# Appendix 1

Karl-Fredrik Boholm Kylesten

May 2019

## Contents



## 1 Introduction

In this Appendix all code used for *Designing method for ESS neutrino Super Beam transfer Line* is listed.

# 2 Transfer matrices

## 2.1 Drift space

```
function M=M_Drift(s,gamma)
%Transfer matrix for drift space
%
M=eye(6);
M(1,2)=s;
M(3,4)=s;
M(5,6)=s/gamma;
end
```

## 2.2 Dipole magnet

```
function M=M_dipole(s,R,gamma2)
%Transfer matrix for a dipole magnet

M=eye(6);
C=cos(s/R);
S=sin(s/R);
M(1,1)=C;
M(2,1)=-S/R;
M(1,2)=R*S;
M(2,2)=C;
M(3,4)=s;
M(1,6) = R*(1-C);
M(2,6) = S;
M(5,1) = -S;
M(5,2) = -R*(1-C);
M(5,6) = s/gamma2;
end
```

## 2.3 Quadrupole magnet

```
function M=M_quad(k,s,gamma2)
%R_quad_long calculate Transfer matrix for a quadrudopol magnet
%
%R_quad_long(k,L) calculate Transfer matrix for a quadrudopol ...
    magnet with
%magnetfield gradiet k and length s;
M=eye(6);
k_rot=sqrt(abs(k));
omega=s*k_rot;
```

```
C=cos(omega);
CH=cosh(omega);
S=sin(omega);
SH=sinh(omega);
if (k≤0)
    M(1,1)=C;
    M(1,2)=S/k_rot;
    M(2,1)=-k_rot*S;
    M(2,2)=C;
    M(3,3)=CH;
    M(3,4)=SH/k_rot;
    M(4,3)=k_rot*SH;
    M(4,4)=CH;
else
    M(1,1)=CH;
    M(1,2)=SH/k_rot;
    M(2,1)=k_rot*SH;
    M(2,2)=CH;
    M(3,3)=C;
    M(3,4)=S/k_rot;
    M(4,3)=-k_rot*S;
    M(4,4)=C;
end
M(5,6)=s/gamma2;
end
```

