     577/816
     577/816
```

Observe that the nearest root is $(1/\sqrt{2}, 1/\sqrt{2})$.

5. For $t \in R \setminus \{0\}$, the pseudo-inverse is defined by $t^+ = 1/t$ and we set $0^+ = 0$. If s is a $m \times n$ diagonal matrix let s^+ be the $n \times m$ diagonal matrix obtained by applying this operation elementwise: $(s^+)_{i,j} = s^+_{i,j}$. For an $m \times n$ matrix a with a SVD $a = usv^T$ we define $a^+ = vs^+u^T$. Check for a number of random matrices whether and how well a^+ agrees with the MATLAB pseudo-inverse $pinv(a)$.

Solution:

```
% FILE d085.m begins.
m=5; n=3;
for j=1:10
  a=rand(m,n)-0.5;
  [u,s,v]=svd(a);  % a= u * s* v^T
  sp=0*transpose(s);
  for jj=1: min(size(sp))
    if (abs(s(jj,jj))>1e-20)
      sp(jj,jj)=1/s(jj,jj);
    end
  end
```

```
  uT=transpose(u);
  pia= v*(sp*transpose(u));
  tst=norm(pia-pinv(a));
  fprintf('%12.4e\n', tst);
end
% FILE d085.m ends.
```

Output:

```
  7.1808e-16
  5.1742e-16
  7.1480e-16
  2.9451e-16
  1.0380e-15
  3.4447e-16
  1.7964e-16
  5.3842e-16
  3.6239e-16
  2.2895e-16
```

6. Finding the root of a linear $n \times n$ system $ax = b$ can be considered as a minimization problem for the function $g(x) = |a*x-b| = norm(a*x-b)$. Use this idea to solve the problem $a*x = b$ when a is $n \times n$ Hilbert matrix. Report for each $n = 2 : 9$ the residual $|ax - b|$ and $|xnum - exact|$ when exact= ones(n.1) and b=a*exact.

Solution:

```
% FILE  d087.m  begins.
% Cf. parf04.m
clear
fprintf(' n     Residual      Error \n')
for n=2:9
a=hilb(n);
exact=ones(n,1);
b=a*exact;
fobj=inline('norm(a*x-b)','x','a','b');
guess=10*rand(n,1);      % Initial guess
x = fminsearch(fobj,guess,[],a,b);
residual=fobj(x,a,b);
fprintf('%3d  %12.4e  %12.4e\n', n,residual, norm(x-exact));
end
% FILE  d087.m  ends.
```


Output:

```
 n     Residual      Error
  2    1.7688e-06    2.6597e-05
  3    1.7981e-03    6.6910e-01

Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 0.000000

  4    2.4744e-08    2.0572e-05
  5    2.4053e-06    7.3146e-01
  6    1.3183e-04    1.8952e+01
  7    1.5465e-04    8.1323e+00
  8    2.1421e-04    1.2178e+01
  9    6.4640e-04    3.0063e+01
```