

# Appendix 1

Karl-Fredrik Boholm Kylesten

May 2019

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Transfer matrices</b>	<b>2</b>
2.1	Drift space . . . . .	2
2.2	Dipole magnet . . . . .	2
2.3	Quadrupole magnet . . . . .	2
<b>3</b>	<b>Simulation method 1</b>	<b>3</b>
3.1	The simulation . . . . .	3
3.2	Measuring the sigma matrix . . . . .	6
3.3	Finding Twiss parameters from Sigma . . . . .	7
3.4	Drawing phase ellipse . . . . .	7
<b>4</b>	<b>Simulation method 2</b>	<b>7</b>
<b>5</b>	<b>simulation method 3</b>	<b>10</b>
<b>6</b>	<b>Geometry</b>	<b>13</b>
6.1	Plot line . . . . .	13
6.2	Finding geometry . . . . .	16
<b>7</b>	<b>Evaluating design and matching</b>	<b>16</b>
7.1	All parameters . . . . .	16
7.2	Finding symmetric functions . . . . .	17
7.3	Fit matching cells . . . . .	19

## 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

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

### 2.2 Dipole magnet

```
1 function M=M_dipole(s,R,gamma2)
2 %Transfer matrix for a dipole magnet
3
4 M=eye(6);
5 C=cos(s/R);
6 S=sin(s/R);
7 M(1,1)=C;
8 M(2,1)=-S/R;
9 M(1,2)=R*S;
10 M(2,2)=C;
11 M(3,4)=s;
12 M(1,6) = R*(1-C);
13 M(2,6) = S;
14 M(5,1) = -S;
15 M(5,2) = -R*(1-C);
16 M(5,6) = s/gamma2;
17 end
```

### 2.3 Quadrupole magnet

```
1 function M=M_quad(k,s,gamma2)
2 %R_quad_long calculate Transfer matrix for a quadrupol magnet
3 %
4 %R_quad_long(k,L) calculate Transfer matrix for a quadrupol ...
   magnet with
5 %magnetfield gradiet k and length s;
6 M=eye(6);
7 k_rot=sqrt(abs(k));
8 omega=s*k_rot;
```

```

9 C=cos(omega);
10 CH=cosh(omega);
11 S=sin(omega);
12 SH=sinh(omega);
13 if (k<=0)
14     M(1,1)=C;
15     M(1,2)=S/k_rot;
16     M(2,1)=-k_rot*S;
17     M(2,2)=C;
18     M(3,3)=CH;
19     M(3,4)=SH/k_rot;
20     M(4,3)=k_rot*SH;
21     M(4,4)=CH;
22 else
23     M(1,1)=CH;
24     M(1,2)=SH/k_rot;
25     M(2,1)=k_rot*SH;
26     M(2,2)=CH;
27     M(3,3)=C;
28     M(3,4)=S/k_rot;
29     M(4,3)=-k_rot*S;
30     M(4,4)=C;
31 end
32 M(5,6)=s/gamma2;
33 end

```

## 3 Simulationsmethod 1

### 3.1 The simulation

```

1 %Simulate a FODO block
2 %Gives phase space and twiss parameters for each step
3 close all
4 clear all
5 %%Properties of Matrices%%
6 s_qf=0.5;
7 k_qf=-0.2;
8
9 %Dipole
10 s_dip=0.5;
11 angle=(15*pi/180)/2; % dipole half bending angle
12 r_dip=s_dip/angle; % Curvature radius of dipole
13 %X-Focusing quadropole 1
14 s_qf1=s_qf;
15 k_f1=k_qf;
16 %X-Focusing quadropole 2
17 s_qf2=s_qf;
18 k_f2=k_qf;
19 %X-Defocusing quadropole

