**CCM code with instructions:**

First, obtain the following data from particle tracks. Each of them are a vector array the size of the total number of particles.

* Xp is the x-locations of particles in the previous time, given time and the future time step.
* Yp is the y-locations of particles in the previous time, given time and the future time step.
* Zp is the z-locations of particles in the previous time, given time and the future time step.
* Up, Vp, Wp are the velocity components of the particles at the 3 time steps.
* axp, ayp, azp are the acceleration components of the particles at the 3 time steps.
* T is the associated time step indicator, e.g. if a particle is from previous time step, it is set to -1, whereas for a particle from future time step the value is 1.
* weight: particles can be given weights based on how much they can be trusted,e.g, particles at end of tracks may be less trust-worthy, by default it is set as an arrays of 1. (no special preference).

**SVD initialization code:**

parfor indexcount=1:1:N1\*N2\*N3 % calculating at all grid points (N1 is the number of gridpoints in x-direction and so on)

k=min(floor(indexcount/(N1\*N2))+1,N3);

j=floor(rem(indexcount,(N1\*N2))/N1)+1;

i= rem(rem(indexcount,(N1\*N2)),N1);

if rem(rem(indexcount,(N1\*N2)),N1)==0

i=N1;j=j-1;

end

if rem(indexcount,(N1\*N2))==0

j=N2;k=k-1;

end

point=[Xg(i),Yg(j),Zg(k)];

 [Sol, count] = SVD\_second(point,Xp,Yp,Zp,Up,Vp,Wp,Region,minN,maxN,T);

%Region is the largest allowed interpolation volume, minN is the minimum points for SVD to work (11 for second order, 4 for first order), maxN is the maximum allowed points.

datarow=[point;Sol(1:12);Sol(end-2:end)];

svd\_v(indexcount,:)=datarow;

[Sol, count] = SVD\_first(point,Xp,Yp,Zp,axp,ayp,azp,Region,minN,maxN,T);

datarow=[point;Sol(1:12);Sol(end-2:end)];

svd\_a(indexcount,:)=datarow;

end

**%% second order 4D SVD**

function [Sol,count] = SVD\_second (point,X,Y,Z,U,V,W,R,minN,maxN,T)

Xg = point(1);

Yg = point(2);

Zg = point(3);

R1 = R(1); % maximum radius of interpolation volume in x-direction

R2 = R(2);

R3 = R(3);

valid=((Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2)<1);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

T=T(valid);

count=length(X);

if count>maxN

R=(Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2);

[B,I] = sort(R);

valid=I(1:maxN);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

T=T(valid);

end

count=length(X);

Sol=zeros(33,1);

if count>=minN

y=zeros(count\*3,1);

A=zeros(count\*3,33);

for i=1:count

dx=X(i)-Xg;

dy=Y(i)-Yg;

dz=Z(i)-Zg;

r=[dx,dy,dz];

r2=[0.5\*dx^2,0.5\*dy^2,0.5\*dz^2,dx\*dy,dx\*dz,dy\*dz];

y((i-1)\*3+1:i\*3)=[U(i);V(i);W(i)];

A((i-1)\*3+1,:)=[1,0,0,r,0,0,0,0,0,0,r2,0,0,0,0,0,0,0,0,0,0,0,0,T(i),0,0];

A((i-1)\*3+2,:)=[0,1,0,0,0,0,r,0,0,0,0,0,0,0,0,0,r2,0,0,0,0,0,0,0,T(i),0];

A((i-1)\*3+3,:)=[0,0,1,0,0,0,0,0,0,r,0,0,0,0,0,0,0,0,0,0,0,0,r2,0,0,T(i)];

end

Sol = pinv(A)\*y;

end

**CCM code**

uprior=reshape(svd\_v(:,4),N1,N2,N3);

vprior=reshape(svd\_v (:,5),N1,N2,N3);

wprior=reshape(svd\_v (:,6),N1,N2,N3);

uprior2=reshape(svd\_a(:,4),N1,N2,N3);

vprior2=reshape(svd\_a (:,5),N1,N2,N3);

wprior2=reshape(svd\_a (:,6),N1,N2,N3);

dudx=reshape(svd\_v (:,7),N1,N2,N3);

dudy=reshape(svd\_v (:,8),N1,N2,N3);

dudz=reshape(svd\_v (:,9),N1,N2,N3);

dvdx=reshape(svd\_v (:,10),N1,N2,N3);

dvdy=reshape(svd\_v (:,11),N1,N2,N3);

