University of Turku / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 04 / Solutions

1. (a) Let X and Y be independent uniformly distributed random variables on $(0,1)$. As we know, samples of X can be generated by `x=rand(1,100);` for instance. Now it is a basic fact (this need not be proven) that the new random variables

$$U = \cos(2\pi X)\sqrt{-2\log Y}; \quad V = \sin(2\pi X)\sqrt{-2\log Y}$$

follow the normal distribution with parameters $(0,1)$, i.e. with mean 0 and variance 1. Use this so called Box-Müller method to generate 200 samples of normal distribution, plot the result with the command `hist`, compute the mean and standard deviation of the sample.

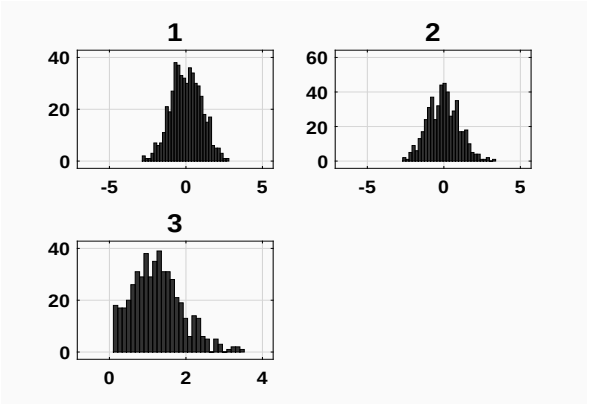
(b) The amplitude distribution of a signal sent by a mobile phone to a base station follows so called Rayleigh distribution. Suppose that X_1, X_2 are zero-mean normally distributed random variables with variance σ^2 and define a new random variable R by $R = \sqrt{X_1^2 + X_2^2}$. Then R follows the Rayleigh distribution. Generate 100 samples of a Rayleigh distribution and plot the histogram.

Solution:

```
% FILE d041.m begins.
close all
m=500;
x=rand(1, m);    y=rand(1, m);
u=cos(2*pi*x).*sqrt(-2*log(y));
v=sin(2*pi*x).*sqrt(-2*log(y));
figure(1)
h=axes;
set(h,'FontSize',20,'FontWeight','bold')

subplot(2,2,1)
hist(u,30)
title('1','FontSize',20,'FontWeight','bold')
fprintf('mean(u)= %8.4f , std(u)= %8.4f \n',mean(u), std(u))
grid on

widemarg(gcf)

subplot(2,2,2)
hist(v,30)
title('2', 'FontSize',20,'FontWeight','bold')
fprintf('mean(v)= %8.4f , std(v)= %8.4f \n',mean(v), std(v))
grid on
widemarg(gcf)
%%%
myvar=rand;
variance=myvar;
variance2=myvar;
X1=normrnd(zeros(1,1000),variance);
X2=normrnd(zeros(1,1000),variance2);
R=sqrt(X1.^2+X2.^2);

%%%
r=sqrt(u.^2 + v.^2);
subplot(2,2,3)
hist(r,30);
title('3', 'FontSize',20,'FontWeight','bold')
fprintf('mean(r)= %8.4f  \n',mean(r) )
widemarg(gcf)
grid on
print -dps d041.ps

% FILE d041.m ends.
```

Output:

```
mean(u)=   0.0168 , std(u)=   0.9899
mean(v)=  -0.0263 , std(v)=   1.0089
mean(r)=   1.2543
```



2. Suppose that $f\colon [a,b] \to [0,\infty)$ is continuous and that $0 \le f(x) \le M$ for all $x \in [a,b]$. Use the Monte Carlo method to approximate the value of

$$\int_a^b f(x)\,dx,$$

that is, choose m random points in $[a,b] \times [0,M]$ and compute the ratio p/m where p is the number of points below the graph of $f(x)$. Apply this method for the function

$$f(x) = \sum_{j=1}^{n} c_j(1 + \sin(d_j x))$$

in $[0,1]$ with $m = 10j$, $j = 10 : 10 : 100$ where $n = 4$, c=rand(1,n), d= 1+3*rand(1,n). Compare your result to the exact value

$$\int_a^b f(x)dx = (b-a)\sum_{j=1}^{n} c_j + \sum_{j=1}^{n}(c_j/d_j)(\cos(d_j * a) - \cos(d_j * b))\,,$$

see Problem 3/Exercise 2.