```

```

20 s_qd=0.5;
21 k_d=-k_qf;
22 %Driftspace
23 s_drift=2;
24 %Relativistic effects
25 gamma2=3.6; %Numerical value
26 %%%The Matrices%%
27 M_dip=M_dipole(s_dip,r_dip,gamma2);
28 M_drift=M_Drift(s_drift,gamma2);
29 M_QD=M_quad(k_d,s_qd,gamma2);
30 M_QF1=M_quad(k_f1,s_qf1,gamma2);
31 M_QF2=M_quad(k_f2,s_qf2,gamma2);
32 %%%The tunnel%%
33 N_blocks=12;
34 Tunnel=zeros(6,6,N_blocks);
35 Z_length=zeros(1,N_blocks);
36 %One tunnel block
37 Tunnel(:, :, 1)=M_QF1; %The matrix
38 Z_length(1)=s_qf; %The length
39 %
40 Tunnel(:, :, 2)=M_drift;
41 Z_length(2)=s_drift;
42 %
43 Tunnel(:, :, 3)=M_dip;
44 Z_length(3)=s_dip;
45 %
46 Tunnel(:, :, 4)=M_dip;
47 Z_length(4)=s_dip;
48 %
49 Tunnel(:, :, 5)=M_drift;
50 Z_length(5)=s_drift;
51 %
52 Tunnel(:, :, 6)=M_QD;
53 Z_length(6)=s_qd;
54 %
55 Tunnel(:, :, 7)=M_QD;
56 Z_length(7)=s_qd;
57 %
58 Tunnel(:, :, 8)=M_drift;
59 Z_length(8)=s_drift;%
60
61 Tunnel(:, :, 9)=M_dip;
62 Z_length(9)=s_dip;
63 %
64 Tunnel(:, :, 10)=M_dip;
65 Z_length(10)=s_dip;
66 %
67 Tunnel(:, :, 11)=M_drift;
68 Z_length(11)=s_drift;
69 %
70 Tunnel(:, :, 12)=M_QF2;
71 Z_length(12)=s_qf;
72 %%%Properties of incoming beam%%
73 bx=14.36; % (m) standard deviation beam size

```

```

74 gx=1/bx;      % (m) standard deviation beam divergence
75 by=5.67;     % (m) standard deviation beam size
76 gy=1/by;     % (m) standard deviation beam divergence
77 N=10000;    %Number of particles in simulation
78 %%%Array structure with particles (dimension,#particle,step)%%
79 XXX=zeros(6,N,N_blocks+1);
80 Z=zeros(N_blocks+1,1); %Vector with length
81 %%%Moving the partikles%%
82 for p=1:N
83     XXX(1,p,1)=sqrt(bx)*randn(1,1);
84     XXX(2,p,1)=sqrt(gx)*randn(1,1);
85     XXX(3,p,1)=sqrt(by)*randn(1,1);
86     XXX(4,p,1)=sqrt(gy)*randn(1,1);
87 for t=2:(N_blocks+1)
88     XXX(:,p,t)=Tunnel(:, :, t-1)*XXX(:,p, (t-1));
89 end
90 end
91 %%%Measuring em, alpha, beta %%%
92 em_x=zeros(1,N_blocks+1);
93 em_y=zeros(1,N_blocks+1);
94 alpha_x=zeros(1,N_blocks+1);
95 alpha_y=zeros(1,N_blocks+1);
96 beta_x=zeros(1,N_blocks+1);
97 beta_y=zeros(1,N_blocks+1);
98 SIGMA=zeros(6,6,N_blocks+1);
99 for t=1:(N_blocks+1)
100     x=XXX(1, :, t);
101     xp=XXX(2, :, t);
102     y=XXX(3, :, t);
103     yp=XXX(4, :, t);
104     Sigma=Sigma_measure(x, xp, y, yp);
105     [em_x(t), alpha_x(t), beta_x(t)]=SigmaEvaluationX(Sigma);
106     [em_y(t), alpha_y(t), beta_y(t)]=SigmaEvaluationY(Sigma);
107     SIGMA(:, :, t)=Sigma;
108 end
109 %%%Plot Beta function %%%
110 Z=[0, cumsum(Z_length)];
111 f1=figure();
112 plot(Z, beta_x, '*-');
113 hold on
114 plot(Z, beta_y, 'x-');
115 xlabel('Z [m]')
116 ylabel('\beta [m]')
117 legend('\beta_x', '\beta_y')
118 grid on
119 hold off
120 %%%Plot phase spase%%
121 disp=0;
122 disp=input('Choose step to display: ');
123
124 while (0<disp) && (disp<16)
125
126     close all
127     %x, x'