# 3 Simulatiomethod 1

## 3.1 The simulation

```
%Simulate a FODO block
%Gives phase space and twiss parameters for each step
close all
clear all
%%%Properties of Matrices%%%
s_qf=0.5;
k_qf=-0.2;

%Dipole
s_dip=0.5;
angle=(15*pi/180)/2; % dipole half bending angle
r_dip=s_dip/angle; % Curvature radius of dipole
%X-Fokusing quadropole 1
s_qf1=s_qf;
k_f1=k_qf;
%X-Fokusing quadropole 2
s_qf2=s_qf;
k_f2=k_qf;
%X-Defocusing quadropole
```

```
s_qd=0.5;
k_d=-k_qf;
%Driftspace
s_drift=2;
%Relatevistic effects
gamma2=3.6; %Nummerical value
%%%The Matrices%%%
M_dip=M_dipole(s_dip,r_dip,gamma2);
M_drift=M_Drift(s_drift,gamma2);
M_QD=M_quad(k_d,s_qd,gamma2);
M_QF1=M_quad(k_f1,s_qf1,gamma2);
M_QF2=M_quad(k_f2,s_qf2,gamma2);
%%%The tunnel%%%
N_blocks=12;
Tunnel=zeros(6,6,N_blocks);
Z_length=zeros(1,N_blocks);
%One tunnel block
Tunnel(:,:,1)=M_QF1; %The matrix
Z_length(1)=s_qf; %The length
%
Tunnel(:,:,2)=M_drift;
Z_length(2)=s_drift;
%
Tunnel(:,:,3)=M_dip;
Z_length(3)=s_dip;
%
Tunnel(:,:,4)=M_dip;
Z_length(4)=s_dip;
%
Tunnel(:,:,5)=M_drift;
Z_length(5)=s_drift;
%
Tunnel(:,:,6)=M_QD;
Z_length(6)=s_qd;
%
Tunnel(:,:,7)=M_QD;
Z_length(7)=s_qd;
%
Tunnel(:,:,8)=M_drift;
Z_length(8)=s_drift;%

Tunnel(:,:,9)=M_dip;
Z_length(9)=s_dip;
%
Tunnel(:,:,10)=M_dip;
Z_length(10)=s_dip;
%
Tunnel(:,:,11)=M_drift;
Z_length(11)=s_drift;
%
Tunnel(:,:,12)=M_QF2;
Z_length(12)=s_qf;
%%%Properties of incoming beam%%%
bx=14.36;    % (m) standard deviation beam size
```

```
gx=1/bx;    % (m) standard deviation beam divergence
by=5.67;    % (m) standard deviation beam size
gy=1/by;    % (m) standard deviation beam divergence
N=10000; %Number of particles in simulation
%%%Array structure with particles (dimension,#particle,step)%%%
XXX=zeros(6,N,N_blocks+1);
Z=zeros(N_blocks+1,1); %Vector with length
%%%Moving the partikles%%%
for p=1:N
    XXX(1,p,1)=sqrt(bx)*randn(1,1);
    XXX(2,p,1)=sqrt(gx)*randn(1,1);
    XXX(3,p,1)=sqrt(by)*randn(1,1);
    XXX(4,p,1)=sqrt(gy)*randn(1,1);
for t=2:(N_blocks+1)
    XXX(:,p,t)=Tunnel(:,:,t-1)*XXX(:,p,(t-1));
end
end
%%%Measuring em, alpha, beta %%%
em_x=zeros(1,N_blocks+1);
em_y=zeros(1,N_blocks+1);
alpha_x=zeros(1,N_blocks+1);
alpha_y=zeros(1,N_blocks+1);
beta_x=zeros(1,N_blocks+1);
beta_y=zeros(1,N_blocks+1);
SIGMA=zeros(6,6,N_blocks+1);
for t=1:(N_blocks+1)
    x=XXX(1,:,t);
    xp=XXX(2,:,t);
    y=XXX(3,:,t);
    yp=XXX(4,:,t);
    Sigma=Sigma_measure(x,xp,y,yp);
    [em_x(t),alpha_x(t),beta_x(t)]=SigmaEvaluationX(Sigma);
    [em_y(t),alpha_y(t),beta_y(t)]=SigmaEvaluationY(Sigma);
    SIGMA(:,:,t)=Sigma;
end
%%%Plot Beta function %%%
Z=[0,cumsum(Z_length)];
f1=figure();
plot(Z,beta_x,'*-');
hold on
plot(Z,beta_y,'x--');
xlabel('Z [m]')
ylabel('\beta [m]')
legend('\beta_x','\beta_y')
grid on
hold off
%%%Plot phase spase%%%
disp=0;
disp=input('Choose step to display: ');

while (0<disp) && (disp<16)

        close all
        %x,x'
```

```
        x=XXX(1,:,disp);
        xp=XXX(2,:,disp);
        subplot(2,1,1);
        hold on
        plot(x,xp,'x');
        [x_e,xp_e]=Get_ellipse(em_x(disp),alpha_x(disp),beta_x(disp));
        plot(x_e,xp_e,'r-');
        xlabel("x")
        ylabel("x'")
        grid on
        title(strcat('Phase space at: (',num2str(disp),')'))
        %y,y'
        y=XXX(3,:,disp);
        yp=XXX(4,:,disp);
        subplot(2,1,2)
        hold on
        plot(y,yp,'x');
        [y_e,yp_e]=Get_ellipse(em_y(disp),alpha_y(disp),beta_y(disp));
        plot(y_e,yp_e,'r-');
        grid on
        xlabel("y")
        ylabel("y'")
        disp=input('choose step to display: ');
end
```

## 3.2 Measuring the sigma matrix

```
function sigma=Sigma_measure(x,xp,y,yp)
%%Computes the Sigma matrix of a beam given
%%coordinates of of the partiklas in 4D phase-space.

sigma=zeros(6);
S11=mean(x.^2);
S21=mean(x.*xp);
S22=mean(xp.^2);

S33=mean(y.^2);
S43=mean(y.*yp);
S44=mean(yp.^2);

sigma(1,1)=S11;
sigma(1,2)=-S21;
sigma(2,1)=-S21;
sigma(2,2)=S22;

sigma(3,3)=S33;
sigma(3,4)=-S43;
sigma(4,3)=-S43;
sigma(4,4)=S44;
```

### 3.3 Finding Twiss parameters from Sigma

```
function [em_x,alpha_x,beta_x]=SigmaEvaluationX(Sigma)
%"Measure" Twiss parameters in X-plane of a Sigma matrix
S_x=Sigma(1:2,1:2);

em_x=sqrt(det(S_x));
alpha_x=S_x(1,2)/em_x;
beta_x=S_x(1,1)/em_x;
end

function [em_y,alpha_y,beta_y,Dy]=SigmaEvaluationY(Sigma)
%"Measure" Twiss parameters in Y-plane of a Sigma matrix
S_y=Sigma(3:4,3:4);

em_y=sqrt(det(S_y));
alpha_y=S_y(1,2)/em_y;
beta_y=S_y(1,1)/em_y;
end
```

### 3.4 Drawing phase elipse

```
function [f_x,f_y]=Ellipse_sigma(Sigma);

Sigma_x=Sigma(1:2,1:2);

emmitance_x=sqrt(det(Sigma_x));
beta_x=Sigma_x(1,1)/emmitance_x;
alpha_x=-Sigma_x(2,1)/emmitance_x;
f_x=Draw_ellipse(emmitance_x,alpha_x,beta_x,'x');

Sigma_y=Sigma(3:4,3:4);
emmitance_y=sqrt(det(Sigma_y));
beta_y=Sigma_y(1,1)/emmitance_y;
alpha_y=-Sigma_y(2,1)/emmitance_y;
f_y=Draw_ellipse(emmitance_y,alpha_y,beta_y,'y');
```

