University of Helsinki / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 03 / Solutions

1. Suppose that the coefficients of a polynomial are known (for instance, generate 10 polynomials with random coefficients). Find the roots with the command "roots" and write the coefficients and the the real and imaginary parts of the roots in a file. Also plot the graph of the function.

Solution:

```
% FILE d031.m begins.
close all
fid=fopen('h031.dat','w');
xx=0:0.01:3;
for m=1:10
  coef= 6*0.001*fix(1000*rand(1,3));
  rt=roots(coef);
  fprintf(fid,'\nKertoimet: ');
  fprintf(fid,' %12.6e',coef');
%TAPA1:
  fprintf(fid,'\nJuuret:');
  fprintf(fid,' %s', mat2str(rt,6));
%TAPA2:
%  fprintf(fid,'\nJuuret:\n');
%  fprintf(fid,'   Re(z)      Im(z) \n');
%  p=size(rt);
%  for pp=1:p
%    fprintf(fid,' %12.6e   %12.6e\n', real(rt(pp)), imag(rt(pp)));
%  end
  yy=polyval(coef,xx);
  plot(xx,yy,'LineWidth',2);
  hold on
end
grid
fclose(fid);
% FILE h031.m ends.
```

Output:

```
Kertoimet:  5.700000e+00 1.386000e+00 3.636000e+00
Juuret: [-0.121579+i*0.789375;-0.121579-i*0.789375]
Kertoimet:  2.910000e+00 5.346000e+00 4.572000e+00
Juuret: [-0.918557+i*0.85287;-0.918557-i*0.85287]
Kertoimet:  2.736000e+00 1.080000e-01 4.926000e+00
Juuret: [-0.0197368+i*1.34166;-0.0197368-i*1.34166]
Kertoimet:  2.664000e+00 3.690000e+00 4.746000e+00
Juuret: [-0.692568+i*1.141;-0.692568-i*1.141]
Kertoimet:  5.526000e+00 4.428000e+00 1.056000e+00
Juuret: [-0.400651+i*0.174857;-0.400651-i*0.174857]
Kertoimet:  2.430000e+00 5.610000e+00 5.496000e+00
Juuret: [-1.15432+i*0.963987;-1.15432-i*0.963987]
Kertoimet:  2.460000e+00 5.358000e+00 3.420000e-01
Juuret: [-2.11223;-0.0658188]
Kertoimet:  2.112000e+00 4.878000e+00 5.400000e-02
Juuret: [-2.29854;-0.0111237]
Kertoimet:  8.280000e-01 1.212000e+00 1.188000e+00
Juuret: [-0.731884+i*0.948224;-0.731884-i*0.948224]
Kertoimet:  3.618000e+00 1.632000e+00 1.188000e+00
Juuret: [-0.225539+i*0.526774;-0.225539-i*0.526774]
```

2. (a) Prove that $\int_a^b \int_c^d xy\,dx\,dy = (d^2 - c^2)(b^2 - a^2)/4$ for $b > a, d >$

(b) Use the MATLAB function doubleint0.m on the www-page to co pute this integral when $(a, b, c, d) = (0, 3, 0, 2)$ and tabulate the differe exact value minus numerical value when the number $m[n]$ of subdivisi in the $x[y]$ direction has the value $m = 10 : 20 : 90, n = 10 : 20 : 90$. may do this as follows

```
exact=  (d^2- c^2)*(b^2 -a^2)/4;
for m=10:20:90
for n=10:20:90
    numer=doubleint0(a,b,c,d,m,n);
    fprintf('  %12.3e', numer-exact);
end
fprintf('\n')
end
```

Solution:

```
% FILE doubleint0.m begins.
% Compute \int_{x_1}{x_2}  \int_{y_1}^{y_2} myf(x,y) dx dy
% This is based on the Riemann sum with m (n) steps in x (y) direc
function val=doubleint0(x1,x2,y1,y2,m,n)
a=x1; b=x2;
s=0.0;
```

```
for ii=0:(m-1)
      x=a+ii*(b-a)/m;
for jj=0:(n-1)
      dy=(y2-y1)/n;
      y=y1+jj*dy;
      s=s+myf(x,y)*dy*(b-a)/m;
end
end
val=s;
% fprintf('%12.5f\n', s)

function z= myf(x,y)
      z= x.*y;
% FILE doubleint0.m ends.
```

```
% FILE h032.m begins.
a=0; b=3; c=0;d=2;
exact=  (d^2- c^2)*(b^2 -a^2)/4;
for m=10:20:90
  for n=10:20:90
    numer=doubleint0(a,b,c,d,m,n);
    fprintf('  %12.3e', numer-exact);
  end
  fprintf('\n')
end
% REMARK: The built-in MATLAB function gives
% accuracy of the order 10E-16
% r2=dblquad('x.*y',a,b,c,d);
% fprintf('%12.3e\n', exact-r2)
% FILE h032.m ends.
```