```

```

128     x=XXX(1, :, disp);
129     xp=XXX(2, :, disp);
130     subplot(2,1,1);
131     hold on
132     plot(x, xp, 'x');
133     [x_e, xp_e]=Get_ellipse(em_x(disp), alpha_x(disp), beta_x(disp));
134     plot(x_e, xp_e, 'r-');
135     xlabel("x")
136     ylabel("x'")
137     grid on
138     title(strcat('Phase space at: (' , num2str(disp), ') '))
139     %y, y'
140     y=XXX(3, :, disp);
141     yp=XXX(4, :, disp);
142     subplot(2,1,2)
143     hold on
144     plot(y, yp, 'x');
145     [y_e, yp_e]=Get_ellipse(em_y(disp), alpha_y(disp), beta_y(disp));
146     plot(y_e, yp_e, 'r-');
147     grid on
148     xlabel("y")
149     ylabel("y'")
150     disp=input('choose step to display: ');
151 end

```

## 3.2 Measuring the sigma matrix

```

1 function sigma=Sigma_measure(x, xp, y, yp)
2 %%Computes the Sigma matrix of a beam given
3 %%coordinates of of the partiklas in 4D phase-space.
4
5 sigma=zeros(6);
6 S11=mean(x.^2);
7 S21=mean(x.*xp);
8 S22=mean(xp.^2);
9
10 S33=mean(y.^2);
11 S43=mean(y.*yp);
12 S44=mean(yp.^2);
13
14 sigma(1,1)=S11;
15 sigma(1,2)=-S21;
16 sigma(2,1)=-S21;
17 sigma(2,2)=S22;
18
19 sigma(3,3)=S33;
20 sigma(3,4)=-S43;
21 sigma(4,3)=-S43;
22 sigma(4,4)=S44;

```

### 3.3 Finding Twiss parameters from Sigma

```
1 function [em_x,alpha_x,beta_x]=SigmaEvaluationX(Sigma)
2 %"Measure" Twiss parameters in X-plane of a Sigma matrix
3 S_x=Sigma(1:2,1:2);
4
5 em_x=sqrt(det(S_x));
6 alpha_x=S_x(1,2)/em_x;
7 beta_x=S_x(1,1)/em_x;
8 end
9
10 function [em_y,alpha_y,beta_y,Dy]=SigmaEvaluationY(Sigma)
11 %"Measure" Twiss parameters in Y-plane of a Sigma matrix
12 S_y=Sigma(3:4,3:4);
13
14 em_y=sqrt(det(S_y));
15 alpha_y=S_y(1,2)/em_y;
16 beta_y=S_y(1,1)/em_y;
17 end
```

### 3.4 Drawing phase ellipse

```
1 function [f_x,f_y]=Ellipse_sigma(Sigma);
2
3 Sigma_x=Sigma(1:2,1:2);
4
5 emittance_x=sqrt(det(Sigma_x));
6 beta_x=Sigma_x(1,1)/emittance_x;
7 alpha_x=-Sigma_x(2,1)/emittance_x;
8 f_x=Draw_ellipse(emittance_x,alpha_x,beta_x,'x');
9
10 Sigma_y=Sigma(3:4,3:4);
11 emittance_y=sqrt(det(Sigma_y));
12 beta_y=Sigma_y(1,1)/emittance_y;
13 alpha_y=-Sigma_y(2,1)/emittance_y;
14 f_y=Draw_ellipse(emittance_y,alpha_y,beta_y,'y');
```

## 4 Simulation method 2

```
1 %Simulate a FODO_line
2 %Gives phase space and twiss parameters for each step
3 clear all
4 close all
5
```

```

6  %%Properties of Matrices%%
7  s_qf=0.5;
8  k_qf=-0.2;
9
10 %Dipole
11 s_dip=0.5;
12 angle=(15*pi/180)/2; % dipole half bending angle
13 r_dip=s_dip/angle; % Curvature radius of dipole
14 %X-Focusing quadropole 1
15 s_qf1=s_qf;
16 k_f1=k_qf;
17 %X-Focusing quadropole 2
18 s_qf2=s_qf;
19 k_f2=k_qf;
20 %X-Defocusing quadropole
21 s_qd=0.5;
22 k_d=-k_qf;
23 %Driftspace
24 s_drift=2;
25 %Relativistic effects
26 gamma2=3.6; %Numerical value
27 %%The Matrices%%
28 M_dip=M_dipole(s_dip,r_dip,gamma2);
29 M_drift=M_Drift(s_drift,gamma2);
30 M_QD=M_quad(k_d,s_qd,gamma2);
31 M_QF1=M_quad(k_f1,s_qf1,gamma2);
32 M_QF2=M_quad(k_f2,s_qf2,gamma2);
33 %%The tunnel%%
34 % (1)QF/2 (3)---(4)D (6)---(7)QD (9)---(10)D (12)---(13)QF/2 (15)
35 N_blocks=12;
36 Tunnel=zeros(6,6,N_blocks);
37 Z_length=zeros(1,N_blocks);
38 %One tunnel block
39 Tunnel(:, :, 1)=M_QF1; %The matrix
40 Z_length(1)=s_qf; %The length
41 %
42 Tunnel(:, :, 2)=M_drift;
43 Z_length(2)=s_drift;
44 %
45 Tunnel(:, :, 3)=M_dip;
46 Z_length(3)=s_dip;
47 %
48 Tunnel(:, :, 4)=M_dip;
49 Z_length(4)=s_dip;
50 %
51 Tunnel(:, :, 5)=M_drift;
52 Z_length(5)=s_drift;
53 %
54 Tunnel(:, :, 6)=M_QD;
55 Z_length(6)=s_qd;
56 %
57 Tunnel(:, :, 7)=M_QD;
58 Z_length(7)=s_qd;
59 %

