University of Helsinki / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 12 / Solutions

1. Show experimentally that for real 2×2 matrices `A=[a b; c d]` the following equality holds;

$$cond(A) = s + \sqrt{s^2 - 1} \quad \text{where} \quad s = (a^2 + b^2 + c^2 + d^2)/(2|\det(A)|).$$

Solution:

```
% FILE: d121.m begins
% Forsythe-Moler p. 25 says that if A=[a b; c d] satisfies
% det(A)=ad-bc not zero and s= (a*a+b*b+c*c+d*d)/(2*det(A))
% then cond(A)= s + sqrt(s*s-1)
n=2;
fprintf('d121:  Error\n')
for p=1:15
  A=3*(rand(n,n)-0.5);
  t= norm(A(:)')^2;
  s=t/(2*abs(det(A)));
  tmp=cond(A)-s-sqrt(abs(s*s-1));
  fprintf('% 12.2e\n',tmp)
end
% FILE: d121.m ends
```

Output:

```
d121:  Error
    0.00e+00
    5.55e-16
   -3.61e-16
   -2.66e-15
    6.66e-16
    5.00e-16
    6.66e-16
   -5.55e-17
    0.00e+00
   -1.11e-15
   -4.44e-15
    2.22e-15
   -9.58e-16
   -2.66e-15
    5.48e-15
```

2. Use the data d122dat.dat to fit the model $f(\lambda_1, \lambda_2, \lambda_3, x) = \lambda_1/(1 + (x - \lambda_2)^2) + 1/(1 + (x - \lambda_3)^2)$. Use the initial values `[1,-1,2]` as a guess. Hint: `parfit`.

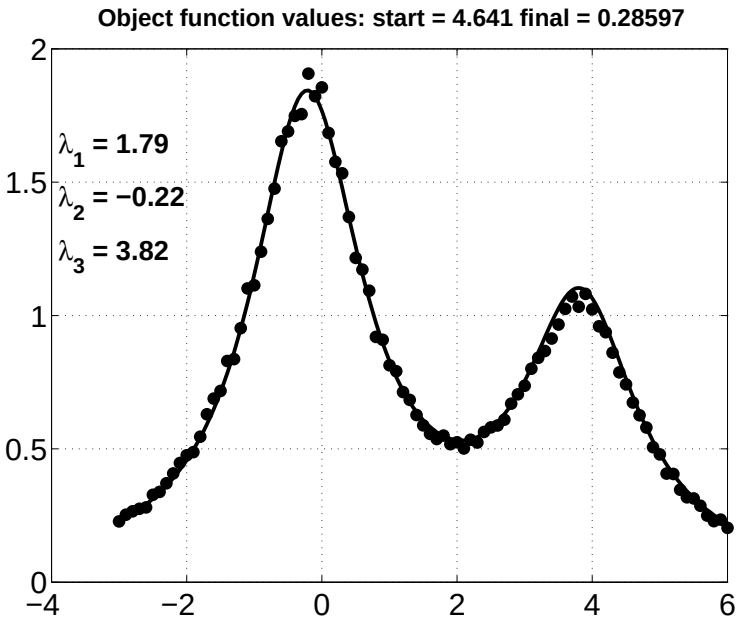
Solution:

```
% FILE d122.m begins.
% Uses modified parfit, see d083.m and d062rajala.m
fmodel =...
 inline('lam(1)./(1+(x-lam(2)).^2)+1./(1+(x-lam(3)).^2)','x','lam');
fobj =...
 inline('norm(feval(fmodel,x,lam) - y)', 'lam','fmodel','x','y');

load d122dat.dat ;   xdat=d122dat(:,1)';     ydat=d122dat(:,2)';
lam0=[1 -1 2];
y0=fobj(lam0,fmodel,xdat,ydat);   % Initial value of object function
lam=fminsearch(fobj,lam0,[],fmodel,xdat,ydat);
fprintf('d122:\n lam= '); fprintf('  %f',lam); fprintf('\n');
                        % lam is the fitted value for
                        % the parameter vector
x=min(xdat):0.05:max(xdat);
yfit=fmodel(x,lam);
yfinal=fobj(lam,fmodel,xdat,ydat);% Final value of the object function
figure(1)
axes('FontWeight','bold','FontSize',20);
plt=plot(x,yfit,xdat,ydat,'k.','MarkerSize',25);  grid on;
title(['Object function values: start = ' num2str(y0) ...
                              ' final = ' num2str(yfinal)],...
      'FontWeight','bold','FontSize',16);
ax=axis;
text(ax(1)+0.1,ax(4)-0.4,...
     ['\lambda_1 = ' num2str(lam(1),'%5.2f')],...
     'FontWeight','bold','FontSize',18);
text(ax(1)+0.1,ax(4)-0.6,...
     ['\lambda_2 = ' num2str(lam(2),'%5.2f')],...
     'FontWeight','bold','FontSize',18);
text(ax(1)+0.1,ax(4)-0.8,...
     ['\lambda_3 = ' num2str(lam(3),'%5.2f')],...
     'FontWeight','bold','FontSize',18);
set(plt,'LineWidth',2.5);
% FILE d122.m ends.
```

