

```

% gda08_03
%
% L1 norm inverse problem, overdetermined case
% solved by transformation to a linear programming problem
% supports Figure 8.3

% auxillary variable, z
N=20;
zmin = 0;
zmax = 1;
Dz = (zmax-zmin)/(N-1);
z = zmin + Dz*[0:N-1]';

% set up for cubic fit
M=4;
mtrue = [4, 3, 2, 1]';
G = [ones(N,1), z, z.^2, z.^3];
dtrue = G*mtrue;
sd = 1 * ones(N,1);
dobs=zeros(N,1);

% add exponentially-distributed random noise
for i=[1:N]
    r=random('exp',sd(i)/sqrt(2)).*(2*(random('unid',2)-1.5));
    dobs(i) = dtrue(i) + r;
end

% outlier
dobs(N)=5;

% least squares solution (sure easier!)
mls = (G'*G)\(G'*dobs);
dls = G*mls;

% L1 solution
% set up for linear programming problem
% min f*x subject to A x <= b, Aeq x = beq

% I need to add an 'options' argument to linprog() for
% MATLAB release 2013a and above, so determine the release year
nMLver = sscanf(version('-release'),' %d');
if( nMLver >= 2013 )
    options = optimoptions('linprog','Algorithm','interior-point-legacy');
end

% variables
% m = mp - mpp
% x = [mp', mpp', alpha', x', xp']
% with mp, mpp length M and alpha, x, xp, length N
L = 2*M+3*N;
x = zeros(L,1);

% f is length L
% minimize sum alpha(i)/sd(i)
f = zeros(L,1);
f(2*M+1:2*M+N)=1./sd;

% equality constraints
Aeq = zeros(2*N,L);

```

```

beq = zeros(2*N,1);
% first equation G(mp-mpp)+x-alpha=d
Aeq(1:N,1:M) = G;
Aeq(1:N,M+1:2*M) = -G;
Aeq(1:N,2*M+1:2*M+N) = -eye(N,N);
Aeq(1:N,2*M+N+1:2*M+2*N) = eye(N,N);
beq(1:N) = dobs;
% second equation G(mp-mpp)-xp+alpha=d
Aeq(N+1:2*N,1:M) = G;
Aeq(N+1:2*N,M+1:2*M) = -G;
Aeq(N+1:2*N,2*M+1:2*M+N) = eye(N,N);
Aeq(N+1:2*N,2*M+2*N+1:2*M+3*N) = -eye(N,N);
beq(N+1:2*N) = dobs;

% inequality constraints A x <= b
% part 1: everything positive
A = zeros(L+2*M,L);
b = zeros(L+2*M,1);
A(1:L,:) = -eye(L,L);
b(1:L) = zeros(L,1);
% part 2; mp and mpp have an upper bound. Note:
% Without this constraint, a potential problem is that
% mp and mpp are individually unbounded, though their
% difference, m=mp-mpp, is not. Thus there might be cases
% where the algoirthm strays to very large mp and mpp.
A(L+1:L+2*M,:) = eye(2*M,L);
mupperbound=10*max(abs(mls)); % might need to be adjusted for problem at hand
b(L+1:L+2*M) = mupperbound;

% solve linear programming problem
if( nMLver>= 2013 )
    [x, fmin] = linprog(f,A,b,Aeq,beq,[],[],options);
else
    [x, fmin] = linprog(f,A,b,Aeq,beq);
end

```

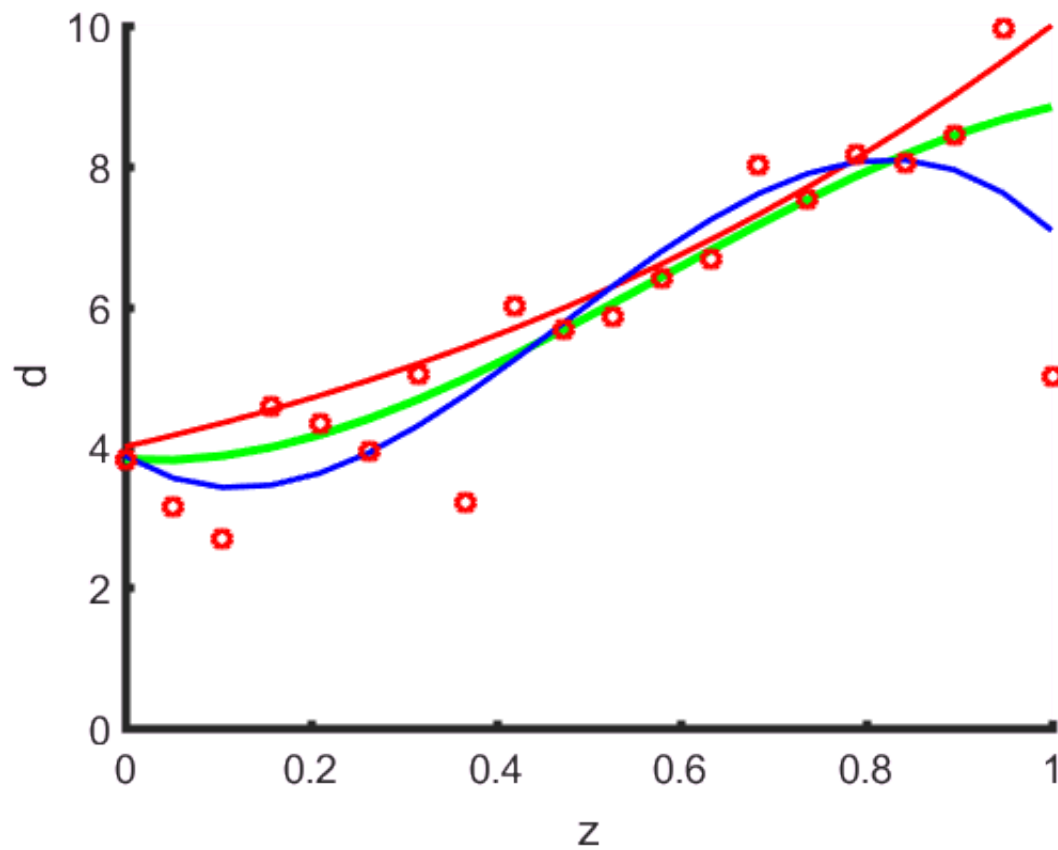
Optimization terminated.

```

fmin=-fmin;
mest = x(1:M) - x(M+1:2*M);
dpre = G*mest;

% plot true, observed & predcted data
figure(1);
clf;
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [zmin, zmax, 0, max(dtrue) ] );
plot( z, dtrue, 'r-', 'Linewidth', 2);
plot( z, dpre, 'g-', 'Linewidth', 3);
plot( z, dls, 'b-', 'Linewidth', 2);
plot( z, dobs, 'ro', 'Linewidth', 2);
xlabel('z');
ylabel('d');

```



% Figure 8.3 Curve fitting using the L1 norm. The true data (red curve) follow a cubic polynomial in an auxiliary variable, z . Observations (red circles) have additive noise with zero mean and variance $\sigma^2 = 1$ drawn from an exponential probability density function. Note the outlier at $z=1$. The L1 fit (green curve) is not as affected by the outlier as a standard L2 (least-squares) fit (blue curve). MatLab script gda08_03.