Development of Quad Scan Software

Adam Clark, Rockhurst University, Department of Physics
John Power, ANL, Argonne Wakefield Accelerator Group

Abstract

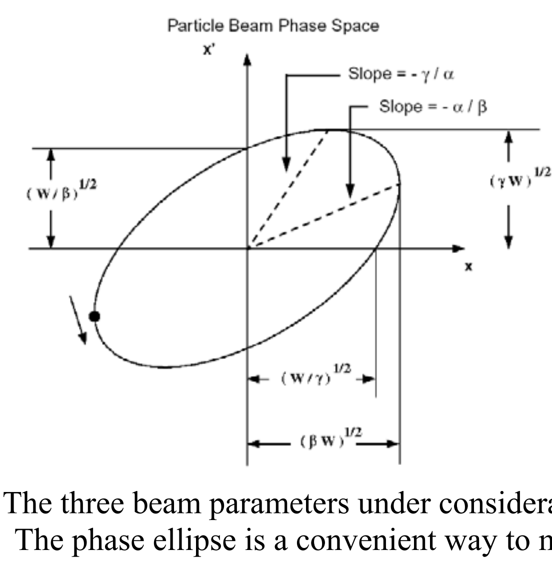
The following paper introduces the quad scan mathematically, presents MATLAB codes to relate three different beam formalisms and generate data for a simulated quad scan run in the Argonne Wakefield Accelerator. By developing a fitting routine, the simulated data can be analyzed to return the initial beam conditions, including emittance.

Introduction

The beam emittance is a measure of both beam size (mm) and beam divergence ($mrad$). Because beam divergence cannot be measured explicitly, the quad scanning technique can be used. By varying the strength of a quadrupole magnet, the focusing conditions are altered and the data obtained from the measurement of the spot sizes under the different conditions can be used to extrapolate the emittance from the envelope equations, the sigma beam parameters, and the Twiss parameters. Quad scan software developed for this paper will; verify the interchangeability of the three formalisms, generate data for a simulated quad scan, and analyze data generated in both MATLAB and Parmela to determine beam emittance.

Theoretical Background

Figure 1: The Phase Ellipse



The three beam parameters under consideration are all related by the phase space ellipse. The phase ellipse is a convenient way to model the behavior of the beam position and momentum. A number of useful identities can be derived from the phase ellipse which will be used to show the mathematical interchangeability of the three formalisms under consideration. These identities are shown below.

$$R_0 = \sqrt{\varepsilon\beta}$$
$$R_0^2 = \varepsilon\beta$$
$$R_0^' = -\alpha\sqrt{\frac{\varepsilon}{\beta}}$$
$$\beta = \frac{R_0^2}{\varepsilon}$$
$$\gamma = \frac{1+\alpha^2}{\beta}$$
$$R_0 R_0^' = -\varepsilon\alpha$$

Section 1: Envelope Formalism

Envelope formalism relates the spot size at the screen to; the size of the spot prior to entering the quadrupole, the emittance, and the derivative of the initial spot sizes as they will change with respect to the focal length. They are described by a second order differential equation. It should be noted that this equation explicitly ignores space charge effects.

$$R^{''} = \frac{\varepsilon^2}{R^3} \tag{1}$$

This can be solved analytically to retrieve the following equation;

$$R_s^2 = R_0^2 + 2R_0(R_0^' - \frac{R_0}{f})L + (\frac{\varepsilon^2}{R_0^2} + (R_0^' - \frac{R_0}{f})^2)L^2 \tag{2}$$

Section 2: Twiss Parameter Formalism

The Twiss parameters are coefficients describing the shape of the ellipse in phase space (Fig 1). The Twiss parameters (β,α,γ) are mathematically linked to the beam envelope parameters (Ro, Ro’,ε) . In the Twiss parameters, γ is dependent on both β and α which will simplify future substitutions. The beam emittance, ε is the same in both the parameters under consideration. The equation of the ellipse in terms of the Twiss parameters is given below.

$$\varepsilon = \gamma x^2 + 2\alpha x x' + \beta x'^2 \tag{3}$$

By applying the 2x2 transformation matrix (Wiedemann 159) to the matrix elements of the thin converging quadrupole and the drift, we can create a new 3x3 composite matrices which will allow us to calculate the Twiss parameters at the end of our quad scan.