Solution:

```
function z=d042f(c,d,x);
tmp=0*x;
m=length(c);
for j=1:m
  tmp=tmp+c(j)*(1+sin(d(j)*x));
end
z=tmp;
% FILE d042f.m end
```

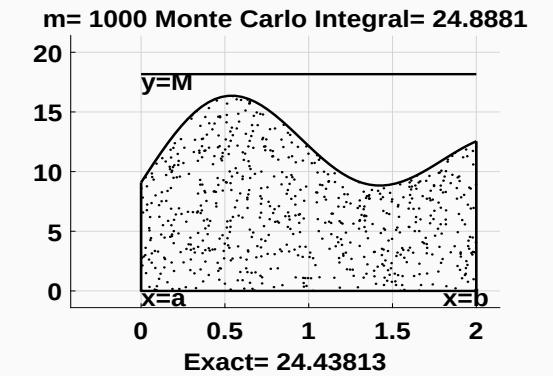
```
% FILE d042.m begin
% USES: d042f.m
path(path,'../util')
clf
n=5;
c=3*rand(1,n); d=1+3*rand(1,n); % All d(j)'s must be different
                                % from 0
a=0; b=2;
exact=(b-a)*sum(c)+...
 sum((c./d).*cos(a*d)) -sum((c./d).*cos(b*d));
```



```
M=2*sum(c);  % Note:  0<=d042f(c,d,x) <= M

% Compute function values for plotting
xx=a:0.02*(b-a):b;
yy=d042f(c,d,xx);
lexx=length(xx);


fprintf('Exact integral = %12.5f\n',exact)

fprintf('  m    Monte Carlo    Exact   Error\n' );

for mm=100:100:1000
  figure(1)
  clf;
  axes('FontSize',[20],'FontWeight','bold'); hold on;
  plot(xx,yy, xx, M*ones(1,lexx),'LineWidth',2)
  plot([a a b b],[yy(1) 0 0 yy(end)],'LineWidth',2)
  grid on

  x=a+(b-a)*rand(1,mm);
  y2=M*rand(1,mm);
  y1=d042f(c,d,x);
  su=sum(y2<=y1);
  plot(x(y2<=y1), y2(y2<=y1),'r.')

  area=M*(b-a)*su/mm;
  fprintf('%4d  %8.4f        %8.4f   %12.4e\n',...
      mm, area, exact, exact-area)

 title(['m= ' num2str(mm) ' Monte Carlo Integral= ' num2str(area)],...
'FontWeight','bold','FontSize',20);
 text(a, -0.5,'x=a', 'FontWeight','bold','FontSize',20);
 text(b-0.2, -0.5,'x=b', 'FontWeight','bold','FontSize',20);
 text(a, M-0.5,'y=M', 'FontWeight','bold','FontSize',20);
 xlabel(['Exact= ' num2str(exact,'%12.5f')],...
'FontWeight','bold','FontSize',20);
 widemarg(gcf)
end;
%delete d042.ps
print -dps d042.ps
% FILE d042.m end
```

Output:

```
Exact integral =     24.43813
  m    Monte Carlo    Exact   Error
 100    22.1631         24.4381     2.2750e+00
 200    23.9798         24.4381     4.5835e-01
 300    24.1009         24.4381     3.3724e-01
 400    23.6165         24.4381     8.2168e-01
 500    24.7791         24.4381    -3.4098e-01
 600    24.2220         24.4381     2.1613e-01
 700    25.2774         24.4381    -8.3926e-01
 800    23.3440         24.4381     1.0942e+00
 900    25.1101         24.4381    -6.7201e-01
1000    24.8881         24.4381    -4.4997e-01
```