# 4 Simulation method 2

```
%Simulate a FODO_line
%Gives phase space and twiss parameters for each step
clear all
close all

```

```
%%%Properties of Matrices%%%
s_qf=0.5;
k_qf=-0.2;

%Dipole
s_dip=0.5;
angle=(15*pi/180)/2; % dipole half bending angle
r_dip=s_dip/angle; % Curvature radius of dipole
%X-Fokusing quadropole 1
s_qf1=s_qf;
k_f1=k_qf;
%X-Fokusing quadropole 2
s_qf2=s_qf;
k_f2=k_qf;
%X-Defocusing quadropole
s_qd=0.5;
k_d=-k_qf;
%Driftspace
s_drift=2;
%Relatevistic effects
gamma2=3.6; %Nummerical value
%%%The Matrices%%%
M_dip=M_dipole(s_dip,r_dip,gamma2);
M_drift=M_Drift(s_drift,gamma2);
M_QD=M_quad(k_d,s_qd,gamma2);
M_QF1=M_quad(k_f1,s_qf1,gamma2);
M_QF2=M_quad(k_f2,s_qf2,gamma2);
%%%The tunnel%%%
% (1)QF/2(3)--(4)D(6)--(7)QD(9)--(10)D(12)--(13)QF/2(15)
N_blocks=12;
Tunnel=zeros(6,6,N_blocks);
Z_length=zeros(1,N_blocks);
%One tunnel block
Tunnel(:,:,1)=M_QF1; %The matrix
Z_length(1)=s_qf; %The length
%
Tunnel(:,:,2)=M_drift;
Z_length(2)=s_drift;
%
Tunnel(:,:,3)=M_dip;
Z_length(3)=s_dip;
%
Tunnel(:,:,4)=M_dip;
Z_length(4)=s_dip;
%
Tunnel(:,:,5)=M_drift;
Z_length(5)=s_drift;
%
Tunnel(:,:,6)=M_QD;
Z_length(6)=s_qd;
%
Tunnel(:,:,7)=M_QD;
Z_length(7)=s_qd;
%
```

```
Tunnel(:,:,8)=M_drift;
Z_length(8)=s_drift;%

Tunnel(:,:,9)=M_dip;
Z_length(9)=s_dip;
%
Tunnel(:,:,10)=M_dip;
Z_length(10)=s_dip;
%
Tunnel(:,:,11)=M_drift;
Z_length(11)=s_drift;
%
Tunnel(:,:,12)=M_QF2;
Z_length(12)=s_qf;
%%%Properties of incoming beam%%%
bx=14.6115;     % (m) standard deviation beam size
gx=1/bx;    % (m) standard deviation beam divergence
by=6.1254;     % (m) standard deviation beam size
gy=1/by;    % (m) standard deviation beam divergence
Sigma1=diag([bx, gx, by, gy, 0, 0]);
DispersionVec=zeros(3,N_blocks);
DispersionVec(3,1)=1;


%%%Array structure with Sigma%%%
SIGMA=zeros(6,6,N_blocks+1);
SIGMA(:,:,1)=Sigma1;
Z=zeros(N_blocks,1); %Vector with length
%%%Stepping sigma%%
Rfodo=eye(6);
for t=1:N_blocks
    Rfodo=Rfodo*Tunnel(:,:,t);
end
D0=Disp_x(Rfodo);
DispersionVec(1,1)=2.8427;



for t=1:N_blocks
    M=Tunnel(:,:,t);
    SIGMA(:,:,t+1)=M*(SIGMA(:,:,t))*M';
    Z(t+1)=sum(Z_length(1:t));
    DispersionVec(:,t+1)=S_dispersion(M)*DispersionVec(:,t); %Eq ...
        84 from Ziemann
    DispersionVec(3,t+1)=1;
end
%%%Finding Twiss parameters%%%
for t=1:(N_blocks+1)
    Sigma=SIGMA(:,:,t);
    [Em_x(t),alpha_x(t),beta_x(t)]=SigmaEvaluationX(Sigma);
    [Em_y(t),alpha_y(t),beta_y(t)]=SigmaEvaluationY(Sigma);
end
%%%Plot Beta function %%%
f1=figure();
```

```
plot(Z,beta_x,'*-');
hold on
plot(Z,beta_y,'x--');
xlabel('Z [m]')
ylabel('\beta [m]')
legend('\beta_x','\beta_y')
grid on
hold off
f2=figure();
plot(Z,DispersionVec(1,:),'*-');
hold on
xlabel('Z [m]')
ylabel('Dispersion [m]')
legend('D_x')
grid on
hold off

disp=0;
%disp=input('Choose step to display: '); %uncomment to plot ...
   phase-ellipse
while (0<disp) && (disp<16)
    close all
    %x,x'
    subplot(2,1,1);
    hold on
    [x_e,xp_e]=Get_ellipse(Em_x(disp),alpha_x(disp),beta_x(disp));
    plot(x_e,xp_e);
    xlabel("x")
    ylabel("x'")
    grid on
    title(strcat('Phase space at: (',num2str(disp),')'))
    %y,y'
    subplot(2,1,2)
    hold on
    [y_e,yp_e]=Get_ellipse(Em_y(disp),alpha_y(disp),beta_y(disp));
    plot(y_e,yp_e);
    grid on
    xlabel("y")
    ylabel("y'")
    disp=input('choose step to display: ');
end
```