Output:

```
d122:
```

```
lam=   1.786561  -0.221072  3.822403
```



3. The program e916.m contains a small error: the exact solution is

```
% N.B. The term   c2*one./(t.^2) is now OK,
% there was an error in lecture notes!!!
exact = (1.1-c2)*t +c2*one./(t.^2) - 0.3*sin(log(t)) - 0.1*cos(log(t));
```

Correct the program and modify it so that it runs for the step size $(b-a)/n, n = 8:20$. Print the maximum error for each value of n.

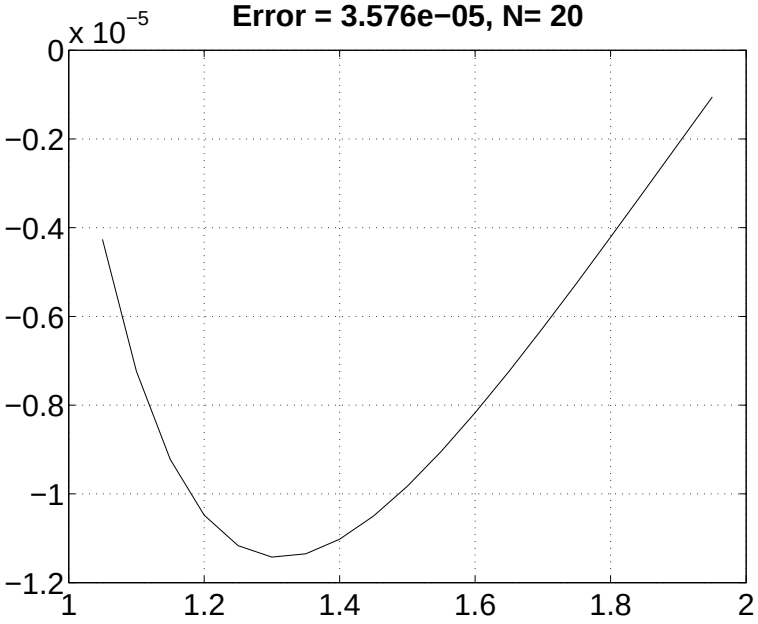
Solution:

```
% FILE  d123.m  begins.
% Corrected version of e916.m
% Solution of boundary value problem by finite difference method:
%    y(a)=bval1, y(b) = bval2, y" = p(t)y' + q(t)y + r(t)
% Put t(j) = a + j*h, j = (b-a)/(n+1), j = 0,1,..,n+1,
% FD approximation gives:
% y(t(j-1)) - 2y(t(j)) + y(t(j+1)) =
%    (h/2)(y(t(j+1)) -y(t(j-1)))p(t(j))+
%     h*h*q(t(j)) y(t(j)) + h*h*r(t(j)) or
% y(t(j-1))(1 +(h/2)p(t(j))) - (2+ h*h*q(t(j))) y(t(j))
%     + y(t(j+1))(1 -(h/2)p(t(j))) = h*h*r(t(j)), j = 1,,.n
% y(t(0)) = bval1, y(t(n+1)) = bval2
```

```
% Compare with exact solution given below
clear all;  clf;
fprintf('d123\n  n     max.error\n')
for N=8:20
a = 1; b = 2; h = (b-a)/N;  bval1 =1; bval2 = 2;
tt= a:h:b; t = tt(2:length(tt)-1);
n = length(t); one = ones(size(1:n));
% There are n interior points and n unknowns
p =-2*one./t; q = 2*one./(t.^2); r = sin(log(t))./(t.^2);
diagon = -(2*one +h*h*q);
tmp = (one- 0.5*h*p);        superd = tmp(1:n-1);
tmp = (one+ 0.5*h*(p));      subdia = tmp(2:n);
mtx = diag(diagon) + diag(superd,1) + diag(subdia,-1);
rhs =h*h*(r');
rhs(1,1) = rhs(1,1) -bval1*(1+(-2/(a+h))*0.5*h);
rhs(n,1) = rhs(n,1) -bval2*(1-(-2/(b-h))*0.5*h);
sol = mtx\rhs;
c2 = (1/70)*(8-12*sin(log(2)) - 4*cos(log(2)));
% N.B. The term   c2*one./(t.^2) is now OK,
% there was an error in lecture notes!!!
exact = (1.1-c2)*t +c2*one./(t.^2) - 0.3*sin(log(t)) - 0.1*cos(log(t));
figure(1)

plot(t, sol,'k+',a,bval1,'ko', b,bval2,'ko',t, exact),
xlabel(['Numerical (+) and exact solutions (E916), N= ' num2str(N)]),
title('Finite-difference method for boundary value problem',...
            'FontWeight','bold','FontSize',16);
grid on

figure(2)
clf
axes('FontWeight','bold','FontSize',20)
plot(t,sol-exact'),
txt=num2str(norm(sol-exact') , '%12.3e');
title(['Error = ' txt ', N= ' num2str(N)],...
            'FontWeight','bold','FontSize',20);
grid on
fprintf('%2d   %12.3e\n',N,norm(sol-exact'))
%pause(1)
end
info(:,1) = sol; info(:,2) = exact';

% FILE  d123.m  ends.
```