Output:

```
 -1.710e+00    -1.170e+00    -1.062e+00    -1.016e+00    -9.900e-01
 -1.170e+00    -5.900e-01    -4.740e-01    -4.243e-01    -3.967e-01
 -1.062e+00    -4.740e-01    -3.564e-01    -3.060e-01    -2.780e-01
 -1.016e+00    -4.243e-01    -3.060e-01    -2.553e-01    -2.271e-01
 -9.900e-01    -3.967e-01    -2.780e-01    -2.271e-01    -1.989e-01
```

3. The eigenvalues (=characteristic roots) of an $n \times n$ complex matrix $(a_{i,j})$ lie in the closed region of the z-plane consisting of all the disks

$$|a_{i,i} - z| \leq \sum_{j=1, j \neq i}^{n} |a_{i,j}| \, , j = 1, ..., n \, .$$

These are so called Gerschgorin disks. Check the validity of this statem
as follows:

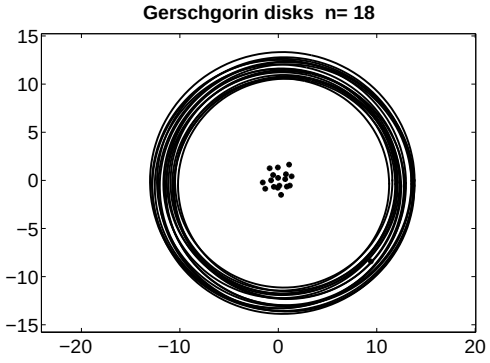
(a) For each n=3:3:18 generate a random complex $n \times n$ matrix
compute its eigenvalues.

(b) For each case plot the Gerschgorin disks and check visually that
statement holds.

Solution:

```
% FILE d033.m begins.
path(path,'../util')
close all
theta=0:0.05:6.3;
x=cos(theta);
y=sin(theta);

for n=3:3:18

a=rand(n,n)-i*rand(n,n);
myeig=eig(a);
figure
h=axes;
set(h,'FontWeight','bold','FontSize',20)
for j=1:n
myrad=sum(abs(a(j,:)))-2*abs(a(j,j));
% xx, yy are the x- and y-coordinates of the
% circle centered at a(j,j) with radius myrad
xx=real(a(j,j))+myrad*x;
yy=imag(a(j,j))+myrad*y;
pic=plot(xx,yy,'b-',...
real(myeig(j)), imag(myeig(j)),'k*');
set(pic,'LineWidth',2)
txt=[' n= ' num2str(n)];
title(['Gerschgorin disks ' txt],...
'FontWeight','bold','FontSize',20)
hold on
axis equal

end
widemarg(gcf)

if (n==6)
print -dps d033.ps
end
```

```
end

% FILE d033.m ends.
```



4. The fixed point method for numerical solution of $f(x) = x$ when $f : \mathbb{R} \to \mathbb{R}$ is based on the fixed point iteration $x_{n+1} = f(x_n)$. This converges for all $x_0 \in [a, b]$ if there exists $c \in (0, 1)$ such that $|f'(x)| < c$. Choose a suitable $\lambda \neq 0$ such that the iteration $x_{n+1} = x_n + (1 - x_n - \sin x_n)/\lambda \equiv g(x_n)$ converges to the root of $1 - x - \sin x = 0$. Then use the method with a fixed λ to find the root.