# 5 simulation method 3

```
%Simulate a FODO_line using B1=M*B0
%Gives phase space and twiss parameters for each step
clear all
%close all

```

```
%%%Properties of Matrices%%%
%Dipole
sDip=1;
angle=(15*pi/180)/2; % dipole half bending angle
r_dip=0.5*sDip/angle; % Curvature radius of dipole
%Quadropoles
kF1=-0.2; %fokusing quadropole1
kF2=-0.2; %fokusing quadropole2
kD=0.2; %defokusing quadropole
sQ=1;
%Driftspace
sD=2;
%Relatevistic effects
gamma2=3.6; %Nummerical value

%incommingbeam
D0=0;
Dp0=0;
betaX0=14.6115;
alphaX0=0;
gammaX0=(1-alphaX0^2)/betaX0;
betaY0=6.1254;
alphaY0=0;
gammaY0=(1-alphaY0^2)/betaY0;


%%%The Matrices%%%
MdipHalf=M_dipole(sDip/2,r_dip,gamma2);
Mdrift=M_Drift(sD,gamma2);
MqdHalf=M_quad(kD,sQ/2,gamma2);
Mqf1=M_quad(kF1,sQ/2,gamma2);
Mqf2=M_quad(kF2,sQ/2,gamma2);
%%%The tunnel%%%
Nblock=12;
Tunnel=zeros(6,6,Nblock);
Z_length=zeros(1,Nblock+1);
%One tunnel block
Tunnel(:,:,1)=Mqf1; %The matrix
Z_length(2)=sQ/2; %The length
%
Tunnel(:,:,2)=Mdrift;
Z_length(3)=sD;
%
Tunnel(:,:,3)=MdipHalf;
Z_length(4)=sDip/2;
%
Tunnel(:,:,4)=MdipHalf;
Z_length(5)=sDip/2;
%
Tunnel(:,:,5)=Mdrift;
Z_length(6)=sD;
%
Tunnel(:,:,6)=MqdHalf;
Z_length(7)=sQ/2;
```

```
%
Tunnel(:,:,7)=MqdHalf;
Z_length(8)=sQ/2;
%
Tunnel(:,:,8)=Mdrift;
Z_length(9)=sD;%

Tunnel(:,:,9)=MdipHalf;
Z_length(10)=sDip/2;
%
Tunnel(:,:,10)=MdipHalf;
Z_length(11)=sDip/2;
%
Tunnel(:,:,11)=Mdrift;
Z_length(12)=sD;
%
Tunnel(:,:,12)=Mqf2;
Z_length(13)=sQ/2;

Z=cumsum(Z_length);
%%%Properties of incoming beam%%%
BFX=zeros(3,Nblock+1);
BFX(:,1)=[betaX0,alphaX0,gammaX0]';
BFY=zeros(3,Nblock+1);
BFY(:,1)=[betaY0,alphaY0,gammaY0]';
Dvec=zeros(3,Nblock+1);
Dvec=[D0,Dp0,1]';

%%%Stepping sigma%%

for i=1:Nblock
   Mx=M_beta(Tunnel(1:2,1:2,i));
   BFX(:,i+1)=Mx*BFX(:,i);
   My=M_beta(Tunnel(3:4,3:4,i));
   BFY(:,i+1)=My*BFY(:,i);
   S=S_dispersion(Tunnel(:,:,i));
   Dvec(:,i+1)=S*Dvec(:,i);
end
%%%Plot Beta function %%%
f1=figure();
plot(Z,BFX(1,:),'*-');
hold on
plot(Z,BFY(1,:),'x--');
xlabel('Z [m]')
ylabel('\beta [m]')
legend('\beta_x','\beta_y')
grid on
hold off
f2=figure();
plot(Z,Dvec(1,:),'*-');
hold on
xlabel('Z [m]')
ylabel('Dispersion [m]')
legend('D_x')
```

```
114 grid on
```