Output:

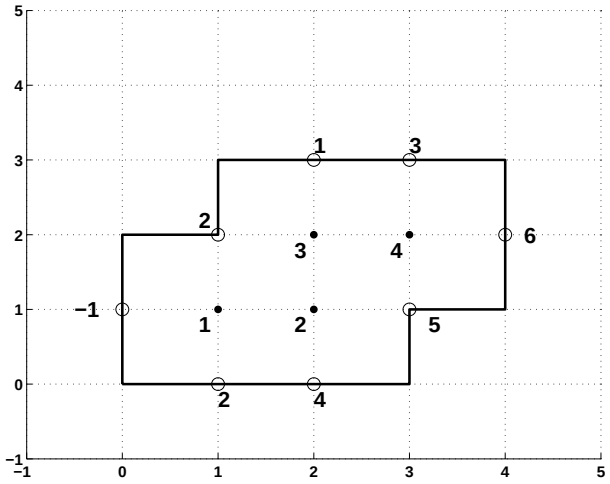
```
d123
  n     max.error
 8      1.402e-04
 9      1.177e-04
10      1.007e-04
11      8.735e-05
12      7.673e-05
13      6.809e-05
14      6.096e-05
15      5.499e-05
16      4.993e-05
17      4.560e-05
18      4.187e-05
19      3.861e-05
20      3.576e-05
```



4. Solve Dirichlet's problem

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

in the situation pictured below, by using the boundary values and the numbering of variables as in the picture. The sidelength of a square is 1.



Solution:

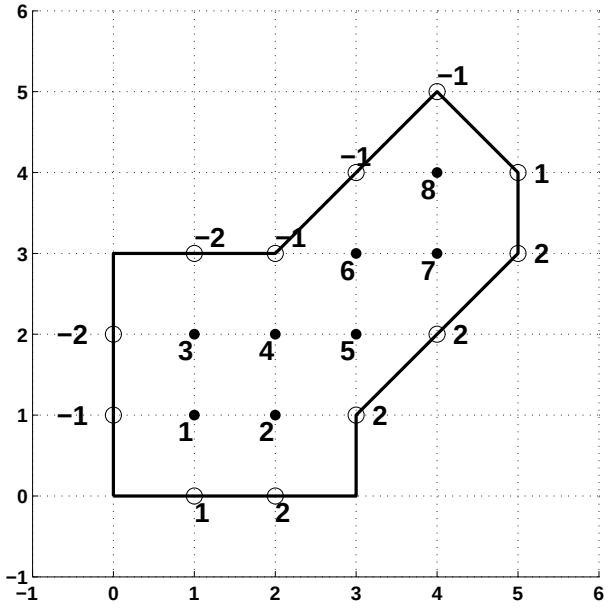
```
% FILE d124.m begins
close all;
%clear all;
clc;
%                              % b
% 4*p1 = -1 + 2 + p2 + 2;     % 3
% 4*p2 = p1 + p3 + 5 + 4;     % 9
% 4*p3 = 2 + 1 + p4 + p2;     % 3
% 4*p4 = p3 + 3 + 6 + 5;      % 14
a = [ 4 -1 0 0;
     -1 4 -1 0;
     0 -1 4 -1;
     0 0 -1 4];
b = [ 3 9 3 14];
disp([a b'])
t = a\b'
% FILE d124.m ends.
```

