```
% gda13_01
% supports Section 13.1
% apply Pavlis and Booker (1980) variable partitioning
% method to a simple inverse problem, randomly generated
% The script shows that the solution to the partitioned
% problem is the same as to the unpartitioned problem.

clear all;

N=100;
M=10;
G=random('uniform',0,1,N,M);      % random data kernel
mtrue=random('uniform',-1,1,M,1); % random true model parameters
dtrue=G*mtrue; % true data
dobs=dtrue; % no noise in this problem, so error should be zero

% solve by least squares
mest=(G'*G)\(G'*dobs);

% some solution statistics
dpre=G*mest;
e=dobs-dpre;
E=e'*e;
fprintf('Prediction error of full inverse problem: %f\n', E );
l = mest-mtrue;
L = l'*l;
fprintf('Distance between full and true solutions: %f\n', L );

% partitioned problem
% divide model parameters into two groups m=[m1',m2']'
M1=5;
M2=M-M1;
m1true=mtrue(1:M1);
m2true=mtrue(M1+1:end);
G1=G(:,1:M1);
G2=G(:,M1+1:end);

% svd of first data kernel
[U,LAMBDA,V] = svd(G1);
lambda=diag(LAMBDA);
Up = U(:,1:M1);
U0 = U(:,M1+1:end);
LAMBDAp=LAMBDA(1:M1,1:M1);
Vp=V(1:M1,1:M1);

% first partitioned problem, Gp*m2=dp
dp = U0'*dobs;
Gp = U0'*G2;

% solve by least squares
m2est = (Gp'*Gp)\(Gp'*dp);

% second partitioned problem, Gpp*m1=dpp
dpp = Up'*(dobs-G2*m2est);
Gpp = LAMBDAp*Vp';

% solve by least squares
m1est = (Gpp'*Gpp)\(Gpp'*dpp);
```

```
% some solution statistics
d12pre=G1*m1est+G2*m2est;
e12=dobs-d12pre;
E12=e12'*e12;
fprintf('Prediction error of partitioned inverse problem: %f\n', E12 );
m12est = [m1est', m2est']';
l12 = m12est-mtrue;
L12 = l12'*l12;
fprintf('Distance between partitioned and true solutions: %f\n', L12 );
```