The transport matrices for the quad and the drift are associated with the Twiss transformation matrix by the identities shown in equations 3 and 4 below.

$$\begin{pmatrix} 1 & 0 \\ -1/f & 1 \end{pmatrix} = \begin{pmatrix} C & S \\ C' & S' \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} C & S \\ C' & S' \end{pmatrix} \tag{5}$$

By substituting elements with the 3x3 Twiss indentity matrix (6), the 2x2 transport matrices can be transformed into the 3x3 Twiss matrix for the thin lens approximation for the quadrupole and the 3x3 Twiss matrix for the drift.

$$\begin{pmatrix} \beta(z) \\ \alpha(z) \\ \gamma(z) \end{pmatrix} = \begin{pmatrix} C^2 & -2CS & S^2 \\ -CC' & CS'+C'S & -SS' \\ C'^2 & -2C'S' & S'^2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \alpha_0 \\ \gamma_0 \end{pmatrix} \tag{6}$$

The result of substituting the quadrupole transport matrix (4) into the Twiss transport matrix (6) for the quadrupole is shown below.

$$\begin{pmatrix} 1 & 0 & 0 \\ 1/f & 1 & 0 \\ 1/f^2 & 2/f & 1 \end{pmatrix} \tag{7}$$

The result of substituting the drift transport matrix (5) into the Twiss transport matrix (6) for the drift is shown below.

$$\begin{pmatrix} 1 & -2L & L^2 \\ 0 & 1 & -L \\ 0 & 0 & 1 \end{pmatrix} \tag{8}$$

Equation (9) describes the transformation of the ellipse or Twiss parameters through the quadrupole and the drift. Since matrix multiplication is not commutative, the quad and drift transport matrices must be ordered as they appear in equation (9). This indicates that the beam line is passing through the quadrupole first and then the drift.

$$\begin{pmatrix}\beta(z)\\ \alpha(z)\\ \gamma(z)\end{pmatrix}=\begin{pmatrix}1 & -2L & L^2\\ 0 & 1 & -L\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 1/f & 1 & 0\\ 1/f^2 & 2/f & 1\end{pmatrix}\begin{pmatrix}\beta_0\\ \alpha_0\\ \gamma_0\end{pmatrix} \tag{9}$$

The 3x1 matrix (10) is the result of performing matrix multiplication. In the following steps, the phase ellipse identities will be useful in constructing the equation for the beam envelope from the Twiss parameters.

$$\begin{pmatrix}\beta(z)\\ \alpha(z)\\ \gamma(z)\end{pmatrix}=\begin{pmatrix}\beta_0+(-2L)(\frac{\beta_0}{f}+\alpha_0)+L^2(\frac{\beta_0}{f^2}+\frac{2\alpha_0}{f}+\gamma_0)\\ \frac{\beta_0}{f}+\alpha_0+(-L)(\frac{\beta_0}{f^2}+\frac{2\alpha_0}{f}+\gamma_0)\\ \frac{\beta_0}{f^2}+\frac{2\alpha_0}{f}+\gamma_0\end{pmatrix} \tag{10}$$

The envelope of the beam creates an incident spot size on the phosphor screen which is described by the equation (11) (Wiedemann 169).

$$R_s=\sqrt{\varepsilon}\sqrt{\beta(z)} \tag{11}$$

By squaring both sides of the expression and replacing $\beta(z)$ with the first element of the 3x1 matrix (10) the phase ellipse identities can be substituted in and the new equation is identical to the equation for the beam envelope (2).

$$\begin{aligned}R_s^2&=\varepsilon\beta(z)=\varepsilon\beta_0+2L(\frac{-\varepsilon\beta_0}{f}-\varepsilon\alpha_0)+L^2(\frac{\varepsilon\beta_0}{f^2}+\frac{2\varepsilon\alpha_0}{f}+\frac{\varepsilon+\varepsilon\alpha_0^2}{\beta_0})\\ R_s^2&=R_0^2+2L(\frac{-R_0^2}{f}+R_0R_0^{'})+L^2(\frac{R_0^2}{f^2}-\frac{2R_0R_0^{'}}{f}+\frac{\varepsilon}{(\frac{R_0^2}{\varepsilon})}+R_0^{'2})\\ R_s^2&=R_0^2+2R_0(R_0^{'}-\frac{R_0}{f})L+(\frac{\varepsilon^2}{R_0^2}+(R_0^{'}-\frac{R_0}{f})^2)L^2\end{aligned} \tag{14}$$