# 6 Geometry

## 6.1 Plot line

```
1  close all
2  clear all
3
4  rDipole=73.45;
5  sDipole=2;
6  phi=sDipole/rDipole;
7  sDrift=0.3;
8  sQ=0.4;
9  ΔTheta=2*asin(28/(2*100));
10
11 L=2*sQ+4*sDrift+2*sDipole;
12 percD=2*sDipole/L;
13
14 startX=0;
15 startY=0;
16 startTheta=pi+6.5*pi/180;
17
18 X=startX;
19 Y=startY;
20 theta=startTheta;
21
22 figure()
23 grid on
24 hold on
25
26 d=0;
27 %DQ
28 [xNext,yNext]=straight(X(end),Y(end),theta,0.4);
29 X=[X(end);xNext];
30 Y=[Y(end);yNext];
31 plot(X,Y,'g-','LineWidth',1)
32
33 %Drift
34 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
35 X=[X(end);xNext];
36 Y=[Y(end);yNext];
37 plot(X,Y,'k:','LineWidth',1)
38
39 [X,Y,theta]=FODOcell(X,Y,theta,rDipole,1.7,sQ/2,sQ,sDrift);
40
41 for i=[1:35]
42 [X,Y,theta]=FODOcell(X,Y,theta,rDipole,sDipole,sQ/2,sQ,sDrift);
43 %d=sqrt(X(end)^2+Y(end)^2);
```

```
end
[X,Y,theta]=FODOcell(X,Y,theta,rDipole,1.7,sQ/2,sQ,sDrift);
%Drift
[xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'k:','LineWidth',1)

%DQ
[xNext,yNext]=straight(X(end),Y(end),theta,0.4);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'g-','LineWidth',1)
theta
thetap=180*(theta-startTheta)/pi
d=sqrt(X(end)^2+Y(end)^2);
%legend('FQ','Drift','Dipole','Drift','DQ')
xlabel('x [m]')
ylabel('y [m]')

axis equal

function  ...
    [X,Y,theta]=FODOcell(X,Y,theta,rDipole,sDipole,sFQ,sDQ,sDrift)
%FODO-cell
phi=sDipole/rDipole;
%%%%%%
%FQ
[xNext,yNext]=straight(X(end),Y(end),theta,sFQ);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'b-','LineWidth',4)

%Drift
[xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'k:','LineWidth',1)

%Dipole
[xNext,yNext]=curve(X(end),Y(end),theta,rDipole,phi);
X=[xNext];
Y=[yNext];
theta=theta+phi;
plot(X,Y,'r-','LineWidth',2)

%Drift
[xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'k:','LineWidth',1)

%DQ
[xNext,yNext]=straight(X(end),Y(end),theta,sDQ);
```

```
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'g-','LineWidth',1)

%Drift
[xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'k:','LineWidth',1)

%Dipole
[xNext,yNext]=curve(X(end),Y(end),theta,rDipole,phi);
X=[xNext];
Y=[yNext];
theta=theta+phi;
plot(X,Y,'r-','LineWidth',2)

%Drift
[xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'k:','LineWidth',1)

%FQ
[xNext,yNext]=straight(X(end),Y(end),theta,sFQ);
X=[X(end);xNext];
Y=[Y(end);yNext];
plot(X,Y,'b-','LineWidth',4)
%%%%%%%%%%
end

function [xNext,yNext]=straight(x0,y0,theta,s)
dx=s/sqrt(1+tan(theta)^2);
if(cos(theta)<0)
    dx=-abs(dx);
end
dy=s*sin(theta);

xNext=x0+dx;
yNext=y0+dy;
end

function [xNext,yNext]=curve(x0,y0,theta,R,phi)
xC=x0+R*cos(pi/2+theta);
yC=y0+R*sin(pi/2+theta);
x=[];
y=[];
T=linspace(theta-pi/2,theta+phi-pi/2,20);
for t=T
    x=[x;R*cos(t)];
    y=[y;R*sin(t)];
end
xNext=x+xC;
yNext=y+yC;
```

```
151 end
```

## 6.2 Finding geometry

```
1  %geometry
2  clear all
3  close all
4
5  nCell=33;
6  q=2/3;
7  distStraight=185;
8  rho=73.45;
9
10
11 R=rho/q;
12 theta= 2*asin(distStraight/(2*R));
13 sBigCsection=R*theta;
14 phi= theta/(2*nCell);
15 sDip=phi*rho;
16 a= (R-rho)*sin(phi)/(1+cos(phi));
17 Lfodo=3*a+2*sDip;
18 qp=2*sDip/Lfodo;
19 sQ=a/1.5;
20
21 Theta= theta*180/pi;
22
23 angle=2*180*phi/pi
```