3. The ASCII codes of capital letters A,...,Z are 65,...,90. A simple ciphering method, so called Caesar cipher, is the following. Fix an integer $p \in [1, 25]$. Each letter is replaced by another, obtained by increasing its ASCII code by the constant p. (Note that we recycle: 91 corresponds to 65 i.e. after Z come A,B,C,...). The program hlp043.m shows how this happens. Use this idea to decipher the following messages:

N L O G J G Y Y N M J O N C H

and

E N F V Y N T B R F P U V A N

Solution:

```
% FILE  d043.m  begins.
fprintf('ASCII codes of some symbols: \n')
fprintf(' Code   Symbol  Shifted \n');
alfab=[];
for j=65:90
  alfab=[alfab char(j)];
end;
s=alfab;
p=length(s);
for j=1:p
  tmp=65+mod(double(s(j)+1)-65,26);  % 26=90-64
  fprintf('%3d   %3s %10s \n',...
    double(s(j)),s(j),char(tmp));
end
fprintf('\n');
%s='QCADIHCSFUCGIA'; % Computo ergo sum!
s='VEWMPEKSIWGLMRE'
%s='NLOGJGYYNMJONCH';
p=length(s);
for shift=1:26
  for j=1:p
    tmp=65+mod(double(s(j)+shift)-65,26); % 26=90-64
    fprintf('% 3s',char(tmp))
  end
  fprintf('\n');
end
% FILE  d043.m  ends.
fprintf('\n\nAlternative solution:\n')
%clear
% create the cipher vector, where A==0, B==1 and so on
cipher=s-65;
% Loop through all possible shifts
for p=1:26
    % Display the decrypted text
    fprintf('%d:\t%s\n', p, char(mod(cipher + p, 26) + 65));
end
% FILE  d043.m  ends.
```

Output:

```
ASCII codes of some symbols:
 Code   Symbol  Shifted
```

65	A	B
66	B	C
67	C	D
68	D	E
69	E	F
70	F	G
71	G	H
72	H	I
73	I	J
74	J	K
75	K	L
76	L	M
77	M	N
78	N	O
79	O	P
80	P	Q
81	Q	R
82	R	S
83	S	T
84	T	U
85	U	V
86	V	W
87	W	X
88	X	Y
89	Y	Z
90	Z	A

```
s =

    'VEWMPEKSIWGLMRE'
```

W	F	X	N	Q	F	L	T	J	X	H	M	N	S	F
X	G	Y	O	R	G	M	U	K	Y	I	N	O	T	G
Y	H	Z	P	S	H	N	V	L	Z	J	O	P	U	H
Z	I	A	Q	T	I	O	W	M	A	K	P	Q	V	I
A	J	B	R	U	J	P	X	N	B	L	Q	R	W	J
B	K	C	S	V	K	Q	Y	O	C	M	R	S	X	K
C	L	D	T	W	L	R	Z	P	D	N	S	T	Y	L
D	M	E	U	X	M	S	A	Q	E	O	T	U	Z	M
E	N	F	V	Y	N	T	B	R	F	P	U	V	A	N
F	O	G	W	Z	O	U	C	S	G	Q	V	W	B	O
G	P	H	X	A	P	V	D	T	H	R	W	X	C	P

