University of Helsinki / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 07 / Solutions

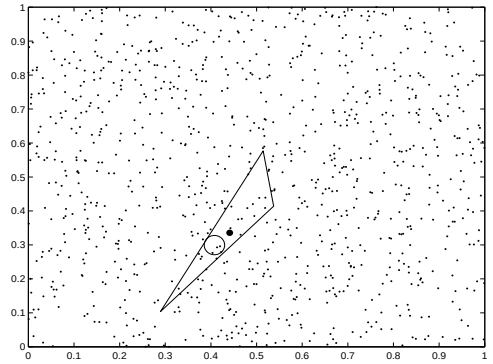
1. Use getpts to draw a triangle and generate random points in a polygon. Compute the mean of the random points and mark it on the polygon. Also compute the mean value of the points in the triangle. Hint: `help inpolygon`.

Solution:

```
% FILE d071.m begins.
close all
w=rand(3,2);
xv=w(:,1); yv=w(:,2);
xv = [xv ; xv(1)]; yv = [yv ; yv(1)];
x = rand(1000,1); y = rand(1000,1);
in = inpolygon(x,y,xv,yv);
plot(xv,yv,x(in),y(in),'.r',x(~in),y(~in),'.b')
xm=mean(x(in))
xp=mean(xv)
ym=mean(y(in))
yp=mean(yv)
hold on
plot(xm,ym,'k.','MarkerSize',20)
plot(xp,yp,'ro','MarkerSize',20)
% FILE d071.m ends.
```

Output:

```
xm =

    0.6142


xp =

    0.6846


ym =

    0.7002


yp =

    0.6563
```



2. Consider functions $f(z) = \sum_{n=0}^{\infty} a_n z^n$ with complex coefficients that satisfy $|f(z)| \leq 1$ for all $z, |z| \leq 1$. For instance, if a sequence (of complex numbers with $S = \sum_{n=0} |a_n| < \infty$ is given, then the funct $f(z) = \sum_{n=0}^{\infty} b_n z^n$, $b_n = a_n/S$ has this property.

(a) Generate polynomial functions with complex coefficients ($a_n = 0$ all large n) and plot image of the circle $\{z : |z| = r\}$ when $r \in (0,1)$. M the point on this circle where $\sup\{|f(z)| : |z| \leq r\}$ is attained. Obse Usually this point is not on the real positive real axis. Can you conf this with your experiment?

(b) An inequality due to Bohr (Lond. M. S. Proc. (2) 13 (1913), states that under the above hypotheses, $\sum_{n=0}^{\infty} |a_n| r^n \leq 1$ for all $r \in (0,1$ Verify this statement.

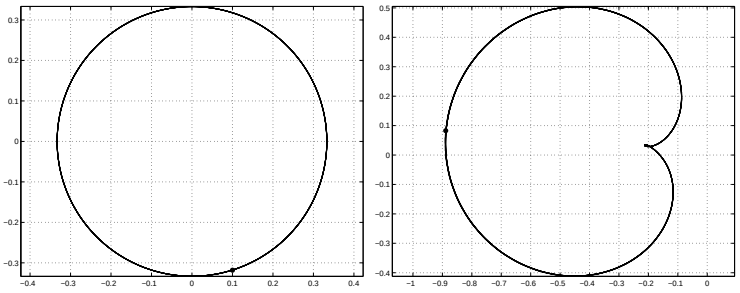
Solution:

```
% FILE d072.m begins.
close all
m=1000;
a= rand(1,m) -0.5*rand(1,m);
a= a-i*(rand(1,m) -0.5*rand(1,m));
a=100*a;
s= sum(abs(a));
```

```
b= a/s;
theta=0:0.1:63;
r=1/3;
z=r*(cos(theta)+i*sin(theta));
w=polyval(b,z);
[mymax,imax]=max(abs(w));

figure
plot(real(z), imag(z),real(z(imax)), imag(z(imax)),'k.','MarkerSize', 20)
hold on
grid on
axis equal
figure
w=1000*w;
plot(real(w), imag(w),real(w(imax)), imag(w(imax)),'k.','MarkerSize', 20)
hold on
axis equal
grid on
t=1- max( polyval(  abs(b), z));
fprintf('(b): 1- polyval(abs(a),r ) %12.4e\n',t)
% FILE d072.m ends.
```