```



```

60 Tunnel(:, :, 8)=M_drift;
61 Z_length(8)=s_drift;%
62
63 Tunnel(:, :, 9)=M_dip;
64 Z_length(9)=s_dip;
65 %
66 Tunnel(:, :, 10)=M_dip;
67 Z_length(10)=s_dip;
68 %
69 Tunnel(:, :, 11)=M_drift;
70 Z_length(11)=s_drift;
71 %
72 Tunnel(:, :, 12)=M_QF2;
73 Z_length(12)=s_qf;
74 %%%Properties of incoming beam%%
75 bx=14.6115; % (m) standard deviation beam size
76 gx=1/bx; % (m) standard deviation beam divergence
77 by=6.1254; % (m) standard deviation beam size
78 gy=1/by; % (m) standard deviation beam divergence
79 Sigma=diag([bx, gx, by, gy, 0, 0]);
80 DispersionVec=zeros(3,N_blocks);
81 DispersionVec(3,1)=1;
82
83
84 %%%Array structure with Sigma%%
85 SIGMA=zeros(6,6,N_blocks+1);
86 SIGMA(:, :, 1)=Sigma;
87 Z=zeros(N_blocks,1); %Vector with length
88 %%%Stepping sigma%%
89 Rfodo=eye(6);
90 for t=1:N_blocks
91     Rfodo=Rfodo*Tunnel(:, :, t);
92 end
93 D0=Disp_x(Rfodo);
94 DispersionVec(1,1)=2.8427;
95
96
97
98 for t=1:N_blocks
99     M=Tunnel(:, :, t);
100     SIGMA(:, :, t+1)=M*(SIGMA(:, :, t))*M';
101     Z(t+1)=sum(Z_length(1:t));
102     DispersionVec(:, t+1)=S_dispersion(M)*DispersionVec(:, t); %Eq ...
        84 from Ziemann
103     DispersionVec(3, t+1)=1;
104 end
105 %%%Finding Twiss parameters%%
106 for t=1:(N_blocks+1)
107     Sigma=SIGMA(:, :, t);
108     [Em_x(t), alpha_x(t), beta_x(t)]=SigmaEvaluationX(Sigma);
109     [Em_y(t), alpha_y(t), beta_y(t)]=SigmaEvaluationY(Sigma);
110 end
111 %%%Plot Beta function %%
112 f1=figure();

```

```

113 plot(Z,beta_x,'*-');
114 hold on
115 plot(Z,beta_y,'x—');
116 xlabel('Z [m]')
117 ylabel('\beta [m]')
118 legend('\beta_x','\beta_y')
119 grid on
120 hold off
121 f2=figure();
122 plot(Z,DispersionVec(1,:), '*-');
123 hold on
124 xlabel('Z [m]')
125 ylabel('Dispersion [m]')
126 legend('D_x')
127 grid on
128 hold off
129
130 disp=0;
131 %disp=input('Choose step to display: '); %uncomment to plot ...
    phase-ellipse
132 while (0<disp) && (disp<16)
133     close all
134     %x,x'
135     subplot(2,1,1);
136     hold on
137     [x_e, xp_e]=Get_ellipse(Em_x(disp), alpha_x(disp), beta_x(disp));
138     plot(x_e, xp_e);
139     xlabel("x")
140     ylabel("x'")
141     grid on
142     title(strcat('Phase space at: (' , num2str(disp), ')'))
143     %y,y'
144     subplot(2,1,2)
145     hold on
146     [y_e, yp_e]=Get_ellipse(Em_y(disp), alpha_y(disp), beta_y(disp));
147     plot(y_e, yp_e);
148     grid on
149     xlabel("y")
150     ylabel("y'")
151     disp=input('choose step to display: ');
152 end