H	Q	I	Y	B	Q	W	E	U	I	S	X	Y	D	Q
I	R	J	Z	C	R	X	F	V	J	T	Y	Z	E	R
J	S	K	A	D	S	Y	G	W	K	U	Z	A	F	S
K	T	L	B	E	T	Z	H	X	L	V	A	B	G	T
L	U	M	C	F	U	A	I	Y	M	W	B	C	H	U
M	V	N	D	G	V	B	J	Z	N	X	C	D	I	V
N	W	O	E	H	W	C	K	A	O	Y	D	E	J	W
O	X	P	F	I	X	D	L	B	P	Z	E	F	K	X
P	Y	Q	G	J	Y	E	M	C	Q	A	F	G	L	Y
Q	Z	R	H	K	Z	F	N	D	R	B	G	H	M	Z
R	A	S	I	L	A	G	O	E	S	C	H	I	N	A
S	B	T	J	M	B	H	P	F	T	D	I	J	O	B
T	C	U	K	N	C	I	Q	G	U	E	J	K	P	C
U	D	V	L	O	D	J	R	H	V	F	K	L	Q	D
V	E	W	M	P	E	K	S	I	W	G	L	M	R	E

```
Alternative solution:
1: WFXNQFLTJXHMNSF
2: XGYORGMUKYINOTG
3: YHZPSHNVLZJOPUH
4: ZIAQTIOWMAKPQVI
5: AJBRUJPXNBLQRWJ
6: BKCSVKQYOCMRSXK
7: CLDTWLRZPDNSTYL
8: DMEUXMSAQEOTUZM
9: ENFVYNTBRFPUVAN
10: FOGWZOUCSGQVWBO
11: GPHXAPVDTHRWXCP
12: HQIYBQWEUISXYDQ
13: IRJZCRXFVJTYZER
14: JSKADSYGWKUZAFS
15: KTLBETZHXLVABGT
16: LUMCFUAIYMWBCHU
17: MVNDGVBJZNXCDIV
18: NWOEHWCKAOYDEJW
19: OXPFIXDLBPZEFKX
20: PYQGJYEMCQAFGLY
21: QZRHKZFNDRBGHMZ
22: RASILAGOESCHINA
23: SBTJMBHPFTDIJOB
24: TCUKNCIQGUEJKPC
25: UDVLODJRHVFKLQD
```

```
26: VEWMPEKSIWGLMRE
```

4. We want to fit a model of the form $f(x) = ae^{bx}$ to the data set

```
 x   1    3    4    6    9   15

 y  4.0  3.5  2.9   2.5 2.75  2.0
```

where a and b are parameters to be determined from the data.

(a) For this purpose we introduce new transformed variables $X = x$, $Y = \log(y)$. Carry out this data transformation and print out the transformed variables.

(b) After the transformation the new model is $F(x) = \log f(x) = bx + \log a$. Apply the usual LSQ method to find b and $\log a$.

(c) Print the results in the following format

```
 x(i)     y(i)     Y(i)    a*exp(b*x(i))     y(i)-a*exp(b*x(i))

 1        4.0      ....

.......

15        2.0
```

and plot the data and the fitted curve in the same figure.

