University of Helsinki / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 10 / Solutions

1. Suppose that we are given some data points $(x_k, y_k), k = 1, ..., m$, which seem to contain "two linear trends with a break point". For instance the data

```
x=-2:0.1:4; y=0.2*sin(3*x);
y(x<1)=y(x<1)+0.5*(x(x<1)-1);
y(x>=1)=y(x>=1)+2*(x(x>=1)-1);
```

has this feature with the break point $x = 1$. It would seem to be a natural idea to fit, instead of the LSQ line, "a line with a breakpoint" to the data.

Write a program that prompts the user to enter by a mouse click a point (s, t) in the plane (the simplest method is to use the built-in function ginput but also getpts.m or some modification of it could be used). Then fit a polygonal line with break point (s, t) to these points. That is, fit a line $y = k_1 x + b_1, x < s$, to the points $(x_k, y_k), x_k < s$, and a line $y = k_2 x + b_2, x > s$, to the points $(x_k, y_k), x_k > s$.

Solution:

```
% FILE d101.m begins.
% Create some artificial data
close all
x=-2:0.1:4; y=0.2*sin(3*x);
y(x<1)=y(x<1)+0.5*(x(x<1)-1);
y(x>=1)=y(x>=1)+2*(x(x>=1)-1);
count=1;
while count<=3
figure
axes('FontSize',[20],'FontWeight','bold');

plot(x,y,'ko')
hold on
grid on
disp('Click the mouse to enter the points')
[s,t]=ginput(1); %
fprintf('s= %12.6f , t= %12.6f\n',s,t)
x1=x(x<s); x2=x(x>=s);
y1=y(x<s); y2=y(x>=s);
% Use problem d021 formulas for k, b:
denom1=sum((x1-s).*(x1-s));
```

```
denom2=sum((x2-s).*(x2-s));
numer1=sum((x1-s).* (y1-t));
numer2=sum((x2-s).* (y2-t));
k1=numer1/denom1;
k2=numer2/denom2;
b1=(t-k1*s);
b2=(t-k2*s);
yy1=k1*x1+b1;
yy2=k2*x2+b2;
plot([ x1 x2],[yy1 yy2],'LineWidth',2)
fprintf('Coefficients k1, b1 %12.6f  %12.6f \n', k1,b1)
fprintf('Coefficients k2, b2 %12.6f  %12.6f \n', k2,b2)
s1=sum((y1-yy1).^2);
s2=sum((y2-yy2).^2);
fprintf('Sum of squares= %12.6f\n',s1+s2)
title(['Sum of squares= ' num2str(s1+s2,5)],...
      'FontWeight','bold','FontSize',20)
eval(['print -dps ' 'd065' num2str(count) '.ps'])
pause
count=count+1;
end
% FILE d101.m ends.
```

Output:

```
Click the mouse to enter the points
s=     1.103687 , t=    -0.187135
Coefficients k1, b1     0.365672     -0.590722
Coefficients k2, b2     2.179091     -2.592169
Sum of squares=     2.695747
Click the mouse to enter the points
s=     1.020737 , t=     0.093567
Coefficients k1, b1     0.519420     -0.436624
Coefficients k2, b2     1.953030     -1.899963
Sum of squares=     1.220940
Click the mouse to enter the points
s=     1.117512 , t=     0.233918
Coefficients k1, b1     0.562384     -0.394552
Coefficients k2, b2     1.978697     -1.977298
Sum of squares=     1.287504
```

2. Suppose that the vertices a, b, c of a triangle are on the boundary of the square $[0,1] \times [0,1]$. Within this triangle we choose 1000 triples of

random points (a_1, b_1, c_1) and for each such triple we compute the ratio of the area of the new triangle to the area of the triangle (a, b, c). What is the mean value of these ratios? Use the built-in function inpolygon to check that the points a_1, b_1, c_1 are within the original triangle.

Solution: The expectation of the ratio is $1/12$ according to G. Grimmett and D. Stirzaker: Probablity and Random Processes 3rd ed. 2001, p. 136. The value of 12*ratio given by this experiment is reasonably close to 1 as it should.

```
% FILE d102.m begins.


a=[rand, 0];
b=[1, rand];
c=[rand, 1];
pt=[a; b; c];
area1=polyarea(pt(:,1), pt(:,2))
count =0;
x1=min(pt(:,1));  x2=max(pt(:,1));
y1=min(pt(:,2));  y2=max(pt(:,2));
val=[];
while (count <1000)
  a1=x1+(x2-x1)*rand;
  a2=y1+(y2-y1)*rand;
  while (inpolygon(a1,a2, pt(:,1), pt(:,2))==0)
    a1=x1+(x2-x1)*rand;
    a2=y1+(y2-y1)*rand;
  end
  b1=x1+(x2-x1)*rand;
  b2=y1+(y2-y1)*rand;
  while (inpolygon(b1,b2, pt(:,1), pt(:,2))==0)
    b1=x1+(x2-x1)*rand;
    b2=y1+(y2-y1)*rand;
  end
  c1=x1+(x2-x1)*rand;
  c2=y1+(y2-y1)*rand;
  while (inpolygon(c1,c2, pt(:,1), pt(:,2))==0)
    c1=x1+(x2-x1)*rand;
    c2=y1+(y2-y1)*rand;
  end
  val=[val polyarea([a1, b1,  c1], [a2, b2, c2])];
  count=count+1;
end
fprintf('12*mean = %12.4f\n', 12*mean(val)/area1)
```

```
% FILE d102.m ends.
```