# 7 Evaluating design and matching

## 7.1 All parameters

```
1  %GlobalValiables.m
2  %This scrip gives all the design parameters
3  close all
4  clear all
5  %Incomming beam from Linacc
6  global betaXin;
7  global betaYin;
8  global Din;
9  global DPin;
10 global alphaXin;
11 global alphaYin;
12 betaXin=28.8788;
13 alphaXin=0.5193;
```

```
14 betaYin=51.9032;
15 alphaYin=-1.9732;
16 Din=0;
17 DPin=0;
18 %IN--LINE--OUT
19
20 %Matrix parameters
21 global gamma2;
22 gamma2=3.6;
23
24 %IN- magnet parameters for matching section into the line (2 FODO ...
       cells)
25 global sDipIn; %Lenght of Dipole
26 sDipIn=1.7;
27 global r_dipIn; %Bending radius of dipole
28 r_dipIn=73.45;
29 global sQIn; %Length of Quadropoles (QF is half as long)
30 sQIn = 0.4;
31 global sDIn; %Length of drift space
32 sDIn = 0.3;
33
34 %LINE-
35 global sDipL;
36 sDipL=2;
37 global r_dipL;
38 r_dipL=73.45;
39 global kL;
40 kL = 0.5;
41 global sQL;
42 sQL = 0.4;
43 global sDL;
44 sDL = 0.3;
45 global ems;
46 ems = 0.35*10^(-6)/(20*sqrt(gamma2));
```

## 7.2 Finding symmetric functions

```
1 %FindSymmetry.m
2 %Find initial values for symmetric and periodic beta functions.
3 clear variables
4 close all
5
6
7 %%%Properties of Matrices%%%
8 global sDipL;
9 global r_dipL;
10 global kL;
11 global sQL;
12 global sDL;
13 global gamma2;
```

```
14 global ems;
15 %%%The Matrices%%%
16 MdipHalf=M_dipole(sDipL/2,r_dipL,gamma2);
17 Mdrift=M_Drift(sDL,gamma2);
18 MqdHalf=M_quad(kL,sQL/2,gamma2);
19 Mqf=M_quad(-kL,sQL/2,gamma2);
20 %%%The tunnel%%%
21 Nblock=12;
22 Tunnel=zeros(6,6,Nblock);
23 Z_length=zeros(1,Nblock+1);
24 %One tunnel block
25 Tunnel(:,:,1)=Mqf; %The matrix
26 Z_length(2)=sQL/2; %The length
27 %
28 Tunnel(:,:,2)=Mdrift;
29 Z_length(3)=sDL;
30 %
31 Tunnel(:,:,3)=MdipHalf;
32 Z_length(4)=sDipL/2;
33 %
34 Tunnel(:,:,4)=MdipHalf;
35 Z_length(5)=sDipL/2;
36 %
37 Tunnel(:,:,5)=Mdrift;
38 Z_length(6)=sDL;
39 %
40 Tunnel(:,:,6)=MqdHalf;
41 Z_length(7)=sQL/2;
42 %
43 Tunnel(:,:,7)=MqdHalf;
44 Z_length(8)=sQL/2;
45 %
46 Tunnel(:,:,8)=Mdrift;
47 Z_length(9)=sDL;%
48
49 Tunnel(:,:,9)=MdipHalf;
50 Z_length(10)=sDipL/2;
51 %
52 Tunnel(:,:,10)=MdipHalf;
53 Z_length(11)=sDipL/2;
54 %
55 Tunnel(:,:,11)=Mdrift;
56 Z_length(12)=sDL;
57 %
58 Tunnel(:,:,12)=Mqf;
59 Z_length(13)=sQL/2;
60
61 Z=cumsum(Z_length);
62 %%%Properties of incoming beam%%%
63 R=Mqf*Mdrift*(MdipHalf)^2*Mdrift*MqdHalf^2*Mdrift*
64 MdipHalf^2*Mdrift*Mqf
65 global betaXL;
66 global betaYL;
67 global DL;
```

```
68 betaXL = sqrt( -R(1,2)*R(2,2)/ (R(2,1) * R(1,1)) );
69 betaYL = sqrt( -R(3,4)*R(4,4)/ (R(4,3) * R(3,3)) );
70 S=S_dispersion(R);
71 DL = -R(2,6)/R(2,1);
72 BFX=zeros(3,Nblock+1);
73 BFX(:,1)=[betaXL,0,1/betaXL]';
74 BFY=zeros(3,Nblock+1);
75 BFY(:,1)=[betaYL,0,1/betaYL]';
76 Dvec=zeros(3,Nblock+1);
77 Dvec(:,1)=[DL,0,1]';
78
79 %%%Stepping sigma%%
80
81 for i=1:Nblock
82    Mx=M_beta(Tunnel(1:2,1:2,i)); %Get the 3x3 matrix
83    BFX(:,i+1)=Mx*BFX(:,i);
84    My=M_beta(Tunnel(3:4,3:4,i));
85    BFY(:,i+1)=My*BFY(:,i);
86    S=S_dispersion(Tunnel(:,:,i));
87    Dvec(:,i+1)=S*Dvec(:,i);
88 end
89 sigmax=beamSize(0.02,BFX(1,:),Dvec(1,:));
90 sigmay=beamSize(0.02,BFY(1,:),0);
91 %%%Plot Beta function %%%
92 f1=figure();
93 subplot(1,2,1)
94 plot(Z,BFX(1,:),'*-');
95 hold on
96 plot(Z,BFY(1,:),'x--');
97 xlabel('Z [m]')
98 ylabel('\beta [m]')
99 legend('\beta_x','\beta_y')
100 grid on
101 hold off
102 subplot(1,2,2);
103 plot(Z,Dvec(1,:),'*-');
104 hold on
105 %plot(Z,sigmax,'+--');
106 %plot(Z,sigmay,'x:');
107 xlabel('Z [m]')
108 ylabel('Dispersion [m]')
109 %legend('D_x','Size_x','Size_y')
110 legend('D_x')
111 grid on
112
113 betaXL
114 betaYL
115 DL
```