Solution:

```
% FILE d044tst.m begins.
close all; clear
figure(1)
x=  [ 1    3    4    6    9   15];
y =[ 4.0  3.5  2.9   2.5 2.75  2.0];

h=axes;
set(h,'FontSize',20)
newx=x;
newy=log(y);
disp('Data:')
```



```
disp([x; y])

disp('Transformed data:')
disp([newx; newy])
% Logarithm yields linearization: bx + log a
coef=polyfit(newx,newy,1);
b=coef(1);
a=exp(coef(2));
fprintf('Fitted/LSQ model y= a*exp(b*x) with a= %8.4f , b= %8.4f \n',a,b);

m=length(x);
% Something not asked in problem:
% We use also parfit based idea: fminsearch
mymodel=@(xval,lam)(lam(1)*exp(lam(2)*xval));
% fobj2 uses given data x, y
fobj2=@(lam) norm(mymodel(x,lam)-y);

fprintf('Object function value/LSQ= %8.4f\n',fobj2([a, b]));
% myparf uses given data x, y
lam1=myparf(x,y,mymodel, [3, -0.7]);
fprintf('Fitted/myparf  y= a*exp(b*x) with a= %8.4f , b= %8.4f \n',...
    lam1(1),lam1(2));
fprintf('Object function value/myparf= %8.4f\n',...
    fobj2([lam1(1), lam1(2)]));
fprintf(' x(i)    y(i)     Y(i)   a*exp(b*x(i))  y(i)-a*exp(b*x(i))  parfit \n')
for j=1:m
   fprintf(' %3d  %8.3f  %8.3f  %8.3f       %12.3e   %12.3e\n',...
   x(j),y(j), newy(j), a*exp(b*x(j)),  y(j)- a*exp(x(j)*b), ...
       y(j)-mymodel(x(j),lam1) );
end

plot(x,y,'k+','MarkerSize',25)

grid on
hold on
xx= 0:0.05:20;
yy=a*exp(b*xx); yy2=mymodel(xx,lam1);
plot(xx,yy,'b-',  xx,yy2,'r-.', 'LineWidth',2)
txt=['y = ' num2str(a) '* exp( ' num2str(b) '*x)'];
%text(x(3),y(2) ,txt,'FontSize',20,'FontWeight','bold')
title(txt,'FontSize',20,'FontWeight','bold')
xlabel('Solid/ dash dotted = LSQ /myparf', 'FontSize',20,'FontWeight','bold')
widemarg(gcf)
```

```
%delete d044tst1.ps
print -dps d044tst1.ps
% Something not asked at all:
% We plot object functions close to found parameter value
la1= (a-0.3):0.03:(a+0.3); la2=  (b-0.3):0.03:(b+0.3);
[xxx, yyy]=meshgrid(la1, la2);
[d1,d2]=size(xxx); zzz=0*xxx;
for ii=1:d1
    for jj=1:d2
        lam=[xxx(ii,jj) , yyy(ii,jj)];
        zzz(ii,jj)=fobj2(lam);
    end
end
% We first find row and column indices of the
% least entry in zzz
[ii,jj]=find(zzz==min(min(zzz)));
% Using these found indices we get candidate for [a, b]
mya=xxx(ii,jj); myb=yyy(ii,jj);
[mya  myb]

figure
% Plot picture of the fobj2 function
mesh(xxx,yyy,zzz)
hold on
plot3(a,b,fobj2([a, b]),'k*', mya,myb,fobj2([mya, myb]),'k.',...
 lam1(1),lam1(2),fobj2([lam1(1), lam1(2)]),'k+', 'MarkerSize',10)
title(' */ . / +  =  LSQ /tabulation/myparf')
xlabel('a'); ylabel('b');
%delete d044tst2.ps
print -dps d044tst2.ps

echo on
% By choice expect tst1 = 0
tst1=fobj2([mya myb])-zzz(ii,jj)
% Because xxx, yyy were created with crude step 0.05
% expect tst2 <0
tst2=fobj2([a b])-zzz(ii,jj)
% expect tst3 <0
tst3=fobj2([lam1(1) lam1(2)])-zzz(ii,jj)
echo off
function w=myparf(xdata,ydata,mymodel, laminit)
% E.g. fmodel=@(x,lam)exp(-lam(1)* x)+lam(2)*exp(-lam(3)*x);
fobj=@(lam) norm(mymodel(xdata,lam)-ydata);
```

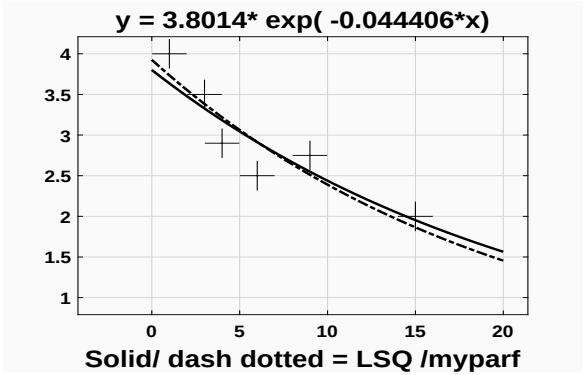


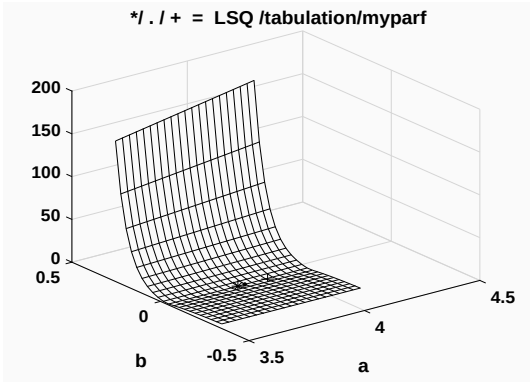
```
[lam fval eflag] = fminsearch(fobj,laminit);
w=[lam ];
end
% FILE d044tst.m ends.
```