Output:

```
area1 =

    0.0646

12*mean =       0.9959
```

3. The triangle inequality states that

$$d(x,y) \leq d(x,z) + d(z,y)$$

for all $x, y, z \in \mathbb{R}^2$ if $d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$. Define now a new function

$$d_2(x,y) = |x_1 - y_1|^{1/2} + |x_2 - y_2|^{1/3}.$$

Carry out tests to verify whether this function satisfies the triangle inequality.

```
% FILE  d103.m  begins.

d2=inline('abs(real(x)-real(y)).^(1/2)+abs(imag(x)-imag(y)).^(1/3)','x','y');

x=10*rand(1,100)-5+10*i*rand(1,100)-5*i;
y=10*rand(1,100)-5+10*i*rand(1,100)-5*i;
z=10*rand(1,100)-5+10*i*rand(1,100)-5*i;

min(d2(x,z)+d2(z,y)-d2(x,y))

% FILE  d103.m  ends.

function y=fitbrf(lam,x)

global xdata;
global ydata;
x1=min(xdata); x2=max(xdata);
xbrp=x1+(x2-x1)*(abs(lam(1)) /(1+abs(lam(1))));
                              % This forces the break point
```

```
                                % between x1 and x2
ybrp=lam(2);
xx= xdata(xdata<=xbrp);
yy= ydata(xdata<=xbrp);
k1=sum((xx-xbrp).*(yy-ybrp));
k1=k1/sum((xx-xbrp).^2);
xx1=x(x<=xbrp);
yy1=ybrp+k1*(xx1-xbrp);
xx= xdata(xdata>=xbrp);
yy= ydata(xdata>=xbrp);
k2=sum((xx-xbrp).*(yy-ybrp));
k2=k2/sum((xx-xbrp).^2);
xx2=x(x>xbrp);
yy2=ybrp+k2*(xx2-xbrp);
y=[yy1 yy2];
% FILE  fitbrf.m  ends.



function y=fitbrobj(lambda)
global xdata;
global ydata;
y=norm(fitbrf(lambda,xdata)-ydata);
% FILE  fitbrobj.m  ends.
```

Output:

```
ans =

    0.5097
```

4. We consider three methods of fitting a second degree polynomial to $\sin(\pi * x)$ on $(0, 1)$.

(a) The first method is to use the program parfit. Let xdata=0:0.1:1; ydata=sin(pi*xdata); and carry out the fitting.

(b) The second method is to use the LSQ-fitting in the sense of the L^2 norm. In other words, we wish to find c_1, c_2, c_3 so that

$$f(c) = \int_0^1 (\sin(\pi * x) - c_1 x^2 - c_2 x - c_3)^2 dx$$

will be minimal. Set up the normal equations and write the resulting 3×3 linear system of equations. Use e626.m to implement this idea.

(c) The third method is to use the built-in function polyfit for fitting second degree polynomial.

Solution:

```
% FILE  d104.m  begins.
% MME03
% Fits a nonlinear model depending on parameter vector lam
% to xdata, ydata
% Define fmodel and fobj as inline functions:
path(path,'../../util')
close all
fmodel=inline('lam(1)* x.^2 + lam(2)*x+ lam(3)','x','lam');
fobj=inline('norm(feval(fmodel,x,lam)-y)','lam','fmodel','x','y');
%Generate some data:
xdata= 0:0.1:1.0;  ydata=sin(pi*xdata);
% Initial guess for lambda
lam0=[-1 1 1];
% Initial value of the object function (before minimization)
y0=fobj(lam0,fmodel,xdata,ydata);

% lam is the fitted value for the parameter vector
lam=fminsearch(fobj,lam0,[],fmodel,xdata,ydata);
% Compute values of the fitted function for plotting
x=0:0.05:1.0;
yfit=fmodel( x,lam);
% Final value of the object function (after minimization)
yfinal=fobj(lam,fmodel,xdata,ydata);
clf;
axes('FontSize',[16],'FontWeight','bold'); hold on;
txt=['Object function values: before =' ];
txt=[ txt num2str(y0) ', after = ' num2str(yfinal)];
title(txt);
plt=plot(x,yfit,xdata,ydata,'k.','MarkerSize',30);  grid;
text(0.1,0.1,['lam= ' mat2str(lam,4)],...
     'FontWeight','bold','FontSize',[20]);
text(0.1,1.1, ['lam(1)* x.^2 + lam(2)*x+ lam(3)'],...
     'FontWeight','bold','FontSize',[20]);
xlabel('Fitting with parfit',...
       'FontWeight','bold','FontSize',[20]);
set(plt,'LineWidth',2.5);
fprintf('parfit gives c= ');
fprintf('%10.4f',lam); fprintf('\n')
widemarg(gcf)
```