Output:

```
     4    -1     0     0     3
    -1     4    -1     0     9
     0    -1     4    -1     3
     0     0    -1     4    14


t =
```

```
1.5742
3.2967
2.6124
4.1531
```

5. As 4, but in the situation of the following picture.



Solution:

```
% FILE: d125.m begins
%clear all;
close all; clc;
%                              % b
% 4*p1 = -1 + p3 + p2 +1;      % 0
% 4*p2 = p1 + p4 + 2 + 2;      % 4
% 4*p3 = -2 + -2 + p4 + p1;    % -4
% 4*p4 = p3 + -1 + p5 + p2;    % -1
% 4*p5 = p4 + p6 + 2 + 2;      % 4
% 4*p6 = -1 + -1 + p7 + p5;    % -2
% 4*p7 = p6 + p8+ 2 + 2;       % 4
% 4*p8 = -1 + -1 + 1 + p7;     % -1
a = [ 4 -1 -1 0 0 0 0 0;
     -1 4 0 -1 0 0 0 0;
```

```
     -1 0 4 -1 0 0 0 0;
     0 -1 -1 4 -1 0 0 0;
     0 0 0 -1 4 -1 0 0;
     0 0 0 0 -1 4 -1 0;
     0 0 0 0 0 -1 4 -1;
     0 0 0 0 0 0 -1 4];
b = [0 4 -4 -1 4 -2 4 -1];
disp([a b'])
t = a\b'
% FILE: d125.m ends
```

Output:

```
     4    -1    -1     0     0     0     0     0     0
    -1     4     0    -1     0     0     0     0     4
    -1     0     4    -1     0     0     0     0    -4
     0    -1    -1     4    -1     0     0     0    -1
     0     0     0    -1     4    -1     0     0     4
     0     0     0     0    -1     4    -1     0    -2
     0     0     0     0     0    -1     4    -1     4
     0     0     0     0     0     0    -1     4    -1
```

```
t =

   -0.0000
    1.0000
   -1.0000
   -0.0000
    1.0000
   -0.0000
    1.0000
         0
```

6. Consider the data $(x_j, y_j), j = 1, ..., m$, and set

$$f(a,b,c,d,x) = ax^2 + bx + c + d/x\,, \quad S = \sum_{j=1}^{m} (y_j - f(a,b,c,d,x_j))^2\,.$$

A researcher is modelling the political awareness in EU countries using this model.

(a) Help the researcher to set up the normal equations. (Recall that these are $\frac{\partial S}{\partial a} = 0, \frac{\partial S}{\partial b} = 0, \frac{\partial S}{\partial c} = 0, \frac{\partial S}{\partial d} = 0$.) Do not solve the normal equations.

(b) Use the method of problem d105 to write the problem in matrix form $X\lambda = Y$, where $X(j,:) = [x_j^2, x_j, 1, 1/x_j], Y(j,1) = y_j$ and $\lambda = [a; b; c; d]$. Then generate synthetic data and use solve this system of equations $\lambda = X \setminus Y$.

