```
% gda12_07
%
% L1, L2 and Linf norm; fitting of a straight line
% L1 and Linf are via transformation to a linear
% programming problem; Ls is via least squares.
% Supports Figure 12.11

clear all;

% auxillary variable, z
N=10;
zmin = 0;
zmax = 1;
Dz = (zmax-zmin)/(N-1);
z = zmin + Dz*[0:N-1]';

% set up for linear fit
M=2;
mtrue = [1, 3]';
G = [ones(N,1), z];
dtrue = G*mtrue;

% syntehtic data, with noise drawn from a two-sided exponential
% distribution with variance sd^2.
sd = 0.4 * ones(N,1);
dobs=zeros(N,1);
for i=[1:N]
    r=random('exp',sd(i)/sqrt(2)).*(2*(random('unid',2)-1.5));
    dobs(i) = dtrue(i) + r;
end

% Part 1: L2 Norm
% least squares solution (sure the easiest!)
mls = (G'*G)\(G'*dobs);
dls = G*mls;

% Part 2: L1 norm
% set up for linear programming problem
% min f*x subject to A x <= b,  Aeq x = beq

% 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 aplha(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 algroithm 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;

% 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

% 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;
mestL1 = x(1:M) - x(M+1:2*M);
dpreL1 = G*mestL1;

% Part 3: Linf Norm
% set up for linear programming problem
% min f*x subject to A x <= b,  Aeq x = beq

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

% f is length L
% minimize alpha
f = zeros(L,1);
f(2*M+1:2*M+1)=1;
```

```
% 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+1)     = -1./sd;
Aeq(1:N,2*M+1+1:2*M+1+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+1)         =  1./sd;
Aeq(N+1:2*N,2*M+1+N+1:2*M+1+2*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:
% A potential problem without this constraint is that
% mp and mpp are individually unbounded, though their
% difference, m=mp-mpp, is not.  Thus there might be cases
% where the algroithm 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;

% 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

% 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
```

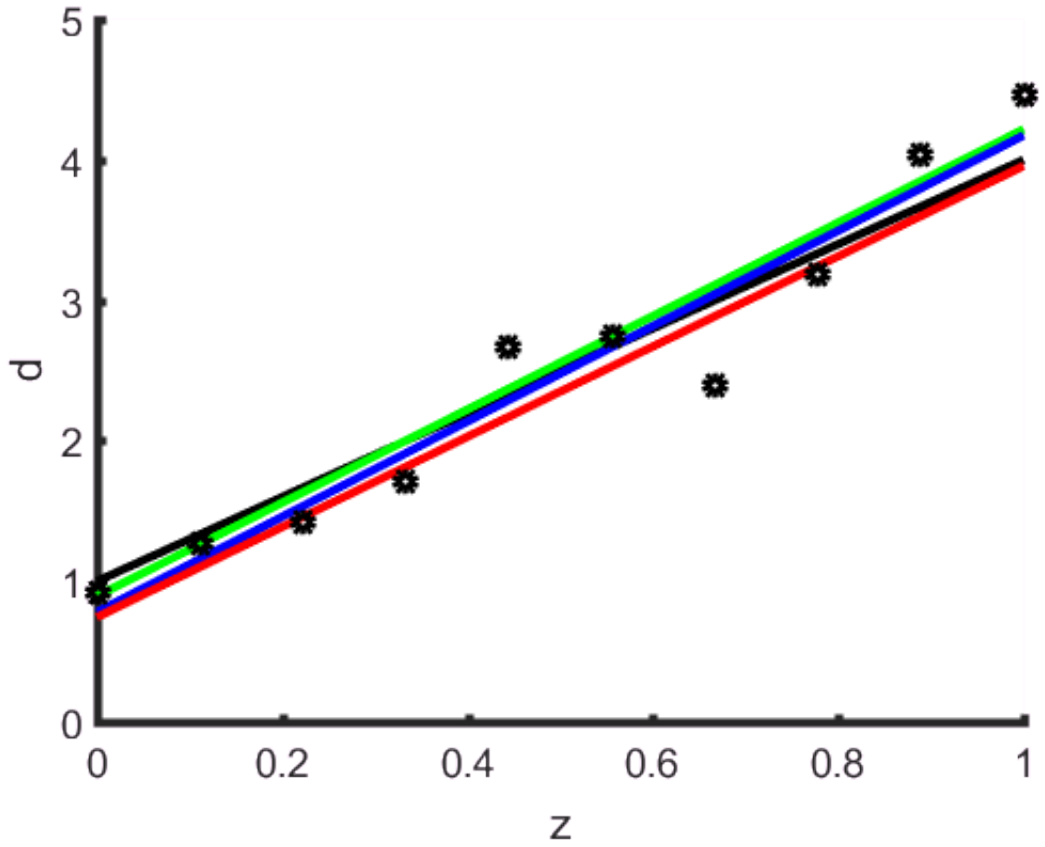
```
Optimization terminated.
```

```
fmin=-fmin;
mestLinf = x(1:M) - x(M+1:2*M);
dpreLinf = G*mestLinf;

% plot the observed and presicted data
figure(1);
clf;
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [zmin, zmax, 0, 5 ] );
plot( z, dtrue, 'k-', 'Linewidth', 3);
plot( z, dpreL1, 'g-', 'Linewidth', 3);
```

```
plot( z, dls, 'b-', 'Linewidth', 3);
plot( z, dpreLinf, 'r-', 'Linewidth', 3);
plot( z, dobs, 'ko', 'Linewidth', 3);

xlabel('z');
ylabel('d');
```




```
% Figure 12.11 Fitting a straight line to data d(z). The true model (black line) is the straig
% line, dtrue(z) = 1 + 3z. The N = 10 observations diobs are the true data perturbed with expo
% distributed noise with variance σd2 = (0.4)2. Three different fits have been computed, by mi
% the L1, L2, and L∞ norms of the error. The corresponding predicted data are shown in green,
% red, respectively. MatLab script gda12_07.

fprintf('black:   d-true\n');
```

```
black:   d-true
```

```
fprintf('circles: d-obs\n');
```

```
circles: d-obs
```

```
fprintf('green:   d-L1\n');
```

```
green:   d-L1
```

```
fprintf('blue:    d-L2\n');
```

```
blue:    d-L2
```

```
fprintf('red:     d-Linf\n');
```

```
red:     d-Linf
```