Output:

```
Data:
    1.0000    3.0000    4.0000    6.0000    9.0000   15.0000
    4.0000    3.5000    2.9000    2.5000    2.7500    2.0000

Transformed data:
    1.0000    3.0000    4.0000    6.0000    9.0000   15.0000
    1.3863    1.2528    1.0647    0.9163    1.0116    0.6931

Fitted/LSQ model y= a*exp(b*x) with a=   3.8014 , b=  -0.0444
Object function value/LSQ=   0.6743
Fitted/myparf  y= a*exp(b*x) with a=   3.9260 , b=  -0.0496
Object function value/myparf=   0.6577
 x(i)    y(i)     Y(i)   a*exp(b*x(i))  y(i)-a*exp(b*x(i))  parfit
   1     4.000     1.386     3.636          3.637e-01      2.642e-01
   3     3.500     1.253     3.327          1.727e-01      1.172e-01
   4     2.900     1.065     3.183         -2.828e-01     -3.189e-01
   6     2.500     0.916     2.912         -4.123e-01     -4.147e-01
   9     2.750     1.012     2.549          2.010e-01      2.386e-01
  15     2.000     0.693     1.953          4.716e-02      1.354e-01

ans =

    3.8314   -0.0444

% By choice expect tst1 = 0
tst1=fobj2([mya myb])-zzz(ii,jj)

tst1 =

     0

% Because xxx, yyy were created with crude step 0.05
% expect tst2 <0
tst2=fobj2([a b])-zzz(ii,jj)

tst2 =
```

```
    0.0022

% expect tst3 <0
tst3=fobj2([lam1(1) lam1(2)])-zzz(ii,jj)

tst3 =

   -0.0144

echo off
```





5. For a complex $n \times n$ matrix a let $P_i = \sum_{j=1, j\neq i}^{n} |a_{i,j}|$, $m_0 = \min\{|a_{i,i}| - P_i : i = 1, ..., n\}$, $m = \max\{m_0, 0\}$, $M = \max\{|a_{i,i}| + P_i : i = 1, ..., n\}$.

By Gerschgorin's theorem (recall Exercise 03) the eigenvalues λ_i of a satisfy

$$m \leq |\lambda_i| \leq M; \quad i = 1, ..., n$$

and it also follows that $m^n \leq D \leq M^n$, $D = |\det(a)|$. Set $m_1 = \min\{|\lambda_i| : i = 1, ..., n\}$ and $m_2 = \max\{|\lambda_i| : i = 1, ..., n\}$. Write a MATLAB script that experimentally confirms these statements, by printing out the test results in the following format

```
 n    m    m1    m2  M     D - m^n      M^n -D
```

Use random complex $n \times n$ matrices, n=5:5:50.

Repeat the experiment for the matrices a=2*n*eye(n)+rand(n,n)+i*rand(n,n).

Solution:

```
% FILE d045.m begins.
fprintf('m1= min(|eig(a)|)   m2= max(|eig(a)|)   D= |det(a)|\n')
fprintf('  n       m         m1        m2       M       D-m^n    M^n -D\n')
for n=5:5:50
  a=rand(n,n)+ i*rand(n,n);
  a=a+n*eye(n);
  m0=sum(sum(abs(a))); ma=-m0;
  for ii=1:n
    tmp= sum(abs(a(ii,:)));
    tmp1=tmp-abs(a(ii,ii));
    tmp2=tmp+abs(a(ii,ii));
    m0=min(tmp1,m0);
    ma=max(ma,tmp2);
  end
  m=max(m0,0);
  D=abs(det(a));
  myeig=eig(a);
  fprintf('%3d  %8.4f  %8.4f  %8.4f  %8.4f %12.4e  %12.4e\n',...
   n, m,min(abs(myeig)), max(abs(myeig)),ma,...
  D-m^n ,ma^n- D);
end
% FILE d045.m ends.
```

Output:

```
m1= min(|eig(a)|)   m2= max(|eig(a)|)   D= |det(a)|
  n       m         m1        m2       M       D-m^n    M^n -D
  5    1.8870    4.2234    7.5202   14.4286   4.2370e+03    6.2108e+05
 10    4.8936    9.3189   15.8776   28.9615   1.7354e+10    4.1514e+14
 15    8.9196   13.6152   23.3202   44.5831   7.3524e+17    5.4645e+24
 20   11.6199   18.4776   32.1313   58.7435   1.8579e+26    2.3944e+35
 25   15.1139   23.1460   38.8989   72.5220   1.2965e+35    3.2490e+46
```