Solution:

(a) The normal equations are

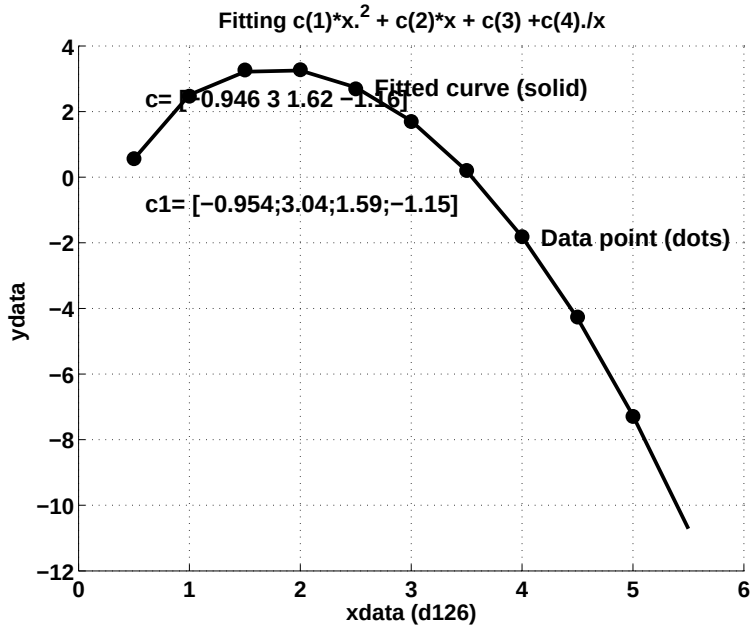
$$\begin{aligned}
\frac{\partial S}{\partial a} &= 2\sum_{j=1}^{m}\left[ax_j^4 + bx_j^3 + (c-y_j)x_j^2 + dx_j\right] = 0\\
\frac{\partial S}{\partial b} &= 2\sum_{j=1}^{m}\left[ax_j^3 + bx_j^2 + (c-y_j)x_j + d\right] = 0\\
\frac{\partial S}{\partial c} &= 2\sum_{j=1}^{m}\left[ax_j^2 + bx_j + (c-y_j) + \frac{d}{x_j}\right] = 0\\
\frac{\partial S}{\partial d} &= 2\sum_{j=1}^{m}\left[ax_j + b + \frac{(c-y_j)}{x_j} + \frac{d}{x_j^2}\right] = 0.
\end{aligned}$$

(b)
```
% FILE  d126.m  begins.
% Fit c(1)*x.^2 + c(2)*x + c(3) +c(4)./x  to data
% This program, with small modifications, applies
% also to some other linear problems

close all
clear
for jj=1:5
figure


fmodel=inline('c(1)*x.^2 + c(2)*x + c(3) +c(4)./x ','x','c');
fobj=inline('norm(feval(fmodel,x,c)-y)','c','fmodel','x','y');
% fobj =  sum of squares
xdata= 0.5:0.5:5;
c1=4*rand(4,1)-3*rand(4,1);

y=fmodel(xdata,c1);   % Generate syntetic data
```

```
                        % with parameters c1
ydata= y.*(0.97+0.05*rand(1,length(xdata)));
                        % Add some "errors"

xx=xdata'; yy=ydata';
d1=length(xx);
d2=length(c1);
A=zeros(d1, d2);
A=[xx.^2  xx  ones(d1,1) 1./xx];
myc=A\yy;
c=myc';
yfinal=fobj(c,fmodel,xdata,ydata);  % Final value of object function
fprintf('\nSum of squares = %12.3f \n c1  =',yfinal)
fprintf(' %12.3f ',c1)
fprintf('\ncest =')
fprintf(' %12.3f ',c')
fprintf( '\n')


x=0.5:0.5:5.5;
yfit=fmodel(x, c);

axes('FontSize',[15],'FontWeight','bold'); hold on;
title(['Fitting c(1)*x.^2 + c(2)*x + c(3) +c(4)./x'])
plt=plot(x,yfit,xdata,ydata,'k.','MarkerSize',30);  grid on;
txt1=' {\bf Fitted curve (solid)}';
text(x(5)+0.05,yfit(5),txt1,'FontWeight','bold','FontSize',[16]);
txt2=' {\bf Data point (dots)}';
text(xdata(8)+0.05,ydata(8),txt2,'FontWeight','bold','FontSize',[16]);
ax=axis; x1=ax(1)+0.1*(ax(2)-ax(1));
y1=ax(3)+0.9*(ax(4)-ax(3));
%y1=max(y1,0.5+max(ydata));
text(x1,y1, ['c= ' mat2str(c,3)],'FontWeight','bold','FontSize',[16]);
y2=ax(3)+0.7*(ax(4)-ax(3));
%y2=max(y2,0.3+max(ydata));
text(x1,y2, ['c1= ' mat2str(c1,3)],'FontWeight','bold','FontSize',[16]);
ylabel('ydata '); xlabel('xdata (d126)')
set(plt,'LineWidth',2.5);
pause(1.5)
end
% FILE  d126.m  ends.
```