dvdz=reshape(svd\_v (:,12),N1,N2,N3);

dwdx=reshape(svd\_v (:,13),N1,N2,N3);

dwdy=reshape(svd\_v (:,14),N1,N2,N3);

dwdz=reshape(svd\_v (:,15),N1,N2,N3);

dudt=reshape(svd\_v (:,16),N1,N2,N3);

dvdt=reshape(svd\_v (:,17),N1,N2,N3);

 dwdt=reshape(svd\_v (:,18),N1,N2,N3);

dudx2=reshape(svd\_a (:,7),N1,N2,N3);

dudy2=reshape(svd\_a (:,8),N1,N2,N3);

dudz2=reshape(svd\_a (:,9),N1,N2,N3);

dvdx2=reshape(svd\_a (:,10),N1,N2,N3);

dvdy2=reshape(svd\_a (:,11),N1,N2,N3);

dvdz2=reshape(svd\_a (:,12),N1,N2,N3);

dwdx2=reshape(svd\_a (:,13),N1,N2,N3);

dwdy2=reshape(svd\_a (:,14),N1,N2,N3);

dwdz2=reshape(svd\_a (:,15),N1,N2,N3);

dudt2=reshape(svd\_a (:,16),N1,N2,N3);

dvdt2=reshape(svd\_a (:,17),N1,N2,N3);

 dwdt2=reshape(svd\_a (:,18),N1,N2,N3);

% If including viscous terms, use the gradient function to get double derivatives of velocity from first derivatives

[dudxdy,dudxdx,dudxdz] = gradient(dudx,delta);

[dudydy,dudydx,dudydz] = gradient(dudy,delta);

[dudzdy,dudzdx,dudzdz] = gradient(dudz,delta);

[dvdxdy,dvdxdx,dvdxdz] = gradient(dvdx,delta);

[dvdydy,dvdydx,dvdydz] = gradient(dvdy,delta);

[dvdzdy,dvdzdx,dvdzdz] = gradient(dvdz,delta);

[dwdxdy,dwdxdx,dwdxdz] = gradient(dwdx,delta);

[dwdydy,dwdydx,dwdydz] = gradient(dwdy,delta);

[dwdzdy,dwdzdx,dwdzdz] = gradient(dwdz,delta);

visc\_d (:,1)=reshape(-nu\*(dudxdx+dudydy+dudzdz),N,1);

visc\_d(:,2)=reshape(-nu\*(dvdxdx+dvdydy+dvdzdz),N,1);

visc\_d(:,3)=reshape(-nu\*(dwdxdx+dwdydy+dwdzdz),N,1);

[dudyv,dudxv,dudzv] = gradient(-nu\*(dudxdx+dudydy+dudzdz),,delta);

[dvdyv,dvdxv,dvdzv] = gradient(-nu\*(dvdxdx+dvdydy+dvdzdz),,delta);

[dwdyv,dwdxv,dwdzv] = gradient(-nu\*(dwdxdx+dwdydy+dwdzdz),,delta);

curl(:,1)=-reshape(dwdyv-dvdzv,N,1);

curl(:,2)=-reshape(dudzv-dwdxv,N,1);

curl(:,3)=-reshape(dvdxv-dudyv,N,1);

for indexcount=1:1:N1\*N2\*N3

k=min(floor(indexcount/(N1\*N2))+1,N3);

j=floor(rem(indexcount,(N1\*N2))/N1)+1;

i= rem(rem(indexcount,(N1\*N2)),N1);

if rem(rem(indexcount,(N1\*N2)),N1)==0

i=N1; j=j-1;

end

if rem(indexcount,(N1\*N2))==0

j=N2; k=k-1;

end

% Setting range for the S matrix

is=max(1,i-3);

ie=min(i+3,N1);

js=max(1,j-3);

je=min(j+3,N2);

ks=max(1,k-3);

ke=min(k+3,N3);

n=(ie-is+1)\*(je-js+1)\*(ke-ks+1);

s=zeros(n,33);

s2=zeros(n,15); s(:,1:3)=[reshape(uprior(is:ie,js:je,ks:ke),[],1),reshape(vprior(is:ie,js:je,ks:ke),[],1),reshape(wprior(is:ie,js:je,ks:ke),[],1)];

s2(:,1:3)=[reshape(uprior2(is:ie,js:je,ks:ke),[],1),reshape(vprior2(is:ie,js:je,ks:ke),[],1),reshape(wprior2(is:ie,js:je,ks:ke),[],1)];