Solution:

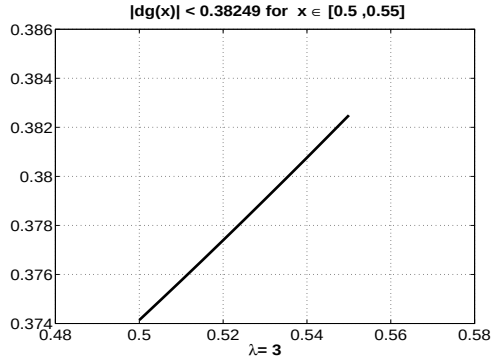
```
% FILE h034.m begins.
function h034
close all
myf=inline('1-x-sin(x)');
x=0:0.05:1;   % Note: myf(0) >0 >myf(1)
y=myf(x);
a=max(x(y>0)); b=a+x(2)-x(1);
myg=inline('x+ (1-x-sin(x))/lam ','x', 'lam');
mydg=inline('1-(1+cos(x))/lam ','x', 'lam');
% Can we find lam \neq 0 and c \in (0,1)
% such that |g'(x)| <c for all x in [a,b]?

mydata=[];
figure
axes('FontSize',[18],'FontWeight','bold');

for j=1:10          % In this loop we find the lambda that gives
```



```
                    % the smallest c
  lam=1+0.2*j;
  xx=a:((b-a)/10):b;
  yy=abs(mydg(xx,lam));
  myy=max(yy);
  plot(xx,yy,'LineWidth',3)
  ax=axis; ax(1)=ax(1)-0.02;
  axis(ax)
  grid on
  txt=[' |dg(x)| < '  num2str(max(myy)) ' for  x \in ['  ];
  txt=[txt  num2str(a) ' ,' num2str(b) ']'];
  title(txt,  'FontSize',18,  'FontWeight','bold')
  xlabel(['\lambda= ' num2str(lam) ],  'FontSize',18,'FontWeight',
  pause(2)
  mydata=[mydata;  lam  myy ];
end
[myc,j]=min(mydata(:,2));    % j is the index of the smallest c
lam=  mydata(j,1);           % We choose lam(j)
fprintf('lambda= %8.4f\n', lam);
x=(a+b)/2;      % Start fixed point iteration at x
                % This loop is for fixed point iteration:
for j=1:10
x=myg(x,lam);
fprintf('%2d .  %16.12f\n',j,x);
end
% FILE h034.m ends.
```

Output:

```
lambda=   1.8000
 1 .    0.510437219626
 2 .    0.510994996924
 3 .    0.510972563529
 4 .    0.510973464152
 5 .    0.510973427993
 6 .    0.510973429445
 7 .    0.510973429386
 8 .    0.510973429389
 9 .    0.510973429389
10 .    0.510973429389
```



5. A $m \times n$ matrix $A = (a_{ij})$ is called upper triangular if $a_{ij} = 0$ whenever $i > j$. Generate upper triangular 7×7 matrices and study experimentally whether (a) the product of two such matrices is again upper triangular, (b) an upper triangular matrix has an upper triangular matrix as the inverse (c) whether the determinant is always nonzero. Write a program to solve the upper triangular linear $n \times n$ system of equations.

Solution:

```
% FILE h035.m begins.
function h035
clear
m=4;
a1=uppertri(m,m);
b1=uppertri(m,m);
c1=a1*b1;
disp('Inverse of a triangular matrix is: ')
inv(a1)
rhs=rand(1,m);
x=trisolve(a1,rhs)
fprintf('Residual =|ax-b| = %12.4e\n',norm(a1*(x')-rhs'))

function a=uppertri(m,n)
a=rand(m,n);
for i=1:m
  for j=1:n
      if (i>j)
      a(i,j)=0.0;
      end
```



```
  end
end

function x=trisolve(a,b)
[d1,d2]=size(a);
if (d1 ~=d2)
  error('Must be square matrix for trisolve')
end
x=zeros(size(b));
x(d1)=b(d1)/a(d1,d1);
for j=(d1-1):(-1):1
  x(j) =(b(j) -sum( a(j,(j+1):d2 ).*x((j+1):d2)  ) )/a(j,j);
end
% FILE h035.m ends.
```

Output:

```
Inverse of a triangular matrix is:

ans =

    1.5794   -0.4694   -2.2559    0.1502
         0    2.0747   -0.6936   -2.7232
         0         0    1.9393   -0.8056
         0         0         0    2.9053


x =

   -1.0705    0.4491    1.2601    0.1185

Residual =|ax-b| =   7.8505e-17
```

6. MATLAB has a built-in function magic which with call magic generates an $n \times n$ matrix with positive integer entries such that the and column sums are all constant (such a matrix is called a magic matr Is it true that the inverse of a magic matrix also is a magic matrix (n the entries of the inverse are no longer integers so it would be helpfu use "format rational" to see the exact entries, not only their numer approximations). If a is a magic matrix, is it true that also a^2 is?