## 7.3 Fit matching cells

```
%StartSectionTuning.m
    %Tune two FODO cells to mach incomming beam to middle line
clear variables
close all

global betaXin;
global betaYin;
global Din;
global DPin;
global alphaXin;
global alphaYin;
%From LinAcc

%ideal values
global betaXL;
global betaYL;
global DL;
Fideal= [betaXL; %betaX
    betaYL; %betaY
    0; %alphaX
    0]; %alphaY
%initial guess
X=[ 1;   %FQ1
    -0.2;   %DQ1
    0.5; %FQ2
    -0.2];  %FQ3
dx=0.0001;
mu=1;
accepance=0.0001;
run=0;
while (mu>accepance)
    F_x1=(F([X(1)+dx;X(2);X(3);X(4)])- ...
        F([X(1)-dx;X(2);X(3);X(4)]))./(2*dx);
    F_x2=(F([X(1);X(2)+dx;X(3);X(4)])- ...
        F([X(1);X(2)-dx;X(3);X(4)]))./(2*dx);
    F_x3=(F([X(1);X(2);X(3)+dx;X(4)])- ...
        F([X(1);X(2);X(3)-dx;X(4)]))./(2*dx);
    F_x4=(F([X(1);X(2);X(3);X(4)+dx])- ...
        F([X(1);X(2);X(3);X(4)-dx]))./(2*dx);
    J=[F_x1,F_x2,F_x3,F_x4];
    %JpI= inv(J'*J)*J';
    Fa=F(X);
    Fdiff=Fideal-Fa;
    Xnext=X+J\Fdiff;
    mu=abs(norm(X-Xnext));
    X=Xnext;
    run=run+1;
end
X
k1=X(1);
k2=X(2);
k3=X(3);
k4=X(4);
```

```
50 global sDipIn;
51 global r_dipIn;
52 global sQIn;
53 global sDIn;
54 global gamma2;
55 %Driftspace
56 %incommingbeam
57
58 gammaX0=(1-alphaXin^2)/betaXin;
59 gammaY0=(1-alphaYin^2)/betaYin;
60 %%%The Matrices%%%
61 %%%The Matrices%%%
62 MdipIn=M_dipole(sDipIn/2,r_dipIn,gamma2);
63 MdriftIn=M_Drift(sDIn,gamma2);
64 Mq1=M_quad(k1,sQIn/2,gamma2);
65 Mq2=M_quad(k2,sQIn/2,gamma2);
66 Mq3=M_quad(k3,sQIn/2,gamma2);
67 Mq4=M_quad(k4,sQIn/2,gamma2);
68 %%%The tunnel%%%
69 %%%The tunnel%%%
70 %
71 %%%Properties of incoming beam%%%
72 BFX0=[betaXin,alphaXin,gammaX0]';
73 BFY0=[betaYin,alphaYin,gammaY0]';
74 Dvec0=[Din,DPin,1]';
75 %
76 %
77 % %
78 %
79 %
80 Nblock=15;
81 Tunnel=zeros(6,6,Nblock);
82 Z_length=zeros(1,Nblock);
83 %One tunnel block
84 Tunnel(:,:,1)=Mq1; %The matrix
85 Z_length(1)=sQIn/2; %The length
86 %
87 Tunnel(:,:,2)=Mq1;
88 Z_length(2)=sQIn/2;
89 %
90 Tunnel(:,:,3)=MdriftIn;
91 Z_length(3)=sDIn;
92 %
93 Tunnel(:,:,4)=Mq2;
94 Z_length(4)=sQIn/2;
95 %
96 Tunnel(:,:,5)=MdriftIn;
97 Z_length(5)=sDIn;
98 %
99 Tunnel(:,:,6)=MdipIn;
100 Z_length(6)=sDipIn/2;
101 %
102 Tunnel(:,:,7)=MdipIn;
103 Z_length(7)=sDIn/2;
```

```
%
Tunnel(:,:,8)=MdriftIn;
Z_length(8)=sDIn;
%
Tunnel(:,:,9)=Mq3;
Z_length(9)=sQIn/2;
%
Tunnel(:,:,10)=Mq3;
Z_length(10)=sQIn/2;
%
Tunnel(:,:,11)=MdriftIn;
Z_length(11)=sDIn;
%
Tunnel(:,:,12)=MdipIn;