```

## 5 simulation method 3

```

1 %Simulate a FODO_line using B1=M*B0
2 %Gives phase space and twiss parameters for each step
3 clear all
4 %close all
5

```

```

6  %%Properties of Matrices%%
7  %Dipole
8  sDip=1;
9  angle=(15*pi/180)/2; % dipole half bending angle
10 r_dip=0.5*sDip/angle; % Curvature radius of dipole
11 %Quadropoles
12 kF1=-0.2; %fokusing quadropole1
13 kF2=-0.2; %fokusing quadropole2
14 kD=0.2; %defokusing quadropole
15 sQ=1;
16 %Driftspace
17 sD=2;
18 %Relativistic effects
19 gamma2=3.6; %Numerical value
20
21 %incommingbeam
22 D0=0;
23 Dp0=0;
24 betaX0=14.6115;
25 alphaX0=0;
26 gammaX0=(1-alphaX0^2)/betaX0;
27 betaY0=6.1254;
28 alphaY0=0;
29 gammaY0=(1-alphaY0^2)/betaY0;
30
31
32 %%The Matrices%%
33 MdipHalf=M_dipole(sDip/2,r_dip,gamma2);
34 Mdrift=M_Drift(sD,gamma2);
35 MqdHalf=M_quad(kD,sQ/2,gamma2);
36 Mqf1=M_quad(kF1,sQ/2,gamma2);
37 Mqf2=M_quad(kF2,sQ/2,gamma2);
38 %%The tunnel%%
39 Nblock=12;
40 Tunnel=zeros(6,6,Nblock);
41 Z_length=zeros(1,Nblock+1);
42 %One tunnel block
43 Tunnel(:, :, 1)=Mqf1; %The matrix
44 Z_length(2)=sQ/2; %The length
45 %
46 Tunnel(:, :, 2)=Mdrift;
47 Z_length(3)=sD;
48 %
49 Tunnel(:, :, 3)=MdipHalf;
50 Z_length(4)=sDip/2;
51 %
52 Tunnel(:, :, 4)=MdipHalf;
53 Z_length(5)=sDip/2;
54 %
55 Tunnel(:, :, 5)=Mdrift;
56 Z_length(6)=sD;
57 %
58 Tunnel(:, :, 6)=MqdHalf;
59 Z_length(7)=sQ/2;

```

```

60 %
61 Tunnel(:, :, 7) = MqdHalf;
62 Z_length(8) = sQ/2;
63 %
64 Tunnel(:, :, 8) = Mdrift;
65 Z_length(9) = sD; %
66
67 Tunnel(:, :, 9) = MdipHalf;
68 Z_length(10) = sDip/2;
69 %
70 Tunnel(:, :, 10) = MdipHalf;
71 Z_length(11) = sDip/2;
72 %
73 Tunnel(:, :, 11) = Mdrift;
74 Z_length(12) = sD;
75 %
76 Tunnel(:, :, 12) = Mqf2;
77 Z_length(13) = sQ/2;
78
79 Z = cumsum(Z_length);
80 %%%Properties of incoming beam%%
81 BFX = zeros(3, Nblock+1);
82 BFX(:, 1) = [betaX0, alphaX0, gammaX0]';
83 BFY = zeros(3, Nblock+1);
84 BFY(:, 1) = [betaY0, alphaY0, gammaY0]';
85 Dvec = zeros(3, Nblock+1);
86 Dvec = [D0, Dp0, 1]';
87
88 %%%Stepping sigma%%
89
90 for i = 1:Nblock
91     Mx = M_beta(Tunnel(1:2, 1:2, i));
92     BFX(:, i+1) = Mx*BFX(:, i);
93     My = M_beta(Tunnel(3:4, 3:4, i));
94     BFY(:, i+1) = My*BFY(:, i);
95     S = S_dispersion(Tunnel(:, :, i));
96     Dvec(:, i+1) = S*Dvec(:, i);
97 end
98 %%%Plot Beta function %%
99 f1 = figure();
100 plot(Z, BFX(1, :), '*-');
101 hold on
102 plot(Z, BFY(1, :), 'x-');
103 xlabel('Z [m]')
104 ylabel('\beta [m]')
105 legend('\beta_x', '\beta_y')
106 grid on
107 hold off
108 f2 = figure();
109 plot(Z, Dvec(1, :), '*-');
110 hold on
111 xlabel('Z [m]')
112 ylabel('Dispersion [m]')
113 legend('D_x')