Section 3: Beam Sigma Formalism

The quad scan technique requires the beam to be transported through a quadrupole and a drift before it hits a phosphor screen. It is convenient to calculate the composite transport matrix (15) of the beam passing through the quadrupole structure (thin lens approximation) and the drift when developing beam sigma formalism.

$$\begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ -1/f & 1 \end{pmatrix} = \begin{pmatrix} 1-L/f & L \\ -1/f & 1 \end{pmatrix} \tag{15}$$

Beam sigma formalism is the most flexible of the three beam formalisms. The flexibility of this formalism comes from considering the beam to be an n-dimensional phase ellipse. Adapting this formalism to compute spot size in three dimensions is possible. For the purpose of this paper, the sigma matrix will only be developed in one dimension. The matrices (16), (17), and (18) are the composite transport matrix, its transpose, and the 1D sigma matrix, respectively.

$$M = \begin{pmatrix} 1-L/f & L \\ -1/f & 1 \end{pmatrix} \tag{16}$$

$$M^T = \begin{pmatrix} 1-L/f & -1/f \\ L & 1 \end{pmatrix} \tag{17}$$

$$\sigma_0 = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} \tag{18}$$

Only three of the four parameters are independent in matrix (18), therefore $\sigma_{12} = \sigma_{21}$ is a useful identity to simplify the matrix multiplication (Wiedemann 162). The equation below shows how the sigma beam matrix transforms in the quadrupole and the drift.

$$\begin{aligned} \sigma_2 &= M\sigma_0 M^T \\ &= \begin{pmatrix} 1-L/f & L \\ -1/f & 1 \end{pmatrix}\begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{pmatrix}\begin{pmatrix} 1-L/f & -1/f \\ L & 1 \end{pmatrix} \\ &= \begin{pmatrix} \sigma_{11}(1-L/f)^2 + L\sigma_{12}(1-L/f) + (\sigma_{12}(1-L/f) + L\sigma_{22})L & \sigma_{2,12} \\ \sigma_{2,21} & \sigma_{2,22} \end{pmatrix} \end{aligned} \tag{19}$$

From the resulting matrix we are interested in the $\sigma_{2,11}$ term. We substitute the identities in (20) into the $\sigma_{2,11}$ term (Wiedemann 163).

$$\begin{aligned} \sigma_{11} &= \varepsilon\beta \\ \sigma_{22} &= \varepsilon\gamma \\ \sigma_{12} &= -\varepsilon\alpha \end{aligned} \tag{20}$$

By algebraically expanding $\sigma_{2,11}$, equation (21) is developed in terms of the Twiss parameters.

$$R_s^2 = \sigma_{2,11} = \varepsilon\beta_0 + 2L(\frac{-\varepsilon\beta_0}{f} - \varepsilon\alpha_0) + L^2(\frac{\varepsilon\beta_0}{f^2} + \frac{2\varepsilon\alpha_0}{f} + \frac{\varepsilon + \varepsilon\alpha_0^2}{\beta_0}) \qquad (21)$$

This is the equivalent of equation (14) which relates the Twiss formalism to the equation for the beam envelope.

Developing a MATLAB Routine

Section 1: Numerical Comparison of Beam Formalisms

The code provided in "Appendix A" shows the three different formalisms developed in sections 1-3 computing the same answer for spot size given initial conditions. The results of the arbitrary parameters given in the code generate a spot size of .0022m.

Section 2: Returning Initial Conditions

Computing the initial beam conditions prior to going through the quadrupole magnet and the drift requires manipulating the matrices describing the motion of the beam according to equation (22) (Wiedemann 165).

$$\begin{pmatrix} \sigma_{0,11} \\ \sigma_{0,12} \\ \sigma_{0,22} \end{pmatrix} = (M_{\sigma,n}^T M_{\sigma,n})^{-1} M_{\sigma,n}^T \begin{pmatrix} \sigma_{1,11} \\ \sigma_{2,11} \\ \sigma_{3,11} \\ \ldots \end{pmatrix} \qquad (22)$$

The MATLAB code which implements this equation is presented in "Appendix B". The code requires another m-file (also in "Appendix B") to generate the data to be analyzed by equation (22). From these programs, initial conditions for alpha, beta, and epsilon are returned. The values in the case of the example are -.1, .8, and 1e-5 respectively.