Solution:

```
% FILE h036.m begins.
% It seems that:
% (a) the inverse of an m xm matrix is magic if m=2n+1
% (b) the square of a magic matrix also is magic
for n=3:9
  fprintf('\nn = %1.d\n', n);
  a=magic(n);
  tmp=sum(a);
  tmp2=sum(a');
  if norm(diff(tmp))+norm(diff(tmp2))+norm(tmp(1)-tmp2(1)) <1e-20
    fprintf('a is magic (with sums of %d)\n',sum(sum(a))/n);
  else fprintf('a is not magic\n');
  end
  format rational
  if (abs(det(a))>1e-20)
    b=inv(a);
    tmp= sum(b);
    tmp2= sum(b');
    if norm(diff(tmp))+norm(diff(tmp2))+norm(tmp(1)-tmp2(1)) <1e-10
      fprintf('b=inv(a) is magic (with sums of 1/%d)\n',1/(sum(sum(b))/n));
    else fprintf('b=inv(a) is not magic\n');
    end
  else fprintf('b=inv(a) is not magic\n');
  end
  c= a^2;
  tmp=sum(c);
  tmp2= sum(c');
  if norm(diff(tmp))+norm(diff(tmp2))+norm(tmp(1)-tmp2(1)) <1e-20
    fprintf('c=a^2 is magic (with sums of %d)\n',sum(sum(c))/n);
  else fprintf('c=a^2 is not magic\n');
  end
  d= a.^2;
  tmp=sum(d);
  tmp2= sum(d');
  if norm(diff(tmp))+norm(diff(tmp2))+norm(tmp(1)-tmp2(1)) <1e-20
    fprintf('d=a.^2 is magic (with sums of %d)\n',sum(sum(d))/n);
  else fprintf('d=a.^2 is not magic\n');
  end
end
format
% FILE h036.m ends.
```


Output:

```
n = 3
a is magic (with sums of 15)
b=inv(a) is magic (with sums of 1/15)
c=a^2 is magic (with sums of 225)
d=a.^2 is not magic

n = 4
a is magic (with sums of 34)
b=inv(a) is not magic
c=a^2 is magic (with sums of 1156)
d=a.^2 is not magic

n = 5
a is magic (with sums of 65)
b=inv(a) is magic (with sums of 1/65)
c=a^2 is magic (with sums of 4225)
d=a.^2 is not magic

n = 6
a is magic (with sums of 111)
b=inv(a) is not magic
c=a^2 is magic (with sums of 12321)
d=a.^2 is not magic

n = 7
a is magic (with sums of 175)
b=inv(a) is magic (with sums of 1/1.750000e+02)
c=a^2 is magic (with sums of 30625)
d=a.^2 is not magic

n = 8
a is magic (with sums of 260)
b=inv(a) is not magic
c=a^2 is magic (with sums of 67600)
d=a.^2 is not magic

n = 9
a is magic (with sums of 369)
b=inv(a) is magic (with sums of 1/369)
c=a^2 is magic (with sums of 136161)
d=a.^2 is not magic
```

Solutions in Python

Problem 1.

```
# FILE d031.py begins.
from Numeric import *
from scipy import *
import random

def rand(n):
   xx=1.0*arange(n)
   for i in range(n):
      xx[i]=random.uniform(0.1,1.0)
   return xx

for m in range(1,10):
  coef= 6.0*0.001*floor(1000.0*rand(3))
  rt=roots(coef)
  print '\nKertoimet: '
  print coef
  print 'Juuret:'
  print rt
# FILE d031.py ends.
```