s(:,4:12)=[(reshape(dudx(is:ie,js:je,ks:ke),[],1)),(reshape(dudy(is:ie,js:je,ks:ke),[],1)),(reshape(dudz(is:ie,js:je,ks:ke),[],1)),(reshape(dvdx(is:ie,js:je,ks:ke),[],1)),(reshape(dvdy(is:ie,js:je,ks:ke),[],1)),(reshape(dvdz(is:ie,js:je,ks:ke),[],1)),(reshape(dwdx(is:ie,js:je,ks:ke),[],1)),(reshape(dwdy(is:ie,js:je,ks:ke),[],1)),(reshape(dwdz(is:ie,js:je,ks:ke),[],1))];

s2(:,4:12)=[(reshape(dudx2(is:ie,js:je,ks:ke),[],1)),(reshape(dudy2(is:ie,js:je,ks:ke),[],1)),(reshape(dudz2(is:ie,js:je,ks:ke),[],1)),(reshape(dvdx2(is:ie,js:je,ks:ke),[],1)),(reshape(dvdy2(is:ie,js:je,ks:ke),[],1)),(reshape(dvdz2(is:ie,js:je,ks:ke),[],1)),(reshape(dwdx2(is:ie,js:je,ks:ke),[],1)),(reshape(dwdy2(is:ie,js:je,ks:ke),[],1)),(reshape(dwdz2(is:ie,js:je,ks:ke),[],1))];

s(:,13:21)=[(reshape(dudxdx(is:ie,js:je,ks:ke),[],1)),(reshape(dudydy(is:ie,js:je,ks:ke),[],1)),(reshape(dudzdz(is:ie,js:je,ks:ke),[],1)),(reshape(dudxdy(is:ie,js:je,ks:ke),[],1)),(reshape(dudxdz(is:ie,js:je,ks:ke),[],1)),(reshape(dudydz(is:ie,js:je,ks:ke),[],1)),(reshape(dvdxdx(is:ie,js:je,ks:ke),[],1)),(reshape(dvdydy(is:ie,js:je,ks:ke),[],1)),(reshape(dvdzdz(is:ie,js:je,ks:ke),[],1))];

s(:,22:30)=[(reshape(dvdxdy(is:ie,js:je,ks:ke),[],1)),(reshape(dvdxdz(is:ie,js:je,ks:ke),[],1)),(reshape(dvdydz(is:ie,js:je,ks:ke),[],1)),(reshape(dwdxdx(is:ie,js:je,ks:ke),[],1)),(reshape(dwdydy(is:ie,js:je,ks:ke),[],1)),(reshape(dwdzdz(is:ie,js:je,ks:ke),[],1)),(reshape(dwdxdy(is:ie,js:je,ks:ke),[],1)),(reshape(dwdxdz(is:ie,js:je,ks:ke),[],1)),(reshape(dwdydz(is:ie,js:je,ks:ke),[],1))];

s(:,31:33)=[(reshape(dudt(is:ie,js:je,ks:ke),[],1)),(reshape(dvdt(is:ie,js:je,ks:ke),[],1)),(reshape(dwdt(is:ie,js:je,ks:ke),[],1))];

s2(:,13:15)=[(reshape(dudt2(is:ie,js:je,ks:ke),[],1)),(reshape(dvdt2(is:ie,js:je,ks:ke),[],1)),(reshape(dwdt2(is:ie,js:je,ks:ke),[],1))];

if det(cov(s))~=0

s=inv(cov(s));

else

s=zeros(33,1);

s(1:3)=[1/cov(reshape(uprior(is:ie,js:je,ks:ke),[],1)),1/cov(reshape(vprior(is:ie,js:je,ks:ke),[],1)),1/cov(reshape(wprior(is:ie,js:je,ks:ke),[],1))];

s(4:12)=[(1/cov(reshape(dudx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdz(is:ie,js:je,ks:ke),[],1)))];

s(13:21)=[(1/cov(reshape(dudxdx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudydy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudzdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudxdy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudxdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudydz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdxdx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdydy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdzdz(is:ie,js:je,ks:ke),[],1)))];

s(22:30)=[(1/cov(reshape(dvdxdy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdxdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdydz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdxdx(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdydy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdzdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdxdy(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdxdz(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdydz(is:ie,js:je,ks:ke),[],1)))];

s(31:33)=[(1/cov(reshape(dudt(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdt(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdt(is:ie,js:je,ks:ke),[],1)))];

s=diag(s);