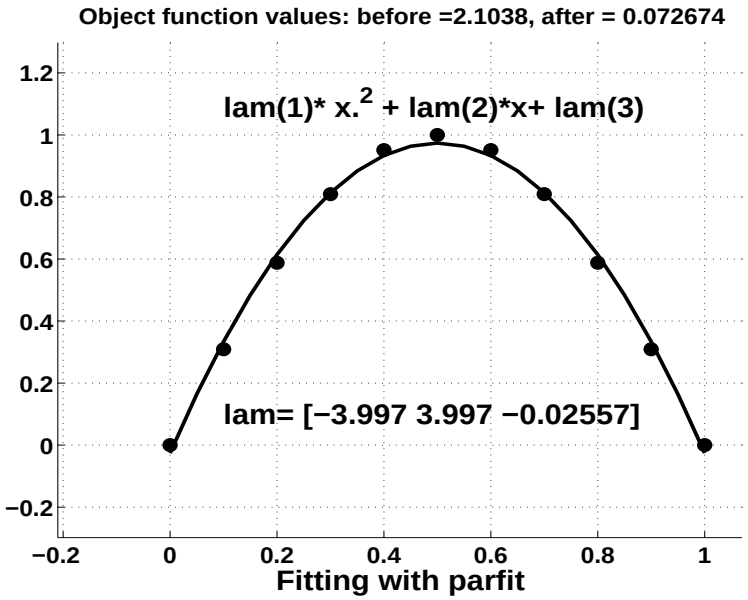
```
% FILE  d104.m  ends.


% d104b.m
% FILE  e626.m  begins.
% Let f(c1,...,cn) =
%    int_{r1}^{ r2}{ [exp(x) - sum_{k= 1}^{n}(c_k x^{n-k}) ]^2 dx}
% To minimize f find c1,...,cn such that df/dcj = 0,
% or equivalently -2 int_{r1}^{ r2}{exp(x) x^{n-j} dx} +
%      2 int_{r1}^{ r2}{sum_{k= 1}^{ n} (c_k x^{n-k}) x^{n-j} dx }
%    = -2 int_{r1}^{r2}{exp(x) x^{n-j} dx} +
%      2 sum_{k= 1}^n ((c_k /(2*(n)-k-j+1))((r2)^{2*(n)-k-j+1}-
%      (r1)^{2*(n)-k-j+1} ) )} = 0
close all
disp('L^2 fitting with the method of e626.m')
figure(1)
hold on
nmax =4;
data =zeros(nmax,nmax);
%r1 = input('Enter r1 : ');
%r2 = input('Enter r2 > r1 : ');
r1=0;  r2= 1;
for n = 2:nmax;
clear a
clear b
clear c
clear y
for j = 1:n
    for k =1:n
        a(j,k) = ((r2).^(2*(n)-k-j+1) -(r1).^(2*(n)-k-j+1))/...
                 (2*(n)-k-j+1);
    end;

% The coefficients of c_j are entries of a_{i,j}
xx=r1:0.001:r2;
yy=sin(pi*xx).*(xx.^(n-j));
% Use trapezoidal rule to compute b_j :
b(j,1)=0.001*(sum(yy)-0.5*(yy(1)+yy(length(yy))));
end;
c = a\b;
data(n,1:n) =c';
fprintf('n= %d, c=',n);
disp(c');
h=  (r2-r1)/40;
```

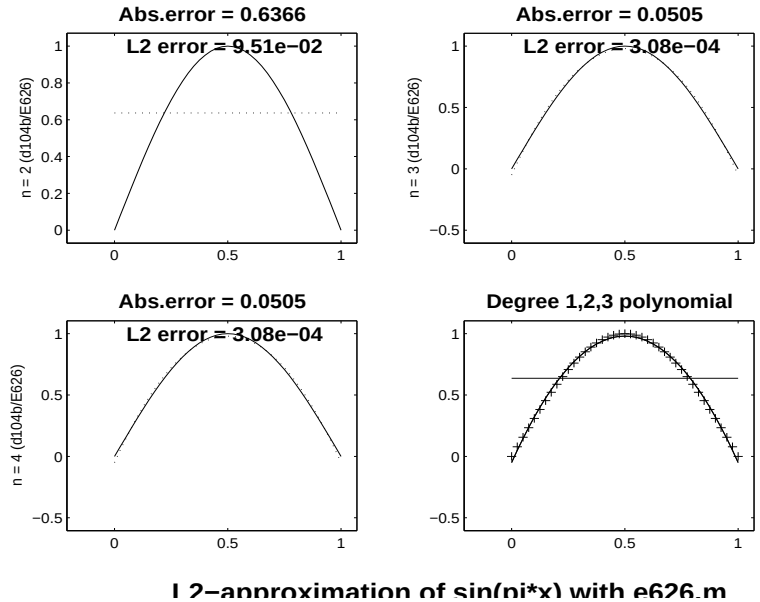
```
x = r1:h:r2;
y1 = sin(pi*x);
y2 = polyval(c,x);
erhe = abs(y2-y1);
le = length(erhe);
maxer = max(erhe);
% Use trapezoidal rule to compute error:
int1 = h*(sum(erhe)-0.5*(erhe(1)+erhe(le)));
int2 = h*(sum(erhe.^2)-0.5*(erhe(1)^2+erhe(le)^2));
txt =['Abs.error = ' num2str(maxer,'%7.4f') ];
txt2 =['  L2 error = ' num2str(int2,'%10.2e') ];
subplot(2,2,n-1);
plot(x,y1,x,y2,'k:');
title(txt, 'FontWeight','bold','FontSize',[14]);
text(min(x),max(y1),txt2,...
     'FontWeight','bold','FontSize',[14]);
widemarg(gcf)
ylabel(['n = ' num2str(n) ' (d104b/E626)']);
end;
x = r1: h :r2;
for j =2:nmax
   y(j,:) = polyval(data(j,1:j),x);
end;
subplot(2,2,4);
for j =2:nmax
   plot(x,y(j,:));
   hold on
end;
subplot(2,2,4);
title('Degree 1,2,3 polynomial',...
'FontWeight','bold','FontSize',[14]);
plot(x,sin(pi*x),'k+');
widemarg(gcf)
figure(1)
text(-1.5, -1.1,'L2-approximation of sin(pi*x) with e626.m',...
       'FontWeight','bold','FontSize',[18]);
% FILE  e626.m  ends.
% FILE  d104b.m  ends.

% FILE  d104c.m  begins.
path(path,'../../util')
close all
%Generate some data:
xdata= 0:0.1:1.0;  ydata=sin(pi*xdata);
```