```

## 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);

```

```

44 end
45 [X,Y,theta]=FODOcell(X,Y,theta,rDipole,1.7,sQ/2,sQ,sDrift);
46 %Drift
47 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
48 X=[X(end);xNext];
49 Y=[Y(end);yNext];
50 plot(X,Y,'k','LineWidth',1)
51
52 %DQ
53 [xNext,yNext]=straight(X(end),Y(end),theta,0.4);
54 X=[X(end);xNext];
55 Y=[Y(end);yNext];
56 plot(X,Y,'g-','LineWidth',1)
57 theta
58 thetap=180*(theta-startTheta)/pi
59 d=sqrt(X(end)^2+Y(end)^2);
60 %legend('FQ','Drift','Dipole','Drift','DQ')
61 xlabel('x [m]')
62 ylabel('y [m]')
63
64 axis equal
65
66 function ...
        [X,Y,theta]=FODOcell(X,Y,theta,rDipole,sDipole,sFQ,sDQ,sDrift)
67 %FODO-cell
68 phi=sDipole/rDipole;
69 %%%
70 %FQ
71 [xNext,yNext]=straight(X(end),Y(end),theta,sFQ);
72 X=[X(end);xNext];
73 Y=[Y(end);yNext];
74 plot(X,Y,'b-','LineWidth',4)
75
76 %Drift
77 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
78 X=[X(end);xNext];
79 Y=[Y(end);yNext];
80 plot(X,Y,'k','LineWidth',1)
81
82 %Dipole
83 [xNext,yNext]=curve(X(end),Y(end),theta,rDipole,phi);
84 X=[xNext];
85 Y=[yNext];
86 theta=theta+phi;
87 plot(X,Y,'r-','LineWidth',2)
88
89 %Drift
90 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
91 X=[X(end);xNext];
92 Y=[Y(end);yNext];
93 plot(X,Y,'k','LineWidth',1)
94
95 %DQ
96 [xNext,yNext]=straight(X(end),Y(end),theta,sDQ);

```