end

if det(cov(s2))~=0

s=inv(cov(s2));

else

s2=zeros(15,1);

s2(1:3)=[1/cov(reshape(uprior2(is:ie,js:je,ks:ke),[],1)),1/cov(reshape(vprior2(is:ie,js:je,ks:ke),[],1)),1/cov(reshape(wprior2(is:ie,js:je,ks:ke),[],1))];

s2(4:12)=[(1/cov(reshape(dudx2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudy2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dudz2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdx2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdy2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdz2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdx2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdy2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdz2(is:ie,js:je,ks:ke),[],1)))];

s2(13:15)=[(1/cov(reshape(dudt2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dvdt2(is:ie,js:je,ks:ke),[],1))),(1/cov(reshape(dwdt2(is:ie,js:je,ks:ke),[],1)))];

s2=diag(s2);

end

point=[Xg(i),Yg(j),Zg(k)];

xprior(1:3)=[svd\_v(indexcount,4); svd\_v (indexcount,5); svd\_v (indexcount,6)];

xprior(4:12)=[dudx(i,j,k);dudy(i,j,k);dudz(i,j,k);dvdx(i,j,k);dvdy(i,j,k);dvdz(i,j,k);dwdx(i,j,k);dwdy(i,j,k);dwdz(i,j,k)];

xprior(31:33)=[dudt(i,j,k);dvdt(i,j,k);dwdt(i,j,k)];

xprior(13:30)=[dudxdx(i,j,k),dudydy(i,j,k),dudzdz(i,j,k),dudxdy(i,j,k),dudxdz(i,j,k),dudydz(i,j,k),dvdxdx(i,j,k),dvdydy(i,j,k),dvdzdz(i,j,k),dvdxdy(i,j,k),dvdxdz(i,j,k),dvdydz(i,j,k),dwdxdx(i,j,k),dwdydy(i,j,k),dwdzdz(i,j,k),dwdxdy(i,j,k),dwdxdz(i,j,k),dwdydz(i,j,k)];

[Sol, count] = CCM\_vel(point,Xp,Yp,Zp,Up,Vp,Wp,Region,maxN,xprior,s,weight,T);

datarow(1:18)=[point,Sol(1:15)'];

ccm\_v(indexcount,:)=datarow;

xprior=zeros(15,1);

xprior(1:3)=[uprior2(i,j,k);vprior2(i,j,k);wprior2(i,j,k)];

xprior(4:12)=[dudx2(i,j,k);dudy2(i,j,k);dudz2(i,j,k);dvdx2(i,j,k);dvdy2(i,j,k);dvdz2(i,j,k);dwdx2(i,j,k);dwdy2(i,j,k);dwdz2(i,j,k)];

xprior(13:15)=[dudt2(i,j,k);dvdt2(i,j,k);dwdt2(i,j,k)];

[Sol, count] = CCM\_acc(point,Xp,Yp,Zp,axp,ayp,azp,Region,maxN,xprior,s2,weight,curl,T);

datarow(1:18)=[point,Sol'];

ccm\_a(indexcount,:)=datarow;

end

**%% 4D CCM for velocity**

function [Sol,count] = CCM\_vel(point,X,Y,Z,U,V,W,R,maxN,xprior,S, weight,T)

Xg = point(1);

Yg = point(2);

Zg = point(3);

R1 = R(1);

R2 = R(2);

R3 = R(3);

valid=((Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2)<1);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

weight=weight(valid);

T=T(valid);

count=max(size(X));

Sol=zeros(33,1);

count=length(X);

if count>maxN

R=(Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2);

[B,I] = sort(R);

valid=I(1:maxN);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

T=T(valid);

weight=weight(valid);

count=maxN;

end

if nnz(T)>0

y=zeros(count\*3,1);

A=zeros(count\*3,33);

for i=1:count

dx=X(i)-Xg;

dy=Y(i)-Yg;

dz=Z(i)-Zg;

r=[dx,dy,dz];

r2=[0.5\*dx^2,0.5\*dy^2,0.5\*dz^2,dx\*dy,dx\*dz,dy\*dz];

y((i-1)\*3+1:i\*3)=[U(i);V(i);W(i)];

A((i-1)\*3+1,:)=[1,0,0,r,0,0,0,0,0,0,r2,0,0,0,0,0,0,0,0,0,0,0,0,T(i),0,0];

A((i-1)\*3+2,:)=[0,1,0,0,0,0,r,0,0,0,0,0,0,0,0,0,r2,0,0,0,0,0,0,0,T(i),0];