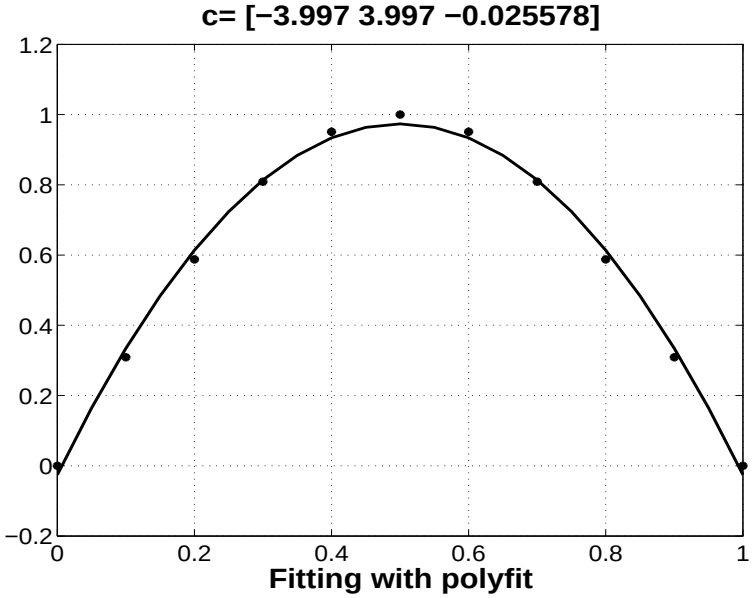
```
coef=polyfit(xdata,ydata,2);
xi=0:0.05:1.0; yi=polyval(coef,xi);
figure
axes('FontSize',[16],'FontWeight','bold');
plot(xi,yi,xdata,ydata,'*','LineWidth',2)
grid on
title(['c= ' mat2str(coef,5)],...
        'FontWeight','bold','FontSize',[20]);
xlabel('Fitting with polyfit', ...
         'FontWeight','bold','FontSize',[20]);
disp('Fitting with polyval gives c=')
disp(coef)
% FILE  d104c.m  ends.
```

Output:

```
Warning: Name is nonexistent or not a directory: ../../util.
> In /usr/local/matlab/toolbox/matlab/general/path.m at line 116
  In /home/vuorinen/mme08/demo08/d10/d104.m at line 6
  In /home/vuorinen/mme08/demo08/d10/d10all.m at line 35
parfit gives c=    -3.9970    3.9970   -0.0256
L^2 fitting with the method of e626.m
n= 2, c=   -0.0000    0.6366

n= 3, c=   -4.1225    4.1225   -0.0505

n= 4, c=   -0.0000   -4.1225    4.1225   -0.0505

Warning: Name is nonexistent or not a directory: ../../util.
> In /usr/local/matlab/toolbox/matlab/general/path.m at line 116
  In /home/vuorinen/mme08/demo08/d10/d104c.m at line 2
  In /home/vuorinen/mme08/demo08/d10/d10all.m at line 39
Fitting with polyval gives c=
   -3.9970    3.9970   -0.0256
```


Object function values: before =2.1038, after = 0.072674
lam(1)* x.^2 + lam(2)*x+ lam(3)
lam= [−3.997 3.997 −0.02557]
Fitting with parfit


Abs.error = 0.6366
L2 error = 9.51e−02
n = 2 (d104b/E626)
Abs.error = 0.0505
L2 error = 3.08e−04
n = 3 (d104b/E626)
Abs.error = 0.0505
L2 error = 3.08e−04
n = 4 (d104b/E626)
Degree 1,2,3 polynomial
L2−approximation of sin(pi*x) with e626.m



5. The built-in function \ of MATLAB solves also systems of linear equations in the LSQ sense. For instance, if we have more linear equations than unknowns, we do not expect to have "an exact solution in the sense of linear algebra" but we still might have an LSQ-solution. Use this fact in the following two cases:

(a) Plot the lines $y = 0.2 - 0.5 * x$, $y = 0.8 - 2 * x$, $y = 0.2 + 2 * x$, $y = -0.3 + 1.5 * x$, $y = 0.0 + 0.95 * x$, Write these equations in the form $a[x; y] = b$ where a is a 5×2 matrix and solve the system in the LSQ sense for $[x; y]$ and plot this LSQ solution of the system in the same figure.

(b) The file d105.dat on the www-page contains 8 temperature measurements in the format $(x_i, y_i, g_i), i = 1, ..., 8$. We wish to fit the quadratic function

$$g(x, y) = ax^2 + by^2 + cxy + dx + ey + f$$

to this data. Write the corresponding linear system of equations and solve it for the unknown vector (a, b, c, d, e, f) . Compute the value of the temperature at the location $(5, 5)$. For each data point (x_i, y_i) , compute the difference $g_i - g(x_i, y_i)$. Graph the surface $z = g(x, y)$.