Z_length(12)=sDipIn/2;%

Tunnel(:,:,13)=MdipIn;
Z_length(13)=sDipIn/2;
%
Tunnel(:,:,14)=MdriftIn; %The matrix
Z_length(14)=sDIn; %The length
%
Tunnel(:,:,15)=Mq4;
Z_length(15)=sQIn/2;
%

Z=cumsum([0,Z_length]);
BFX=zeros(3,Nblock+1);
BFX(:,1)=BFX0;
BFY=zeros(3,Nblock+1);
BFY(:,1)=BFY0;
Dvec=zeros(3,Nblock+1);
Dvec(:,1)=Dvec0;

for i=1:Nblock
   Mx=M_beta(Tunnel(1:2,1:2,i)); %Get the 3x3 matrix
   BFX(:,i+1)=Mx*BFX(:,i);
   My=M_beta(Tunnel(3:4,3:4,i));
   BFY(:,i+1)=My*BFY(:,i);
   S=S_dispersion(Tunnel(:,:,i));
   Dvec(:,i+1)=S*Dvec(:,i);
end
sigmax=beamSize(0.02,BFX(1,:),Dvec(1,:));
sigmay=beamSize(0.02,BFY(1,:),0);
%%%Plot Beta function %%%
subplot(1,2,1)
plot(Z,BFX(1,:),'*-');
hold on
plot(Z,BFY(1,:),'x--');
xlabel('Z [m]')
ylabel('\beta [m]')
legend('\beta_x','\beta_y')
grid on
subplot(1,2,2)
```

```
158
159 plot(Z,Dvec(1,:),'*-');
160 hold on
161 % plot(Z,sigmax,'+--');
162 % plot(Z,sigmay,'x:');
163 xlabel('Z [m]')
164 ylabel('Dispersion [m]')
165 %legend('D_x','Size_x','Size_y')
166 legend('D_x')
167 grid on
168 function Ftwiss=F(param)
169 %%%Properties of Matrices%%%
170 global sDipIn;
171 global r_dipIn;
172 global sQIn;
173 global sDIn;
174 global gamma2;
175 %Quadropoles
176 k1=param(1);
177 k2=param(2);
178 k3=param(3);
179 k4=param(4);
180 %Driftspace
181 %incommingbeam
182 global betaXin;
183 global betaYin;
184 global Din;
185 global DPin;
186 global alphaXin;
187 global alphaYin;
188
189 gammaX0=(1-alphaXin^2)/betaXin;
190 gammaY0=(1-alphaYin^2)/betaYin;
191
192
193 %%%The Matrices%%%
194 MdipIn=M_dipole(sDipIn/2,r_dipIn,gamma2);
195 MdriftIn=M_Drift(sDIn,gamma2);
196 Mq1=M_quad(k1,sQIn/2,gamma2);
197 Mq2=M_quad(k2,sQIn/2,gamma2);
198 Mq3=M_quad(k3,sQIn/2,gamma2);
199 Mq4=M_quad(k4,sQIn/2,gamma2);
200 %%%The tunnel%%%
201 R=Mq4*MdriftIn*MdipIn*MdipIn*MdriftIn*Mq3*
202 Mq3*MdriftIn*MdipIn*MdipIn*MdriftIn*Mq2*MdriftIn*
203 Mq1*Mq1;
204
205 %%%Properties of incoming beam%%%
206 BFX0=[betaXin,alphaXin,gammaX0]';
207 BFY0=[betaYin,alphaYin,gammaY0]';
208 Dvec0=[Din,DPin,1]';
209
210
211 Mx=M_beta(R(1:2,1:2));
```

```
212 BFXend=Mx*BFX0;
213 My=M_beta(R(3:4,3:4));
214 BFYend=My*BFY0;
215 S=S_dispersion(R);
216 DvecEnd=S*Dvec0;
217 %
218 betaXend=BFXend(1);
219 betaYend=BFYend(1);
220 Dend=DvecEnd(1);
221 alphaXend=BFXend(2);
222 alphaYend=BFYend(2);
223 DpEnd=DvecEnd(2);
224
225 Ftwiss = [betaXend;
226     betaYend;
227     alphaXend;
228     alphaYend];
229 end
```