Output:

```
Sum of squares =        0.093
 c1  =       -0.468        1.618        2.261        1.710
cest =       -0.461        1.562        2.391        1.617

Sum of squares =        0.673
 c1  =        1.210       -0.155        2.660        0.888
cest =        1.281       -0.689        3.554        0.513

Sum of squares =        0.145
 c1  =       -0.611        0.606       -1.287        2.951
cest =       -0.642        0.811       -1.603        3.045

Sum of squares =        0.298
 c1  =        0.036        2.121        1.931       -0.678
cest =        0.015        2.212        1.870       -0.673

Sum of squares =        0.096
 c1  =       -0.954        3.037        1.587       -1.152
cest =       -0.946        3.000        1.617       -1.161
```



HSolutions in Python

Problem 1.

```
# FILE: d121.py begins
# Forsythe-Moler p. 25 says that if A=[a b; c d] satisfies
# det(A)=ad-bc not zero and s= (a*a+b*b+c*c+d*d)/(2*det(A))
# then cond(A)= s + sqrt(s*s-1)
from mmeutil import *
from Numeric import *
from LinearAlgebra import *
n=2
print 'd121:  Error'
for p in range(15):
   A=3.0*(ranmat(n,n)-0.5)
   t= sum(sum(A[:]*A[:]))
   s=t/(2.0*abs(determinant(A)))
   tmp=cond(A)-s-sqrt(abs(s*s-1.0))
   print '% 12.2e' %(tmp)
# FILE: d121.py ends
```

Output:

```
d121:  Error
   -4.44e-16
    2.31e-14
   -6.66e-16
    9.44e-16
    4.00e-15
    1.11e-16
    1.42e-14
    2.22e-15
    1.78e-15
   -8.88e-16
    2.66e-15
    4.44e-16
    9.99e-16
    0.00e+00
    2.22e-15
```

Problem 2.

```
# FILE  d122.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]/(1.0+(x-lam[1])**2.0)+1.0/(1.0+lam[2]**2.0)

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

xy=getmat2('d122dat.dat')
xdata=xy[:,0]
ydata=xy[:,1]
lam0=array([1.0,-1.0,2.0])  # 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=arange(min(xdata),max(xdata),0.01)
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 d122.py ends
```

Output:

```
Optimization terminated successfully.
         Current function value: 2.246949
         Iterations: 78
         Function evaluations: 143
   1.252
  -0.153
```

```
   1.153

0.437691491932
```

Problem 3.

```
# FILE  d123.py  begins.
# Corrected version of e916.m
# Solution of boundary value problem by finite difference method:
#    y(a)=bval1, y(b) = bval2, y" = p(t)y' + q(t)y + r(t)
# Put t(j) = a + j*h, j = (b-a)/(n+1), j = 0,1,..,n+1,
# FD approximation gives:
# y(t(j-1)) - 2y(t(j)) + y(t(j+1)) =
#    (h/2)(y(t(j+1)) -y(t(j-1)))p(t(j))+
#     h*h*q(t(j)) y(t(j)) + h*h*r(t(j)) or
# y(t(j-1))(1 +(h/2)p(t(j))) - (2+ h*h*q(t(j))) y(t(j))
#     + y(t(j+1))(1 -(h/2)p(t(j))) = h*h*r(t(j)), j = 1,,.n
# y(t(0)) = bval1, y(t(n+1)) = bval2
# Compare with exact solution given below
from mmeutil import *
from Numeric import *
from LinearAlgebra import *

print 'd123\n  n     max.error'
for N in range(8,20):
   a,b=1.0,2.0
   h,bval1,bval2 =(b-a)/N,1.0,2.0
   tt= arange(a,b,h)
   t = tt[1:len(tt)-1]
   n = len(t)
   one = ones(n)
   # There are n interior points and n unknowns
   p,q,r =-2.*one/t,2.*one/(t**2.), sin(log(t))/(t**2.)
   diagon = -(2.*one +h*h*q)
   tmp = (one- 0.5*h*p)
   superd = tmp[0:n-1]
   tmp = (one+ 0.5*h*(p))
   subdia = tmp[1:n]
   mtx = diag(diagon) + diag(superd,1) + diag(subdia,-1)
   rhs =h*h*transpose(r)
   rhs[0] = rhs[0] -bval1*(1.+(-2/(a+h))*0.5*h)
   rhs[n-1] = rhs[n-1] -bval2*(1.-(-2./(b-h))*0.5*h)
```

```
   sol = SVDsolve(mtx,rhs)
   c2 = (1./70.)*(8.0-12.*sin(log(2)) - 4.*cos(log(2.)))
   exact =(1.1-c2)*t +c2*one/(t**2.0)-0.3*sin(log(t))-0.1*cos(log(t))
   print '%2d   %12.3e'% (N,mnormp(sol-transpose(exact)))
# FILE  d123.py  ends.
```