Output:

(a)



```
(b): 1- polyval(abs(a),r )   9.9935e-01
```

3. Recall the function $\mathrm{erf}(x)$ studied in Problem 1/Exercise 5. Utilize it to express the function $P(x)$ in terms of $\mathrm{erf}(x)$ and to tabulate the values $P(x), x = 0 : 0.2 : 2$ when $P(x) = \int_{-\infty}^{x} Z(t)dt$ and $Z(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.

You may want to look at the table 26.1 on p. 966 of Abramowitz-Ste
http://www.math.sfu.ca/~cbm/aands/, which contains these values.

Solution:

```
% FILE d073.m begins.
close all
myP=inline('0.5+0.5*(erf(x/sqrt(2))-erf(0))','x');
x=0:0.1:2;
y=myP(x);
figure
axes('FontSize',[18],'FontWeight','bold');
plot(x,y)
grid on
title('P(x)', 'FontSize',[18],'FontWeight','bold');
xlabel('x', 'FontSize',[18],'FontWeight','bold');
fprintf('  %6.2f  %12.10f\n', [x;y])
% FILE d073.m ends.
```

Output:

```
  0.00  0.5000000000
  0.10  0.5398278373
  0.20  0.5792597094
  0.30  0.6179114222
  0.40  0.6554217416
  0.50  0.6914624613
  0.60  0.7257468822
  0.70  0.7580363478
  0.80  0.7881446014
  0.90  0.8159398747
  1.00  0.8413447461
  1.10  0.8643339391
  1.20  0.8849303298
  1.30  0.9031995154
  1.40  0.9192433408
  1.50  0.9331927987
  1.60  0.9452007083
  1.70  0.9554345372
  1.80  0.9640696809
  1.90  0.9712834402
  2.00  0.9772498681
```

4. Recall from linear algebra that $[a,b;c,d]^{-1} = T[d,-b;-c\,a]$ if $1/T = ad-bc \neq 0$. Recall also that the Newton method for solving $h(w)=0, h: \mathbb{R}^2 \to \mathbb{R}^2$ is the iteration $w_{n+1} = w_n - J_h(w_n)^{-1}h(w_n), n = 0,1,2,3,...$

a) Utilize the linear algebra formula to show that the Newton method for solving $f(x,y)=0, g(x,y)=0$ yields the sequence ($f_1 = \partial f/\partial x, f_2 = \partial f/\partial y$)

$$x_{n+1} = x_n - \frac{fg_2 - gf_2}{f_1g_2 - g_1f_2}, \quad y_{n+1} = y_n - \frac{f_1g - g_1f}{f_1g_2 - g_1f_2}.$$

(b) Apply this method when $f(x,y) = x^2+y^2-1, g(x,y) = y - e^x$ with $x_0 = -0.9, y_0 = 0.2$.

Solution:

```
% FILE d074.m begins.
% (b)
myf=inline('x.^2+y.^2-1','x','y');
myg=inline('y-exp(x)','x','y');
myf1=inline('2*x','x');
myf2=inline('2*y','y');
myg1=inline('-exp(x)','x');
myg2=inline('1','y');
x=-0.9; y= 0.2;
val=[0, x ,y,  myf(x,y), myg(x,y)];
fprintf('%3s   %8s   %8s   %12s   %12s  \n',...
        'n','x','y','f(x,y)','g(x,y)')
for j=1:5
den=myf1(x)*myg2(y)-myg1(x)*myf2(y);
x=x -(myf(x,y)*myg2(y)-myg(x,y)*myf2(y))/den;
y=y -(myf1(x)*myg(x,y)-myg1(x)*myf(x,y))/den;
val=[val;j x ,y,  myf(x,y), myg(x,y)];
end
fprintf('%3d   %8.5f   %8.5f   %12.4e   %12.4e  \n',val')
% FILE d074.m ends.
```