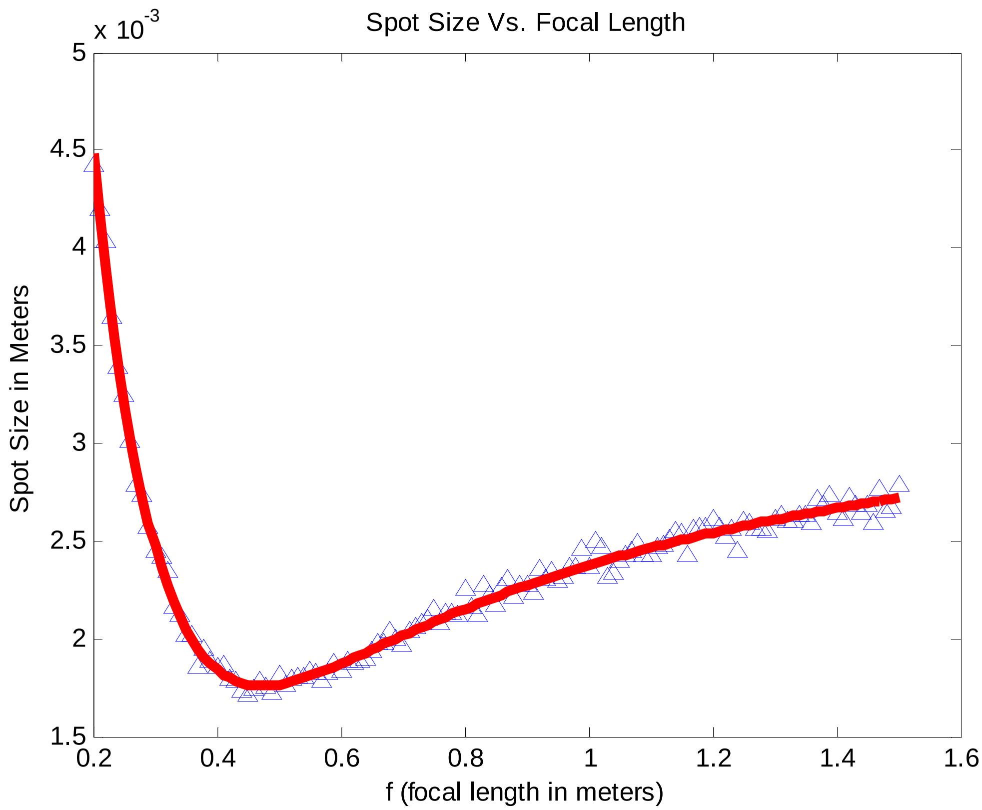
Using the MATLAB random number generator, a more realistic model of the data can be obtained. The code for generating data with 'noise' is identical to the m-file used in returning the ideal values with the exception that it includes a random number (Gaussian) multiplied by the computation for the spot size at the end of the 'for' loop. This change is annotated in "Appendix B".

The data generated with 'noise' is accessed by the script given in "Appendix C" which implements equation (23) (Wiedemann 165) and uses the 'fminsearch' function in MATLAB to create a best fit line to the generated data.

$$\sigma_{1,11}(k) = C^2(k)\sigma_{0,11} + 2C(k)S(k)\sigma_{0,12} + S^2(k)\sigma_{0,22} \qquad (23)$$

The best fit line is plotted on the graph of the generated data and is presented in Graph 1. The estimate of the initial beam conditions are presented in Table 1.