Solution:

```
% FILE d105.m begins.
close all
% y= 0.2-0.5*x;
```

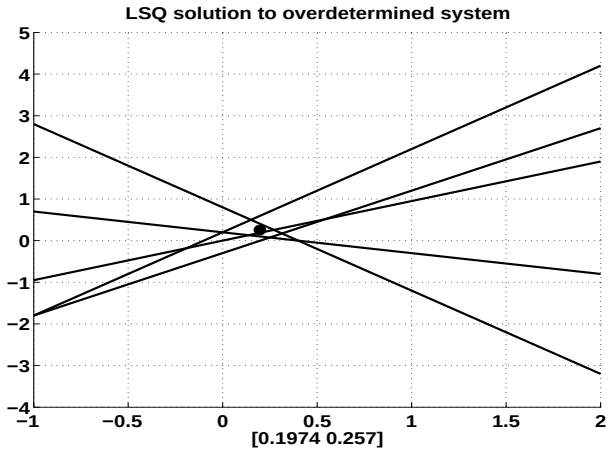
```
a(1,1)=0.5; b(1)=0.2;
% y= 0.8-2*x;
a(2,1)=2; b(2)=0.8;
% y= 0.2+2*x;
a(3,1)=-2; b(3)=0.2;
% y= -0.3+1.5*x;
a(4,1)=-1.5; b(4)=-0.3;
% y= 0.0+0.95*x;
a(5,1)=-0.95; b(5)=0.0;
%a(5,1)=a(4,1); b(5)=b(4);
a(:,2)=ones(5,1);
t=[-1 2];
figure(1)
axes('FontSize',[16],'FontWeight','bold');
hold on
for j=1:5
  plot(t,b(j)-a(j,1)*t,'LineWidth',2)
end
x=a\(b');
disp('LSQ solution to overdetermined system is:')
fprintf('%8.4f',x')
plot(x(1), x(2),'.','MarkerSize', 30)
grid on
title('LSQ solution to overdetermined system',...
      'FontSize',[16],'FontWeight','bold');
xlabel(mat2str(x',4),...
       'FontSize',[16],'FontWeight','bold');
print -dps d105a.ps

% Part (b):
g=inline('a*x*x + b*y*y + c*x*y + d*x + e*y +f',...
 'a','b','c','d','e','f','x','y');
% ax*x + b*y*y + c*x*y + d*x + e*y +f = g(x,y)
m=load('d105.dat');  % temperature measurements
[d1,d2]=size(m);
coef=[m(:,1).^2  m(:,2).^2  m(:,1).*m(:,2)  m(:,1)  m(:,2) ones(d1,1)];
rhs=m(:,3);
v=coef\rhs;
fprintf('\nCoefficients a b c d e f:\n')
disp(v')
a=v(1);
b=v(2);
c=v(3);
```

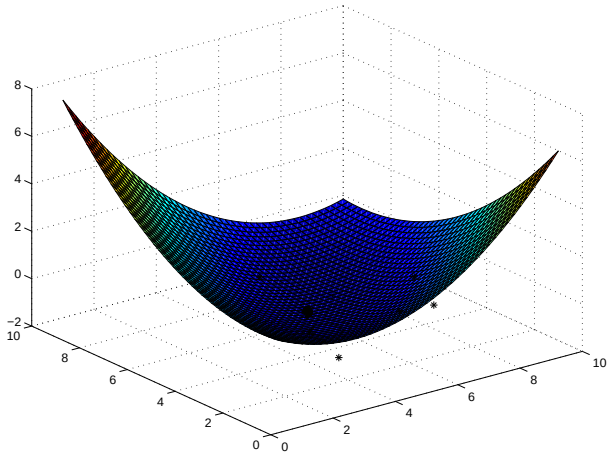
```
d=v(4);
e=v(5);
f=v(6);
t=g(a,b,c,d,e,f,5,5);
disp('Temperature at (5,5):')
disp(t)
x1=1; x2=10; dx=0.2;
y1=1; y2=10; dy=0.2;
[xx,yy]=meshgrid(x1:dx:x2,y1:dy:y2);
px=xx(:);  py=yy(:);
pz=[];
for j=1:length(px)
  pz=[pz g(a,b,c,d,e,f,px(j),py(j))];
end
zz= reshape(pz,size(xx));
figure(2)
surf(x1:dx:x2,y1:dy:y2, zz)
hold on
plot3(m(:,1),m(:,2),m(:,3),'*')
plot3(5,5,t,'k.','MarkerSize',30)
fprintf('   x        y     t       t-g(x,y)\n')
for j=1:d1
  tt=g(a,b,c,d,e,f,m(j,1), m(j,2));
  fprintf('% 6.2f % 6.2f % 6.2f  % 6.2f \n',m(j,1), m(j,1), m(j,2), ...
  m(j,3)-tt);
end
print -dps d105b.ps
% FILE d105.m ends.

4.6       0.6       1.3
6.0       1.0       0.9
7.6       3.9       0.2
6.6       6.1       -1.2
5.1       7.1       -0.4
3.7       5.7       -0.4
2.4       1.5       0.8
2.8       0.8       -0.1
```

Output:

```
LSQ solution to overdetermined system is:
  0.1974  0.2570
Coefficients a b c d e f:
```

```
    0.1328    0.0904   -0.1501   -0.7746   -0.1772    2.0552

Temperature at (5,5):
   -0.8783

   x        y     t       t-g(x,y)
  4.60   4.60   0.60     0.49
  6.00   6.00   1.00    -0.30
  7.60   7.60   3.90     0.13
  6.60   6.60   6.10    -0.16
  5.10   5.10   7.10     0.18
  3.70   3.70   5.70    -0.17
  2.40   2.40   1.50     0.44
  2.80   2.80   0.80    -0.61
```





6. Let A, B be $m \times n$ and $p \times q$ matrices. Their Kronecker product $kron(A, B)$ (also called tensor or direct product) is the $mp \times nq$ matrix

```
[a(1,1)B ... a(1,n)B; ...; a(m,1)B ... a(m,n)B ];
```

For simplicity we write here $Kron(A,B) = [A,B]$. Verify experimentally the following properties with the help of MATLAB's built-in function kron and also state possible restrictions for the dimensions of the involved matrices:

(a) $[A,[B,C]] = [[A,B],C]$,
(b) $[A,B+C] = [A,B] + [A,C]$,
(c) $[A^{-1},B^{-1}] = [A,B]^{-1}$
(d) $[A^T,B^T] = [A,B]^T$
(e) If B and C have SVDs $B = U_1W_1V_1^T$ $C = U_2W_2V_2^T$, then $[B,C] = [U_1,U_2][W_1,W_2]([V_1,V_2]^T)$. In the study of so called MIMO channels in digital communication the Kronecker product has become very important. For instance, a Google search with the key words MIMO + "Kronecker product" +SVD will give you some idea of the applications of Kronecker products and SVD to the future generations of mobile phones.