```

97 X=[X(end);xNext];
98 Y=[Y(end);yNext];
99 plot(X,Y,'g-','LineWidth',1)
100
101 %Drift
102 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
103 X=[X(end);xNext];
104 Y=[Y(end);yNext];
105 plot(X,Y,'k:', 'LineWidth',1)
106
107 %Dipole
108 [xNext,yNext]=curve(X(end),Y(end),theta,rDipole,phi);
109 X=[xNext];
110 Y=[yNext];
111 theta=theta+phi;
112 plot(X,Y,'r-', 'LineWidth',2)
113
114 %Drift
115 [xNext,yNext]=straight(X(end),Y(end),theta,sDrift);
116 X=[X(end);xNext];
117 Y=[Y(end);yNext];
118 plot(X,Y,'k:', 'LineWidth',1)
119
120 %FQ
121 [xNext,yNext]=straight(X(end),Y(end),theta,sFQ);
122 X=[X(end);xNext];
123 Y=[Y(end);yNext];
124 plot(X,Y,'b-', 'LineWidth',4)
125 %%%%%%%%%%%
126 end
127
128 function [xNext,yNext]=straight(x0,y0,theta,s)
129 dx=s/sqrt(1+tan(theta)^2);
130 if(cos(theta)<0)
131     dx=-abs(dx);
132 end
133 dy=s*sin(theta);
134
135 xNext=x0+dx;
136 yNext=y0+dy;
137 end
138
139 function [xNext,yNext]=curve(x0,y0,theta,R,phi)
140 xC=x0+R*cos(pi/2+theta);
141 yC=y0+R*sin(pi/2+theta);
142 x=[];
143 y=[];
144 T=linspace(theta-pi/2,theta+phi-pi/2,20);
145 for t=T
146     x=[x;R*cos(t)];
147     y=[y;R*sin(t)];
148 end
149 xNext=x+xC;
150 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 %GlobalVariables.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 Quadrupoles (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

```

1 %StartSectionTuning.m
2     %Tune two FODO cells to mach incomming beam to middle line
3 clear variables
4 close all
5
6 global betaXin;
7 global betaYin;
8 global Din;
9 global DPin;
10 global alphaXin;
11 global alphaYin;
12 %From LinAcc
13
14 %ideal values
15 global betaXL;
16 global betaYL;
17 global DL;
18 Fideal= [betaXL; %betaX
19         betaYL; %betaY
20         0; %alphaX
21         0]; %alphaY
22 %initial guess
23 X=[ 1; %FQ1
24     -0.2; %DQ1
25     0.5; %FQ2
26     -0.2]; %FQ3
27 dx=0.0001;
28 mu=1;
29 acceptance=0.0001;
30 run=0;
31 while (mu>acceptance)
32     F_x1=(F([X(1)+dx;X(2);X(3);X(4)])- ...
33           F([X(1)-dx;X(2);X(3);X(4)])) ./ (2*dx);
34     F_x2=(F([X(1);X(2)+dx;X(3);X(4)])- ...
35           F([X(1);X(2)-dx;X(3);X(4)])) ./ (2*dx);
36     F_x3=(F([X(1);X(2);X(3)+dx;X(4)])- ...
37           F([X(1);X(2);X(3)-dx;X(4)])) ./ (2*dx);
38     F_x4=(F([X(1);X(2);X(3);X(4)+dx])- ...
39           F([X(1);X(2);X(3);X(4)-dx])) ./ (2*dx);
40     J=[F_x1,F_x2,F_x3,F_x4];
41     %JpI= inv(J'*J)*J';
42     Fa=F(X);
43     Fdiff=Fideal-Fa;
44     Xnext=X+J\Fdiff;
45     mu=abs(norm(X-Xnext));
46     X=Xnext;
47     run=run+1;
48 end
49 X
50 k1=X(1);
51 k2=X(2);
52 k3=X(3);
53 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)=sDipIn/2;

```

```

104 %
105 Tunnel(:, :, 8)=MdriftIn;
106 Z_length(8)=sDIn;
107 %
108 Tunnel(:, :, 9)=Mq3;
109 Z_length(9)=sQIn/2;
110 %
111 Tunnel(:, :, 10)=Mq3;
112 Z_length(10)=sQIn/2;
113 %
114 Tunnel(:, :, 11)=MdriftIn;
115 Z_length(11)=sDIn;
116 %
117 Tunnel(:, :, 12)=MdipIn;
118 Z_length(12)=sDipIn/2;%
119
120 Tunnel(:, :, 13)=MdipIn;
121 Z_length(13)=sDipIn/2;
122 %
123 Tunnel(:, :, 14)=MdriftIn; %The matrix
124 Z_length(14)=sDIn; %The length
125 %
126 Tunnel(:, :, 15)=Mq4;
127 Z_length(15)=sQIn/2;
128 %
129
130 Z=cumsum([0, Z_length]);
131 BFX=zeros(3, Nblock+1);
132 BFX(:, 1)=BFX0;
133 BFY=zeros(3, Nblock+1);
134 BFY(:, 1)=BFY0;
135 Dvec=zeros(3, Nblock+1);
136 Dvec(:, 1)=Dvec0;
137
138 for i=1:Nblock
139     Mx=M_beta(Tunnel(1:2, 1:2, i)); %Get the 3x3 matrix
140     BFX(:, i+1)=Mx*BFX(:, i);
141     My=M_beta(Tunnel(3:4, 3:4, i));
142     BFY(:, i+1)=My*BFY(:, i);
143     S=S_dispersion(Tunnel(:, :, i));
144     Dvec(:, i+1)=S*Dvec(:, i);
145 end
146 sigmax=beamSize(0.02, BFX(1, :), Dvec(1, :));
147 sigmay=beamSize(0.02, BFY(1, :), 0);
148 %%%Plot Beta function %%%
149 subplot(1, 2, 1)
150 plot(Z, BFX(1, :), '*-');
151 hold on
152 plot(Z, BFY(1, :), 'x—');
153 xlabel('Z [m]')
154 ylabel('\beta [m]')
155 legend('\beta_x', '\beta_y')
156 grid on
157 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
```