Graph 1



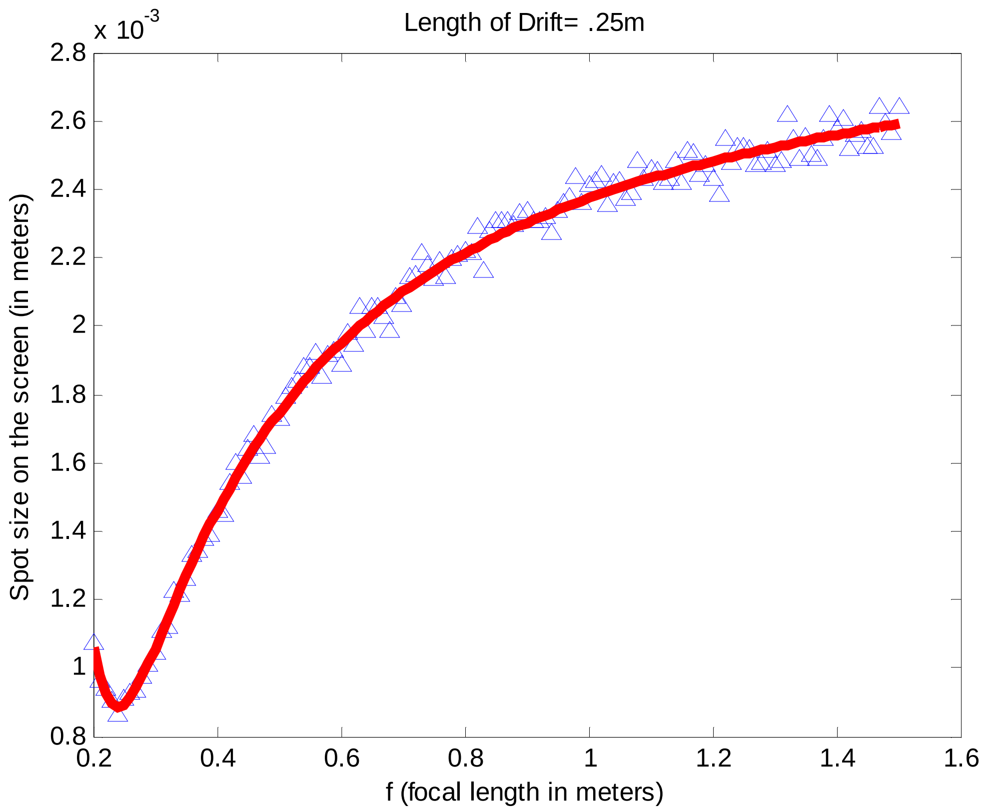
The lowest point on the graph indicates the beam waist. In terms of the phase ellipse, the beam waist is the smallest possible spot size given the focusing conditions and drift length. As the spot size gets smaller approaching the beam waist and the phase ellipse is said to be converging (Wiedemann 160). As the spot size increases at longer focal lengths, the beam is said to be diverging. For the purposes of this paper, we are most concerned with the narrow band of focal lengths that create a parabola around the beam waist.

When the lattice of focusing elements is set up to produce a waist, beam-beam effects are minimized and luminosity is maximized which is of importance in collision experiments and FEL's.

Section 3: Changing the initial conditions for drift length

By changing the length of the drift tube, a broad or narrow beam waist can be produced depending on the effect desired. In Graph 2, a narrow beam waist can be produced by shortening the drift which results in higher luminosity. In Graph 2 the beam waist is approximately .9mm. If the length of the drift is increased, the beam waist will be broader. In the case of Graph 3, the waist will be approximately 2.5mm. In the MATLAB model developed for this paper, changing the drift length impacts the accuracy of calculated initial conditions (Table 1).