Solution:

```
% FILE d106.m begins.
d=2+fix(5*rand(1,6));
a=rand(d(1), d(2));
b=rand(d(3), d(4));
c=rand(d(5), d(6));
bc=kron(b,c); lhs=kron(a,bc);
ab=kron(a,b); rhs=kron(ab,c);
t=norm(lhs-rhs);
% (a)  $[A,[B,C]]=[[A,B],C],$
fprintf('\n(a): Error = %12.3e\n',t);
c2=rand(size(b));
bpc=b+c2; lhs=kron(a,bpc);
rhs=kron(a,b)+ kron(a,c2);
t=norm(lhs-rhs);
% $$$ (b)  $[A,B+C]= [A,B]+ [A,C],$
fprintf('\n(b): Error = %12.3e\n',t);
a2=rand(d(1), d(1));
b2=rand(d(3), d(3));
ai=inv(a2);bi=inv(b2);
lhs=kron(ai,bi); rhs= inv(kron(a2,b2));
t=norm(lhs-rhs);
% $$$ (c) $ [A^{-1},B^{-1}]= [A,B]^{-1} \, $
fprintf('\n(c): Error = %12.3e\n',t);
at=a'; bt=b';
```

```
lhs=kron(at,bt); rhs= (kron(a,b)');
t=norm(lhs-rhs);
% $$$ (d) $ [A^T,B^T]= [A,B]^T \, $
fprintf('\n(d): Error = %12.3e\n',t);
% $$$ (e) If $B$ and $C$ have SVDs $B = U_1 W_1 V_1^T$
% $$$      $ C= U_2 W_2 V_2^T\, , $ then
% $$$ $ [B,C]= [U_1,U_2][W_1, W_2]  ( [V_1, V_2]^T)\,.$
[u1, w1, v1]=svd(b);
[u2, w2, v2]=svd(c);
lhs=kron(b,c);
tmp=kron(u1, u2);
tmp=tmp*kron(w1, w2);
rhs=tmp*(kron(v1,v2)');
t=norm(lhs-rhs);
% $$$ (e) $ [A^T,B^T]= [A,B]^T \, $
fprintf('\n(e): Error = %12.3e\n',t);
a=diag(1:3);
a(1,3)=4;
b=eye(3);
disp('a=')
disp(a)
disp('b=')
disp(b)
disp('kron(a,b)=')
disp(kron(a,b))
% FILE d106.m ends.
```

Output:

```
(a): Error =    2.916e-16

(b): Error =    4.518e-16

(c): Error =    3.712e-12

(d): Error =    0.000e+00

(e): Error =    1.736e-15
a=
     1     0     4
     0     2     0
     0     0     3
```

```
b=
     1     0     0
     0     1     0
     0     0     1

kron(a,b)=
     1     0     0     0     0     0     4     0     0
     0     1     0     0     0     0     0     4     0
     0     0     1     0     0     0     0     0     4
     0     0     0     2     0     0     0     0     0
     0     0     0     0     2     0     0     0     0
     0     0     0     0     0     2     0     0     0
     0     0     0     0     0     0     3     0     0
     0     0     0     0     0     0     0     3     0
     0     0     0     0     0     0     0     0     3
```

Solutions in Python

Problem 1.

```
# FILE: d101.py begins
# Create some artificial data

from scipy import *
from Numeric import *
from random import *

x=arange(-2,4.01,0.1)
y=0.2*sin(3.*x);
for j in range(len(y)):
   if x[j]<1.0: y[j]=y[j]+0.5*(x[j]-1.)
   else: y[j]=y[j]+2.*(x[j]-1.)
count=1
while count<=3:
   gplt.plot(x,y,'w p ps 3')
   gplt.hold('on')
   s,t= -1.+4.*random(),-1.+4.*random()
   print 's= %12.6f , t= %12.6f' % (s,t)
   x1,x2,y1,y2=list(),list(),list(),list()
```

```
   for j in range(len(x)):
      if x[j]<s:
         x1.append(x[j])
         y1.append(y[j])
      else:
         x2.append(x[j])
         y2.append(y[j])
   x1,x2=array(x1),array(x2)
   y1,y2=array(y1),array(y2)
   # Use problem d021 formulas for k, b:
   denom1=sum((x1-s)*(x1-s))
   denom2=sum((x2-s)*(x2-s))
   numer1=sum((x1-s)*(y1-t))
   numer2=sum((x2-s)*(y2-t))
   print numer1
   k1=numer1/denom1
   k2=numer2/denom2
   b1=(t-k1*s);
   b2=(t-k2*s);
   yy1=k1*x1+b1;
   yy2=k2*x2+b2;
   gplt.plot(x1,yy1,x2,yy2)
   print 'Coefficients k1, b1 %12.6f  %12.6f'%(k1,b1)
   print 'Coefficients k2, b2 %12.6f  %12.6f'%(k2,b2)
   s1=sum((y1-yy1)**2);
   s2=sum((y2-yy2)**2);
   print 'Sum of squares= %12.6f'%(s1+s2)
   gplt.hold('off')
# FILE d101.m ends.
```