Output:

```
Kertoimet:
[ 0.948  5.886  2.034]
Juuret:
[-5.84156727+0.j -0.36729349+0.j]

Kertoimet:
[ 1.368  3.012  1.458]
Juuret:
[-1.48316128+0.j -0.71859311+0.j]

Kertoimet:
[ 5.814  3.852  5.304]
Juuret:
[-0.33126935+0.89584671j -0.33126935-0.89584671j]
```



```
Kertoimet:
[ 3.552  2.46   5.112]
Juuret:
[-0.34628378+1.14859772j -0.34628378-1.14859772j]

Kertoimet:
[ 3.186  3.834  1.152]
Juuret:
[-0.62302213+0.j -0.5803677 +0.j]

Kertoimet:
[ 2.532  5.904  3.348]
Juuret:
[-1.35821444+0.j -0.97353911+0.j]

Kertoimet:
[ 3.09   1.176  2.826]
Juuret:
[-0.19029126+0.93720454j -0.19029126-0.93720454j]

Kertoimet:
[ 3.264  3.03   1.458]
Juuret:
[-0.46415441+0.48088653j -0.46415441-0.48088653j]

Kertoimet:
[ 3.942  4.452  4.776]
Juuret:
[-0.56468798+0.9448255j -0.56468798-0.9448255j]
```

Problem 2.

```
# FILE doubleint0.m begins.
# Compute \int_{x_1}{x_2}  \int_{y_1}^{y_2} myf(x,y) dx dy
# This is based on the Riemann sum with m (n) steps in x (y)
# direction
def myf(x,y):
   return x*y

def doubleint0(x1,x2,y1,y2,m,n):
   a,b=x1,x2
   s=0.0
```

```
   for ii in range(0,m):
      x=a+ii*(b-a)/m
      for jj in range(n):
         dy=(y2-y1)/n
         y=y1+jj*dy;
         s=s+myf(x,y)*dy*(b-a)/m
   return s
# FILE doubleint0.m ends.

# FILE h032.py begins.
from Numeric import *
from doubleint0 import *
a,b,c,d=0.0,4.0,0.0,1.0
exact=  (d*d- c*c)*(b*b -a*a)/4.0
for m in range(100,200,20):
  for n in range(100,200,20):
    numer=doubleint0(a,b,c,d,m,n)
    print '  %12.3e' %(numer-exact)
  print
# FILE h032.py ends.
```

Output:

```
 -7.960e-02
 -7.300e-02
 -6.829e-02
 -6.475e-02
 -6.200e-02

 -7.300e-02
 -6.639e-02
 -6.167e-02
 -5.813e-02
 -5.537e-02

 -6.829e-02
 -6.167e-02
 -5.694e-02
 -5.339e-02
 -5.063e-02

 -6.475e-02
```



```
 -5.813e-02
 -5.339e-02
 -4.984e-02
 -4.708e-02

 -6.200e-02
 -5.537e-02
 -5.063e-02
 -4.708e-02
 -4.432e-02
```

Problem 3.

```
# FILE d033.py begins
from mmeutil import *
from scipy import *
from Numeric import *
from LinearAlgebra import *

theta=0.05*arange(0,127)
x=cos(theta)
y=sin(theta)

n=3*arange(1,6)

for i in n:
    a=ranmat(i,i)-ranmat(i,i)*(1j)
    myeig=eigenvalues(a)
    for k in range(0,i):
        myrad=sum(abs(a[k,:]))-2*abs(a[k,k])
        # xx, yy are the x- and y-coordinates of the
        # circle centered at a(j,j) with radius myrad
        xx=real(a[k,k])+myrad*x
        yy=imag(a[k,k])+myrad*y
        gplt.plot(xx,yy,'notitle w l')
        gplt.hold('on')
        gplt.plot(array([real(myeig[k]),real(myeig[k])]),\
          array([imag(myeig[k]),imag(myeig[k])]),'notitle w
    gplt.title('n = '+str(i))
    gplt.hold('off')
    gplt.output('d033_'+str(i)+'.png','png color')
```

```
# FILE d033.py ends
```

Problem 4.

```
# FILE h034.py begins.
from Numeric import *

def myf(x): return 1.0-x-sin(x)
def myg(x,lam): return x+(1.0-x-sin(x))/lam
def mydg(x,lam): return 1.0-(1.0+cos(x))/lam
# Can we find lam \neq 0 and c \in (0,1)
# such that |g'(x)| <c for all x in [a,b]?

x=arange(0.0,1.01,0.05) # Note: myf(0) >0 >myf(1)
y=myf(x)
a=max(x*(y>0))
b=a+x[1]-x[0]
mydata=[]
for j in range(1,11): # In this loop we find the lambda that gives
                      # the smallest c
   lam=1.0+0.2*j
   xx=arange(a,b,((b-a)/10))
   yy=abs(mydg(xx,lam))
   myy=max(yy)
   mydata.append([lam,myy]);
mydata=array(mydata)
myc=min(mydata[:,1])
n=len(mydata[:,1])
j=min(arange(n)*(mydata[:,1]==myc))
# j is the index of the smallest c
lam=mydata[j,0]           # We choose lam(j)
print 'lambda= %8.4f'%(lam)
x=(a+b)/2.0     # Start fixed point iteration at x
                # This loop is for fixed point iteration:
for j in range(1,11):
   x=myg(x,lam)
   print '%2d .  %16.12f'%(j,x)
# FILE h034.py ends.
```