Graph 2



Graph 3

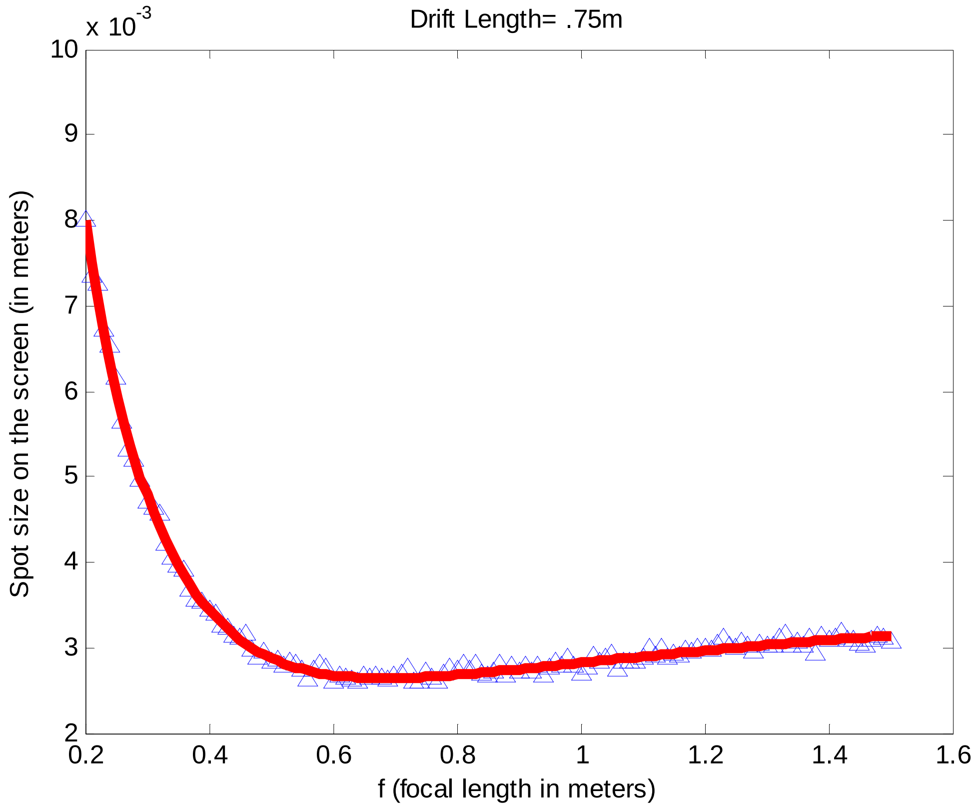


Table 1 shows the values of the parameters returned by the MATLAB code and the percent difference in calculated emittance at three different drift lengths. The emittance is calculated according to equation (24) (Wiedemann 163), shown below. Figure 2 shows a graphical representation of the screen at different positions downstream from the quadrupole.

$$\varepsilon = \sigma_{11}\sigma_{22} - \sigma_{12}^2 \tag{24}$$

Figure 2: Positioning the screen at different drift lengths.