Output:

```
s=     2.295729 , t=     2.640253
316.68555732
Coefficients k1, b1     1.157762     -0.017654
Coefficients k2, b2     1.886523     -1.690692
Sum of squares=    19.120628
s=    -0.249433 , t=     0.577880
27.8417852254
Coefficients k1, b1     1.431942      0.935054
Coefficients k2, b2     0.935106      0.811127
Sum of squares=    61.329586
```

```
s=     0.551324 , t=    -0.153901
30.1763819218
Coefficients k1, b1     0.514476     -0.437544
Coefficients k2, b2     1.665580     -1.072175
Sum of squares=     4.582122
s=     1.512895 , t=     0.089211
61.8109754545
Coefficients k1, b1     0.410075     -0.531190
Coefficients k2, b2     2.527677     -3.734898
Sum of squares=     8.132213
s=     2.600401 , t=     0.343461
105.117853856
Coefficients k1, b1     0.313601     -0.472026
Coefficients k2, b2     4.838393    -12.238301
Sum of squares=    81.987683
s=     1.631730 , t=    -0.370185
33.3511720033
Coefficients k1, b1     0.200520     -0.697379
Coefficients k2, b2     2.985005     -5.240908
Sum of squares=    26.615637
s=     0.192900 , t=     1.398077
62.7007953062
Coefficients k1, b1     1.667937      1.076332
Coefficients k2, b2     0.821832      1.239545
Sum of squares=    83.427602
...
```

Problem 2.

```
# FILE d102.py begins.
from mmeutil import *
from Numeric import *
from scipy 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

# Get point inside polygon
def getp(x,y):
    n = len(x) - 1
    p = ranvec(2)
    while(not inpoly(n, x, y, p[0], p[1])):
          p = ranvec(2)
    return p

# Return polygon area
def polyarea(x,y):
    formarea = 0
    for i in range(1,len(x)):
        formarea = formarea + 0.5*(x[i-1] * y[i] -x[i] * y[i-1])
    return abs(formarea)

xp = array([0, 0.5, 1, 0])
yp = array([0, 1, 0.5, 0])
origarea = polyarea(xp, yp)
ratio = list()

while (len(ratio) < 1000):
    a,b,c = getp(xp,yp),getp(xp,yp),getp(xp,yp)
    xx = array([ a[0], b[0], c[0], a[0] ])
    yy = array([ a[1], b[1], c[1], a[1] ])
    ratio.append(polyarea(xx,yy)/origarea)
    if (len(ratio) < 10):
        gplt.plot(xp, yp, xx, yy)

print "ratio (should be 1):"
print 12.0*mean(ratio)

# FILE d102.py ends.
```

Output:

```
ratio (should be 1):
0.991878953737
```

Problem 3.

```
# FILE  d103.py begins.
from mmeutil import *
from Numeric import *

i=complex(0.0,1.0)

def d2(xx,yy):
  zz=1.0*arange(len(xx))
  for j in range(len(xx)):
    x,y=xx[j],yy[j]
    zz[j]=pow(abs(x.real-y.real),0.5)+pow(abs(x.imag-y.imag),1.0/3.0)
  return zz

x=10.0*ranvec(100)-5+10.0*i*ranvec(100)-5.0*i
y=10.0*ranvec(100)-5+10.0*i*ranvec(100)-5.0*i
z=10.0*ranvec(100)-5+10.0*i*ranvec(100)-5.0*i

print min(d2(x,z)+d2(z,y)-d2(x,y))

# FILE d103.py ends
```

Output:

```
0.745260904133
```

Problem 4.

```
# FILE  d104.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 distribution
from Numeric import *
from scipy import *

global xdata
global ydata

def fmodel(lam,x):
```

```
   return lam[0]*x*x+lam[1]*x+lam[2]

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

x=arange(0,1,0.1)
y=sin(pi*x)
xdata=x
ydata=y
lam0=array([1.0,1.0,8.0])  # Initial guess for lambda
y0=fobj(lam0)              # Initial value of object function
lam=fmin(fobj,lam0)
showvec2(lam)
                      # lam is the fitted value for
                      # the parameter vector
x=0.01*arange(0.,100.)
yfit=fmodel(lam,x)
yfinal=fobj(lam)      # Final value of the object function
gplt.plot(x,yfit,'notitle w l')
gplt.hold('on')
gplt.plot(xdata,ydata,'notitle w p')
# FILE d104.py ends
```

Output:

```
Optimization terminated successfully.
         Current function value: 0.061010
         Iterations: 142
         Function evaluations: 254
  -4.104
   4.076
  -0.033
```