Output:

```
lambda=   1.2000
```



```
 1 .    0.503155829439
 2 .    0.515365431964
 3 .    0.508516864560
 4 .    0.512350883986
 5 .    0.510202133092
 6 .    0.511405649303
 7 .    0.510731327463
 8 .    0.511109072657
 9 .    0.510897442513
10 .    0.511016000242
```

Problem 5.

```
# FILE h035.py begins.
from Numeric import *
from LinearAlgebra import *
import random

def norm(a): return sum(abs(a*a))

def rand(n):
   xx=1.0*arange(n)
   for i in range(n):
      xx[i]=random.uniform(0.1,1.0)
   return xx

# Solves a linear system of the form Ux = b'
# where U is an upper diagonal matrix
def usolve(a,y):
   m,n=shape(a)
   x=arange(1.*n) # initialize the vector x
                  # (note the type)
   # we begin to solve the system from the last row
   for i in range(n-1,-1,-1):
      x[i]=y[i]/a[i][i]  # x[i] has been solved
      # we substitute the value of x[i] into the
      # remaining equations
      for j in range(i):
         y[j]-=x[i]*a[j][i]
   return x

def uppertri(m,n):
```

```
   aa=rand(m*n)
   a = reshape(aa,(m,n))
   for i in range(m):
      for j in range(n):
        if i>j: a[i,j]=0.0
   return a

m=4
a1=uppertri(m,m)
b1=uppertri(m,m)
c1=dot(a1,b1)
print 'Inverse of a triangular matrix is: '
a2=array(a1)
print inverse(a2)
rhs=rand(m)
r=array(rhs)
x=usolve(a1,r)
print 'Residual =|ax-b| = %12.4e' %(norm(dot(a1,x)-rhs))
# FILE h035.py ends.
```

Output:

```
Inverse of a triangular matrix is:
[[ 1.42857666 -0.14887985 -0.97912772 -0.45876257]
 [ 0.          1.01202884 -0.4223571  -3.4201084 ]
 [ 0.          0.          2.48794232 -5.25502343]
 [ 0.          0.          0.          4.31533165]]
Residual =|ax-b| =   3.2433e-31
```

Problem 6.

```
# FILE h036.m begins.
# It seems that:
# (a) the inverse of an m xm matrix is magic if m=2n+1
# (b) the square of a magic matrix also is magic
from Numeric import *
from LinearAlgebra import *
from magic import *
def norm(a): return sum(abs(a*a))

for n in range(3,10):
  a=magic(n)
```



```
  tmp=sum(a)
  tmp=tmp-tmp[0]
  if norm(tmp) <1e-10:
    print 'sum(a) =%f'%(sum(a)[0])
    print 'a is magic'
  if abs(determinant(a))>1e-20:
    b=inverse(a)
    tmp= sum(b)
    if norm(tmp) <1e-10:
      print 'sum(b) = %f'%(sum(b)[0])
      print 'b is magic'
  c= a*a
  tmp=sum(c)
  tmp=tmp-tmp[0]
  if norm(tmp) <1e-10:
    print 'sum(c) = %f'%(sum(c)[0])
    print 'c is magic'
  d= a*a
  tmp=sum(d)
  tmp=tmp-tmp[0]
  if norm(tmp) <1e-10:
    print 'tmp=sum(d) =' %(sum(d)[0])
    print 'd is magic'
# FILE h036.m ends.
```

Output:

```
sum(a) =15.000000
a is magic
sum(a) =34.000000
a is magic
sum(a) =65.000000
a is magic
sum(a) =175.000000
a is magic
sum(a) =260.000000
a is magic
sum(a) =369.000000
a is magic
```