A((i-1)\*3+3,:)=[0,0,1,0,0,0,0,0,0,r,0,0,0,0,0,0,0,0,0,0,0,0,r2,0,0,T(i)];

w((i-1)\*3+1:i\*3)=[1/cov(U),1/cov(V),1/cov(W)];

w((i-1)\*3+1:i\*3)=w((i-1)\*3+1:i\*3).\*weight(i);

end

Wi=diag(w);

w=[U;V;W];

if det(cov(w'))~=0

M=inv(cov(w'));

for j=1:count

Wi(3\*j-2:3\*j,3\*j-2:3\*j)=M.\*weight(j);

end

end

S=S./500;% Scalar k

b=zeros(33,1);

b(4)=1;

b(8)=1;

b(12)=1;

I=pinv(S+transpose(A)\*Wi\*A);

D=transpose(b)\*(I\*b);

for it=1:numberOfIterations

mu=transpose(b)\*(I\*(transpose(A)\*Wi\*y-S\*xprior))./D;

xest=I\*(transpose(A)\*Wi\*y-S\*xprior-mu.\*b);

xprior=xest;

end

Sol=xprior;

end

**%% 4D CCM for acceleration**

function [Sol,count] = CCM\_acc (point,X,Y,Z,U,V,W,R,maxN,xprior,S, weight,curl,T)

Xg = point(1);

Yg = point(2);

Zg = point(3);

R1 = R(1);

R2 = R(2);

R3 = R(3);

valid=((Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2)<1);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

weight=weight(valid);

T=T(valid);

count=max(size(X));

Sol=zeros(15,1);

count=length(X);

if count>maxN

R=(Xg-X).^2./(R1^2) + (Yg-Y).^2./(R2^2)+ (Zg-Z).^2./(R3^2);

[B,I] = sort(R);

valid=I(1:maxN);

X=X(valid);

Y=Y(valid);

Z=Z(valid);

U=U(valid);

V=V(valid);

W=W(valid);

T=T(valid);

weight=weight(valid);

count=maxN;

end

if nnz(T)>0

y=zeros(count\*3,1);

A=zeros(count\*3,15);

for i=1:count

dx=X(i)-Xg;

dy=Y(i)-Yg;

dz=Z(i)-Zg;

r=[dx,dy,dz];

y((i-1)\*3+1:i\*3)=[U(i);V(i);W(i)];

A((i-1)\*3+1,:)=[1,0,0,r,0,0,0,0,0,0,T(i),0,0];

A((i-1)\*3+2,:)=[0,1,0,0,0,0,r,0,0,0,0,T(i),0];

A((i-1)\*3+3,:)=[0,0,1,0,0,0,0,0,0,r,0,0,T(i)];

w((i-1)\*3+1:i\*3)=[1/cov(U),1/cov(V),1/cov(W)];

w((i-1)\*3+1:i\*3)=w((i-1)\*3+1:i\*3).\*weight(i);

end

Wi=diag(w);

w=[U;V;W];

if det(cov(w'))~=0

M=inv(cov(w'));

for j=1:count

Wi(3\*j-2:3\*j,3\*j-2:3\*j)=M.\*weight(j);

end

end

S=S./500;% Scalar k

b=zeros(15,1);

 c=zeros(15,1);

d=zeros(15,1);

b(11)=1;

b(9)=-1;

c(6)=1;

c(10)=-1;

d(7)=1;

d(5)=-1;

I=pinv(S+transpose(A)\*Wi\*A);

O(1,:) = [transpose(b)\*(I\*b), transpose(b)\*(I\*c), transpose(b)\*(I\*d)];

O(2,:) = [transpose(c)\*(I\*b), transpose(c)\*(I\*c), transpose(c)\*(I\*d)];

O(3,:) = [transpose(d)\*(I\*b), transpose(d)\*(I\*c), transpose(d)\*(I\*d)];

for i=1:numOfiterations

P=[transpose(b)\*(I\*(transpose(A)\*Wi\*y-S\*xprior))-curl(1);transpose(c)\*(I\*(transpose(A)\*Wi\*y-S\*xprior))-curl(2);transpose(d)\*(I\*(transpose(A)\*Wi\*y-S\*xprior))-curl(3)];

Mu = linsolve(O,P);

xprior=I\*(transpose(A)\*Wi\*y-S\*xprior-Mu(1)\*b-Mu(2)\*c-Mu(3)\*d);

end

Sol=xprior;

end