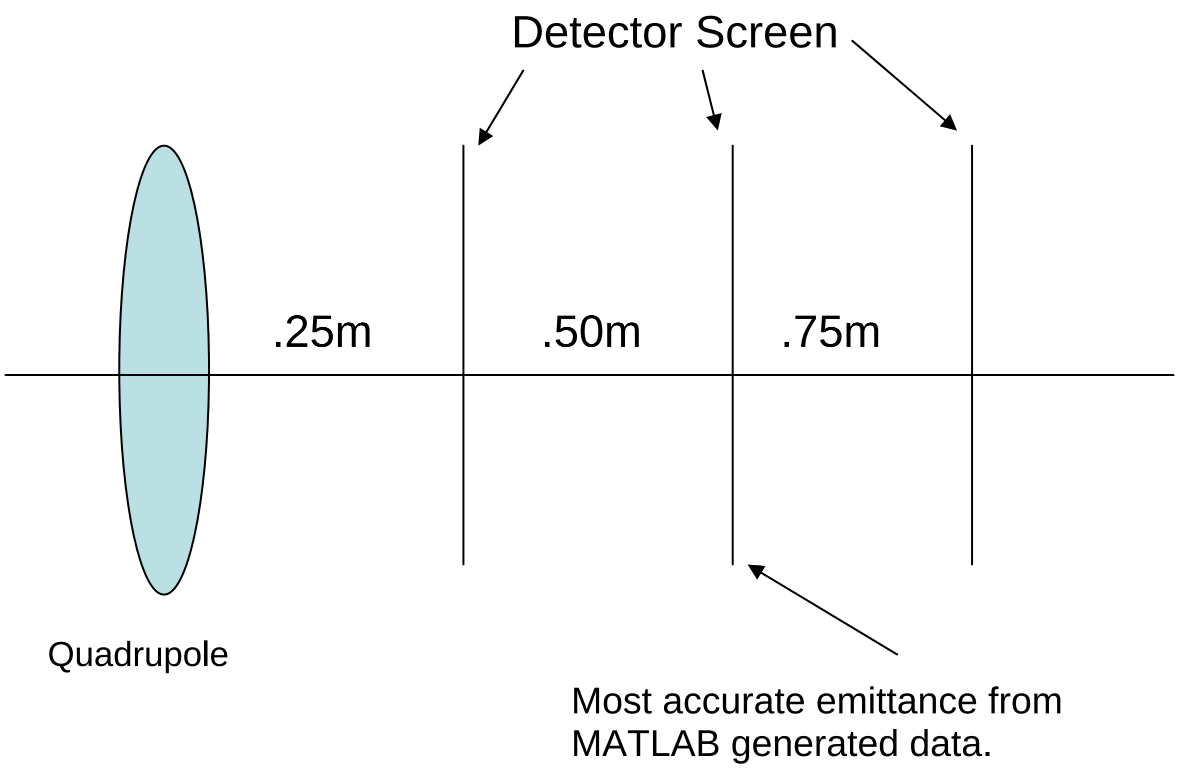


Table 1

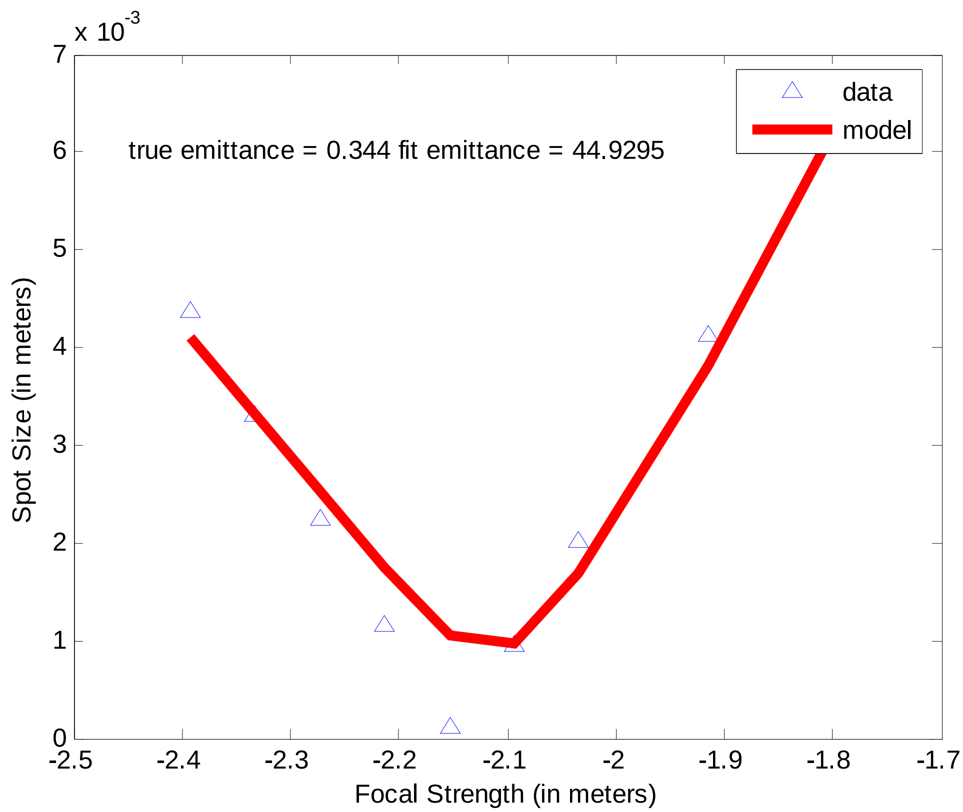
Initial values: Ideal Data
emittance = 1.000000e-005, alpha = -1.000000e-001, and beta= 8.000000e-001

Drift Length in meters	Emittance	Alpha	Beta	% Difference in Emittance
.25	1.005824e-005	-8.754183e-002	8.016145e-001	.5824%
.50	1.000668e-005	-9.981304e-002	8.067167e-001	.0668%
.75	9.936803e-006	-8.581043e-002	7.905808e-001	.63197%

The initial conditions retrieved from the data generated with ‘noise’ are less than 1% different from those retrieved from the ideal data. The best drift length for our model is around .5 meters.

Section 3: Parmela Data

Parmela generated data is an industry standard in accelerator physics modeling. The data generated by Parmela and analyzed using the MATLAB code from “Appendix B” is presented in Graph 3. The emittance calculated by the MATLAB script is displayed along with the emittance used by Parmela in generating the data. The percent difference between the expected emittance and the emittance calculated by the MATLAB code is 12960%.