30	19.6314	28.1199	47.6079	87.3491	3.5412e+44	1.7288e+58
35	22.0173	32.7556	55.3620	99.8437	1.7361e+54	9.4673e+69
40	27.0931	37.8644	63.4628	116.0663	1.9767e+64	3.8748e+82
45	30.5405	42.2604	71.7989	130.4655	3.7633e+74	1.5751e+95
50	30.7883	47.5025	79.0314	146.1620	1.4551e+85	1.7447e+108

6. The arithmetic-geometric mean $\mathrm{ag}(a,b)$ of two positive numbers $a > b > 0$ is defined as $\mathrm{ag}(a,b) = \lim a_n$, where $a_0 = a, b_0 = b$, and

$$a_{n+1} = (a_n + b_n)/2, \quad b_{n+1} = \sqrt{a_n b_n}, \quad n = 0, 1, 2, \ldots$$

(a) Write a function, which takes two arguments (`double`), computes ag and returns the value (`double`).

(b) The hypergeometric function ${}_2F_1(a,b;c;x)$ is defined as a sum of the series,

$${}_2F_1(a,b;c;x) = 1 + \frac{ab}{c}\frac{x}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)}\frac{x^2}{2!} + \ldots$$
$$+ \frac{a(a+1)\ldots(a+j-1)b(b+1)\ldots(b+j-1)}{c(c+1)\ldots(c+j-1)}\frac{x^j}{j!} + \ldots \quad .$$

This hypergeometric series converges for $\mathrm{abs}\, x < 1$. Gauss proved in 1799 that there is a connection between the hypergeometric function and the arithmetic-geometric mean,

$${}_2F_1(\tfrac{1}{2},\tfrac{1}{2};1;r^2) = \frac{1}{\mathrm{ag}(1,\sqrt{1-r^2})}$$

for $0 < r < 1$. Tabulate the difference of the two sides of this identity for $r = 0.05k, k = 1,\ldots,19$. Use the routine on the web-page to calculate the values of the ${}_2F_1$ or the MATLAB built-in program (`help hypergeom`).

Solution:

```
% FILE  d046.m
fprintf('2F1        agmean     error\n')
for k=1:19,
  r=0.05*k;
  s=hypergeom(0.5,0.5,1,r*r);
  t=1/ag(1,sqrt(1-r*r));
  fprintf('%f %f %10.6e\n',s,t,s-t)
end
% FILE  d046.m  ends.
```



```
% FILE:  ag.m  begins.
% The Gauss arithmetic-geometric mean is computed
function y=ag(y,x)
[m,n]=size(x);
if ( [m,n] ~= size(y))
   disp('Argument error in ag.m');
   return;
end
a=y;
b=x; ind =0;
while (ind < 10)
    c=a; a=.5*(a+b); b=sqrt(c.*b);
    ind = ind+1;
end
y=a;
end
% FILE:  ag.m  ends.    -------------
```

Output:

```
2F1         agmean     error
1.000626 1.000626 2.220446e-16
1.002514 1.002514 -2.220446e-16
1.005697 1.005697 -2.220446e-16
1.010231 1.010231 0.000000e+00
1.016199 1.016199 -2.220446e-16
1.023716 1.023716 2.220446e-16
1.032933 1.032933 -2.220446e-16
1.044056 1.044056 -2.220446e-16
1.057353 1.057353 0.000000e+00
1.073182 1.073182 -2.220446e-16
1.092029 1.092029 -4.440892e-16
1.114564 1.114564 -8.881784e-16
1.141748 1.141748 -4.440892e-16
1.175005 1.175005 -6.661338e-16
1.216574 1.216574 -8.881784e-16
1.270249 1.270249 -1.332268e-15
1.343227 1.343227 -3.774758e-15
1.451843 1.451843 -4.884981e-15
1.648852 1.648852 2.220446e-16
```