Problem 5.

```
# FILE d105.py begins.
from mmeutil import *
from scipy import *
from Numeric import *
from LinearAlgebra import *
a=0.0*matrix(5,2)+1.0
```

```
b=ranvec(5)
# y= 0.2-0.5*x
a[0,0]=0.5
b[0]=0.2
# y= 0.8-2*x
a[1,0]=2.0
b[1]=0.8
# y= 0.2+2*x
a[2,0]=-2.0
b[2]=0.2
# y= -0.3+1.5*x
a[3,0]=-1.5
b[3]=-0.3
# y= 0.0+0.95*x
a[4,0]=-0.95
b[1]=0.0
a[4,0]=a[3,0]
b[4]=b[3]
t=array([-1.0,2.0])
for j in range(5):
  gplt.plot(t,b[j]-a[j,0]*t)
  gplt.hold('on')
x=SVDsolve(a,b)
print 'LSQ solution to overdetermined system is:'
print x
gplt.plot([x[0]], [x[1]])

# part (b)

def g(a,b,c,d,e,f,x,y):
   return a*x*x + b*y*y + c*x*y + d*x + e*y +f
# ax*x + b*y*y + c*x*y + d*x + e*y +f = g(x,y)

m=getmat2('d105.dat');  # temperature measurements
d1,d2=m.shape
coef=[m[:,0]**2.0,m[:,1]**2.0,m[:,0]*m[:,1],m[:,0],m[:,1],0.0*ranvec(d1)+1.0]
rhs=m[:,2]
ic=generalized_inverse(transpose(coef))
v=matrixmultiply(ic,rhs)
print '\nCoefficients a b c d e f:'
print v
```

```
a,b,c,d,e,f=list(v)
t=g(a,b,c,d,e,f,5,5);
print 'Temperature at (5,5):'
print t
x1=1.0
x2=10.0
dx=0.2
y1=1.0
y2=10.0
dy=0.2
print '   x        y     t       t-g(x,y)'
for j in range(d1):
   tt=g(a,b,c,d,e,f,m[j,0], m[j,1]);
   print '% 6.2f % 6.2f % 6.2f %6.2f'%(m[j,0],m[j,0],m[j,1],m[j,2]-tt)
# FILE d105.py ends.
```

Output:

```
LSQ solution to overdetermined system is:
[ 0.04347826 -0.01826087]

Coefficients a b c d e f:
[ 0.13277294  0.09037936 -0.15013263 -0.77459878 -0.17720623  2.05521825]
Temperature at (5,5):
-0.878315121595
   x        y     t       t-g(x,y)
  4.60   4.60   0.60   0.49
  6.00   6.00   1.00  -0.30
  7.60   7.60   3.90   0.13
  6.60   6.60   6.10  -0.16
  5.10   5.10   7.10   0.18
  3.70   3.70   5.70  -0.17
  2.40   2.40   1.50   0.44
  2.80   2.80   0.80  -0.61
```

Problem 6.

```
# FILE d106.py begins
from mmeutil import *
from scipy import *
from Numeric import *
from LinearAlgebra import *
```

```
def kron(a,b):
   if not a.iscontiguous():
      a = reshape(a, a.shape)
   if not b.iscontiguous():
      b = reshape(b, b.shape)
   o = outerproduct(a,b)
   o.shape = a.shape + b.shape
   return concatenate(concatenate(o, axis=1), axis=1)

d=2.0+(5.0*ranvec(6))
a=ranmat(int(d[0]),int(d[1]))
b=ranmat(int(d[2]),int(d[3]))
c=ranmat(int(d[4]),int(d[5]))

bc=kron(b,c)
lhs=kron(a,bc)
ab=kron(a,b)
rhs=kron(ab,c);
t=mnormp(lhs-rhs)
# (a)  $[A,[B,C]]=[[A,B],C],$
print '\n(a): Error = %12.3e'%(t)
c2=ranmat(b.shape[0],b.shape[1])
bpc=b+c2
lhs=kron(a,bpc)
rhs=kron(a,b)+ kron(a,c2)
t=mnormp(lhs-rhs)
# $$$ (b)  $[A,B+C]= [A,B]+ [A,C],$
print '\n(b): Error = %12.3e'%(t)
a2=ranmat(int(d[0]), int(d[0]))
b2=ranmat(int(d[2]), int(d[2]))
ai=generalized_inverse(a2)
bi=generalized_inverse(b2)
lhs=kron(ai,bi)
rhs= generalized_inverse(kron(a2,b2))
t=mnormp(lhs-rhs)
# $$$ (c) $ [A^{-1},B^{-1}]= [A,B]^{-1} \,
print '\n(c): Error = %12.3e ' %(t)
at=transpose(a)
bt=transpose(b)
lhs=kron(at,bt)
```

```
rhs= transpose(kron(a,b))
t=mnormp(lhs-rhs)
# $$$ (d) $ [A^T,B^T]= [A,B]^T \, $
print '\n(d): Error = %12.3e' %(t)
# $$$ (e) If $B$ and $C$ have SVDs $B = U_1 W_1 V_1^T$
# $$$      $ C= U_2 W_2 V_2^T\, , $ then
# $$$ $ [B,C]= [U_1,U_2][W_1, W_2]  ( [V_1, V_2]^T)\,.$
[u1, w1, v1]=singular_value_decomposition(b)
[u2, w2, v2]=singular_value_decomposition(c)
lhs=kron(b,c)
tmp=kron(u1, u2)
tmp=matrixmultiply(tmp,kron(diag(w1), diag(w2)))
rhs=matrixmultiply(tmp,kron(v1,v2))
t=mnormp(lhs-rhs)
print '\n(e): Error = %12.3e'%(t)
a=diag(arange(1.,4.))
a[0,2]=4.0
b=1.0*eye(3)
print 'a='
print a
print 'b='
print b
print 'kron(a,b)='
print kron(a,b)
# FILE d106.py ends.
```

Output:

```
(a): Error =    9.184e-16

(b): Error =    8.041e-16

(c): Error =    1.113e-13

(d): Error =    0.000e+00

(e): Error =    9.473e-16
a=
[[ 1.  0.  4.]
 [ 0.  2.  0.]
 [ 0.  0.  3.]]
```

```
b=
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
kron(a,b)=
[[ 1.  0.  0.  0.  0.  0.  4.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  4.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  4.]
 [ 0.  0.  0.  2.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  2.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  2.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  3.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  3.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  3.]]
```