Output:

```
d123
  n     max.error
 8      1.884e-01
 9      1.177e-04
10      1.779e-01
11      1.728e-01
12      7.673e-05
13      1.635e-01
14      6.096e-05
15      5.499e-05
16      1.516e-01
17      4.560e-05
18      4.187e-05
19      3.861e-05
```

Problem 4.

```
# FILE d124.py begins
from mmeutil import *
from Numeric import *
#                             % b
# 4*p1 = -1 + 2 + p2 + 2;   % 3
# 4*p2 = p1 + p3 + 5 + 4;   % 9
# 4*p3 = 2 + 1 + p4 + p2;   % 3
# 4*p4 = p3 + 3 + 6 + 5;    % 14
a = array([[4.,-1.,0.,0.],[-1.,4.,-1.,0.],\
           [0.,-1.,4.,-1],[0.,0.,-1.,4.]])
b = array([ 3., 9., 3., 14.])
print a,b
t = SVDsolve(a,b)
print t
# FILE d124.py ends.
```

Output:

```
[[ 4. -1.  0.  0.]
 [-1.  4. -1.  0.]
 [ 0. -1.  4. -1.]
 [ 0.  0. -1.  4.]] [  3.   9.   3.  14.]
[ 1.57416268  3.29665072  2.61244019  4.15311005]
```

Problem 5.

```
# FILE: d125.py begins
from mmeutil import *
from Numeric import *
#                               # b
# 4*p1 = -1 + p3 + p2 +1        # 0
# 4*p2 = p1 + p4 + 2 + 2        # 4
# 4*p3 = -2 + -2 + p4 + p1      # -4
# 4*p4 = p3 + -1 + p5 + p2      # -1
# 4*p5 = p4 + p6 + 2 + 2        # 4
# 4*p6 = -1 + -1 + p7 + p5      # -2
# 4*p7 = p6 + p8+ 2 + 2         # 4
# 4*p8 = -1 + -1 + 1 + p7       # -1
a = array([[ 4., -1., -1., 0., 0., 0., 0., 0.],\
     [-1., 4., 0., -1., 0., 0., 0., 0.],\
     [-1., 0., 4., -1., 0., 0., 0., 0.],\
     [0., -1., -1., 4., -1., 0., 0., 0.],\
     [0., 0., 0., -1., 4., -1., 0., 0.],\
     [0., 0., 0., 0., -1., 4., -1., 0.,],\
     [0., 0., 0., 0., 0., -1., 4., -1.,],\
     [0., 0., 0., 0., 0., 0., -1., 4.]])
b = array([0., 4., -4., -1., 4., -2., 4., -1.,])
print a
print b
t = SVDsolve(a,b)
print t
# FILE: d125.py ends
```

Output:

```
[[ 4. -1. -1.  0.  0.  0.  0.  0.]
 [-1.  4.  0. -1.  0.  0.  0.  0.]
 [-1.  0.  4. -1.  0.  0.  0.  0.]
 [ 0. -1. -1.  4. -1.  0.  0.  0.]
 [ 0.  0.  0. -1.  4. -1.  0.  0.]
```

```
 [ 0.  0.  0.  0. -1.  4. -1.  0.]
 [ 0.  0.  0.  0.  0. -1.  4. -1.]
 [ 0.  0.  0.  0.  0.  0. -1.  4.]]
[ 0.  4. -4. -1.  4. -2.  4. -1.]
[  5.55111512e-16   1.00000000e+00  -1.00000000e-00   3.33066907e-16
        1.00000000e+00  -6.10622664e-16   1.00000000e+00  -2.49800181e-16]
```