Output:

n	x	y	f(x,y)	g(x,y)
0	-0.90000	0.20000	-1.5000e-01	-2.0657e-01
1	-0.94115	0.40094	4.6506e-02	1.0756e-02
2	-0.91701	0.39993	8.4820e-04	2.1536e-04
3	-0.91656	0.39989	3.0146e-07	8.6263e-08
4	-0.91656	0.39989	3.8192e-14	1.3045e-14
5	-0.91656	0.39989	0.0000e+00	0.0000e+00

5. Use parfit.m or parf04.m (instead of the algorithm of Problem 2 Exercise 6) to fit the model

$$y = \lambda_1 + \lambda_2 * \sin(2 * \pi * (x - \lambda_3)/24))$$

to the data from Problem 2 of Exercise 6

```
x   0.0  2.0  4.0  6.0  8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0 24.
y   6.3  4.0  6.6 10.9 14.6 19.1 24.3 25.7 22.9 19.5 15.9 10.3  5.
```

Solution: lecture notes, Section 5.

```
% FILE  d075.m  begins.
% Fits a nonlinear model depending on parameter vector lam
% to xdata, ydata
% Define fmodel and fobj as inline functions:
fmodel2=inline(' lam(1)+lam(2)*sin(2*pi*(x-lam(3))/24)','x','lam')
fobj2=inline('norm(feval(fmodel2,x,lam)-y)','lam','fmodel2','x','y
x= [  0.0  2.0  4.0  6.0  8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0 2
y= [  6.3  4.0  6.6 10.9 14.6 19.1 24.3 25.7 22.9 19.5 15.9 10.3
xdata=x;
ydata=y;
% Initial guess for lambda
lam0=[1 1 8];
% Initial value of the object function (before minimization)
y0=fobj2(lam0,fmodel2,xdata,ydata);
% lam is the fitted value for the parameter vector
lam=fminsearch(fobj2,lam0,[],fmodel2,xdata,ydata);
% Compute values of the fitted function for plotting
xi=0:0.05:24;
yfit=fmodel2( xi,lam);
% Final value of the object function (after minimization)
yfinal=fobj2(lam,fmodel2,xdata,ydata);
clf;
axes('FontSize',20,'FontWeight','bold'); hold on;
title(['Object function values: before =' num2str(y0) ', after = '
       num2str(yfinal)])
plt=plot(xi,yfit,xdata,ydata,'k.','MarkerSize',30);  grid;
txt1=' {\bf Fitted curve (solid)}';
%text(x(5),yfit(5),txt1,'FontWeight','bold','FontSize',20);
txt2=' {\bf Data point (dots)}';
%text(xdata(8),ydata(8),txt2,'FontWeight','bold','FontSize',20);
text(0.2,27,['lam= ' mat2str(lam,4)],...
     'FontWeight','bold','FontSize',20);
```

```
text(0.2,2.7, ...
     ['lam(1)+lam(2)*sin(2*pi*(x-lam(3))/24)'],...
     'FontWeight','bold','FontSize',20);
ylabel('ydata '); xlabel('xdata (d083/MME03)')
set(plt,'LineWidth',2.5);
fprintf('lambda=\n');       fprintf(' %6.4f ',lam)
fprintf('\n')
% FILE  d075.m  ends.
```

Output:

```
lambda=
 14.9446  10.1278  -16.0066
```

6. Familiarize yourself with the content of the MATLAB help information for the commands griddata and contour. Use the corresponding examples from the MATLAB helpdesk to graph the temperature over Finland, given the x,y, coordinates and the temperatures in the following cities

```
nimi=str2mat('Turku','Tre','Hki','Oulu', 'Jkyla','Mikkeli', 'Vaasa');
x= [0 100 160    40 160 400   0];
y= [0  80   0   500 200 100 300];
t= [10  8  10     5   5   3   7];
```

Use this data and the function interp2 to obtain by interpolation the temperature in Salo (x=50, y=0).

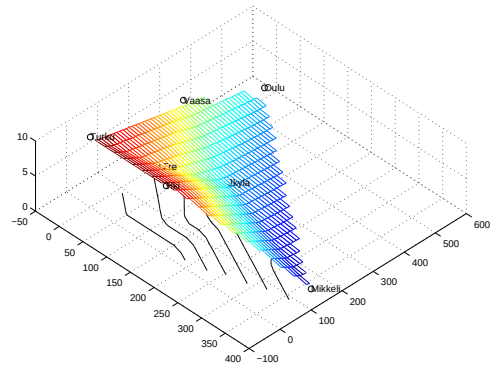
Solution:

```
% FILE d076.m begins.
% From MATLAB helpdesk:
close all
clear
set(0,'defaultfigurerenderer','painters');
opengl('neverselect');

rand('seed',0)
x = rand(100,1)*4-2; y = rand(100,1)*4-2;
z = x.*exp(-x.^2-y.^2);
% x, y, and z are now vectors containing nonuniformly sampled data.
% Define a regular grid, and grid the data to it:
```



```
ti = -2:.25:2;
[XI,YI] = meshgrid(ti,ti);
ZI = griddata(x,y,z,XI,YI);
% Plot the gridded data along with the nonuniform data points
% used to generate it:

mesh(XI,YI,ZI), hold
plot3(x,y,z,'o')

pause(3)
close all
% Some temperatures:
figure
clear
nimi=str2mat('Turku','Tre','Hki','Oulu', 'Jkyla','Mikkeli', 'Vaasa
x= [ 0 100 160  40 160 400   0];
y= [ 0  80   0 500 200 100 300];
t= [10   8  10   5   5   3   7];
xi=-10:20:400;
yi=-10:10:550;

[XI,YI]=meshgrid(xi,yi);
ZI= griddata(x,y,t, XI, YI);

mesh(XI,YI,ZI);
hold on;
plot3(x,y,t,'o');
text(x,y,t,nimi);
contour(XI, YI, ZI);
print -dps d066.ps
% Salo temperature:
z=interp2(XI,YI,ZI,50,0);
fprintf('Salo temperature = %4.1f\n',z)
% FILE d076.m ends.
```

Output:



Solutions in Python

Problem 1.

```
# FILE d071.py begins.
from mmeutil import *
from scipy import gplt
from Numeric import *

# function to check if point inside polygon
# See:
# http://openmap.bbn.com/~kanderso/performance/postscript/in-poly.ps
def inpoly(npol, xp, yp, x,y):
    c = False
    j = npol - 1
    for i in range(npol):
        if ((yp[i] <= y < yp[j] or yp[j] <= y < yp[i]) and
            (x<(xp[j] - xp[i]) * (y-yp[i]) / (yp[j] -yp[i]) + xp[i])):
            c = not c
        j = i
    return c

def mean(x): return sum(x)/len(x)

w=ranmat(3,2)
#xv=w
xv=w[:,0]
#yv=w
```



```
yv=w[:,1]
x = ranvec(1000)
y = ranvec(1000)
xi = list()
yi = list()
for j in range(len(x)):
   if inpoly(3,xv,yv,x[j],y[j]):
      xi.append(x[j])
      yi.append(y[j])
gplt.plot(x,y,'notitle w p')
gplt.hold('on')
gplt.plot(xi,yi,'notitle w p pt 3')
gplt.plot(xv,yv,'notitle w l')
xm=mean(xi)
xp=mean(xv)
ym=mean(yi)
yp=mean(yv)
gplt.plot([xm],[ym],'notitle w p pt 4 ps 3')
gplt.plot([xp],[yp],'notitle w p pt 5 ps 3')
# FILE d071.py ends.
```

Problem 3.

```
# FILE d073.py begins.
from scipy import gplt
from scipy.special import *
from Numeric import *
def myP(x): return 0.5+0.5*(erf(x/sqrt(2))-erf(0))
x=arange(0,2.1,0.1)
y=myP(x)
gplt.plot(x,y)
for j in range(len(x)):
   print '  %6.2f  %12.10f' %(x[j],y[j])
# FILE d073.py ends.
```

Output:

```
  0.00  0.5000000000
  0.10  0.5398278373
  0.20  0.5792597094
  0.30  0.6179114222
  0.40  0.6554217416
```

```
   0.50  0.6914624613
   0.60  0.7257468822
   0.70  0.7580363478
   0.80  0.7881446014
   0.90  0.8159398747
   1.00  0.8413447461
   1.10  0.8643339391
   1.20  0.8849303298
   1.30  0.9031995154
   1.40  0.9192433408
   1.50  0.9331927987
   1.60  0.9452007083
   1.70  0.9554345372
   1.80  0.9640696809
   1.90  0.9712834402
   2.00  0.9772498681
```

Problem 4.

```
# FILE d074.py begins.
from Numeric import *
# (b)
def myf(x,y): return x*x+y*y-1
def myg(x,y): return y-exp(x)
def myf1(x): return 2*x
def myf2(y): return 2*y
def myg1(x): return -exp(x)
def myg2(y): return 1
x,y=-0.9,0.2
val=[[0, x ,y,  myf(x,y), myg(x,y)]]
print'%3s   %8s   %8s   %12s   %12s'%('n','x','y','f(x,y)','g(x,y)')
for j in range(5):
   den=myf1(x)*myg2(y)-myg1(x)*myf2(y)
   x=x -(myf(x,y)*myg2(y)-myg(x,y)*myf2(y))/den
   y=y -(myf1(x)*myg(x,y)-myg1(x)*myf(x,y))/den
   val.append([j, x ,y,  myf(x,y), myg(x,y)])
   print '%3d   %8.5f   %8.5f   %12.4e   %12.4e'%(j,x,y,myf(x,y),myg(x,y) )
# FILE d074.py ends.
```

Output:

```
  n          x          y         f(x,y)         g(x,y)
```



```
  0   -0.94115    0.40094     4.6506e-02     1.0756e-02
  1   -0.91701    0.39993     8.4820e-04     2.1536e-04
  2   -0.91656    0.39989     3.0146e-07     8.6263e-08
  3   -0.91656    0.39989     3.8636e-14     1.3101e-14
  4   -0.91656    0.39989    -1.1102e-16     0.0000e+00
```

Problem 5.

```
# FILE  d075.py begins.
# Fits a nonlinear model depending on parameter vector lam
# to xdata, ydata
from mmeutil import *
from scipy.stats import *      # histogram2
from scipy.optimize import *   # fmin (Nelder-Mead algorithm)
from RandomArray import *      # standard normal distributio
from Numeric import *
from scipy import *

global xdata
global ydata

def fmodel(lam,x):
   return lam[0]+lam[1]*sin(2*pi*(x-lam[2])/24)

def fobj(lam):
   return vnorm(fmodel(lam,xdata)-ydata)

x=array([0.0,2.0,4.0,6.0,8.0,10.0,12.0,\
  14.0,16.0,18.0,20.0,22.0,24.0])
y=array([6.3,4.0,6.6,10.9,14.6,19.1,24.3,\
  25.7,22.9,19.5,15.9,10.3,5.4])
xdata=x
ydata=y
lam0=array([1.0,1.0,8.0])  # Initial guess for lambda
y0=fobj(lam0)              # Initial value of object functio
lam=fmin(fobj,lam0)
showvec2(lam)
                      # lam is the fitted value for
                      # the parameter vector
x=0.1*arange(0.,150.)
yfit=fmodel(lam,x)
yfinal=fobj(lam)      # Final value of the object function
```

```
print fmodel(lam,12)
gplt.plot(x,yfit,'notitle w l')
gplt.hold('on')
gplt.plot(xdata,ydata,'notitle w p')
# FILE d075.py ends
```

Output:

```
Optimization terminated successfully.
         Current function value: 2.488443
         Iterations: 117
         Function evaluations: 213
  14.945
  10.128
 -16.007

23.7243076878
```