Results

The MATLAB code accurately analyzes beam data from code generated in MATLAB, but performs poorly when analyzing data generated in Parmela. This occurs for several reasons. The MATLAB model was developed without space charge and uses the thin lens approximation for the transport matrix through a quadrupole magnet. Parmela includes space charge and a more accurate transport matrix through the focusing element which accounts for the length of the device. The failure of the MATLAB code to accurately predict Parmela data disqualifies the code from being used to perform a quad scan in the Argonne Wakefield Accelerator in its present form.

Conclusion

The beam parameters in an accelerator can be determined by comparing the incident spot size on a YAG screen with the focusing conditions of the quadrupole magnet which produced the spot. This technique can be mathematically verified in the thin lens approximation using Twiss formalism, envelope formalism, and beam sigma formalism. A code was developed in MATLAB to model the behavior of a particle beam under different focusing condtions. Conditions can be varied within the code in order to study the effect of different parameters on various beam properties such as beam waist. Our MATLAB model calculated input conditions which were numerically accurate to within 1% of those used in the theoretical model using the three different formalisms.

Failure of the MATLAB code to accurately analyze data from Parmela should not preclude future development of this technique. MATLAB code incorporating the realistic quadrupole transport matrix and space charge would improve the accuracy of the model and possibly make it suitable for determining beam emittance in a particle accelerator.

Works Cited

H. Wiedemann. Particle Accelerator Physics 3ed. Springer Berlin Heidelberg New York 2007

K. Wille. The Physics of Particle Accelerators: an Introduction. Oxford University Press Oxford, UK 2001.

Appendix A

Numerically compare three different beam formalisms in MATLAB

```
%define initial beam
alpha0=0;
beta0=.8;
gamma0=(1+alpha0^2)/beta0;
epsillon=1e-5;

%define beamline matricies
f=.9;
L=.5;
MD1=[1 L;0 1];
MQ1=[1 0;-1/f 1];
%M=MD1*MD2*MQ1*MD3*MQ2
%
%
%envelope formalism {R0, ROprime, epsillon}
R0=sqrt(epsillon*beta0);
Rprime0=-alpha0*sqrt(epsillon/beta0);
R_screen_envelope=sqrt(...
    (R0^2) + ...
    (2*R0)*(Rprime0-R0./f)*L + ...
    ( (epsillon/R0)^2 + (Rprime0-R0./f).^2 ) * L^2)
%
%Twiss parameters formalism (alpha, beta, epsillon}
MD=[1 -2*L L^2; 0 1 -L; 0 0 1];
MQ=[1 0 0;1/f 1 0;1/f^2 2/f 1];
Twiss0=[beta0;alpha0;gamma0];
Twiss=MD*MQ*Twiss0;
beta_final=Twiss(1);
Rssquared=epsillon*beta_final;
R_screen_twiss=sqrt(Rssquared)
%
%Sigma matrix formalism. { sigma(1,1), sigma(1,2), sigma(2,2) }
MDsig=[1 L;0 1];
MQsig=[1 0;-1/f 1];
M=MDsig*MQsig;
sig11=epsillon*beta0;
sig22=epsillon*gamma0;
sig12=-epsillon*alpha0;
Sig=[sig11 sig12;sig12 sig22];
SigFinal=M*Sig*M';
R_screen_matrix=sqrt(SigFinal(1,1))
%
```

Appendix B

Retrieving initial conditions from generated data.

```
%Inverting the Matrix to Recover Initial Conditions
clear
L=.5;
MquadScan=dlmread('quadScanData.txt');
fquad=MquadScan(:,1);
sigma11Screen=MquadScan(:,2).^2;
Msigma=[];
 for iquad=1:length(fquad)
     f=fquad(iquad);
     C=1-L/f;
     S=L;
     Msigma=[Msigma; C^2 2*C*S S^2];
 end
 Msigma;
 sigma11Screen;
 MsigN=inv(Msigma'*Msigma)*Msigma';
 sigma0=MsigN*sigma11Screen;

 sigma11=sigma0(1);
 sigma12=sigma0(2);
 sigma22=sigma0(3);

 epsillon=sqrt(sigma11*sigma22-sigma12^2);
 beta0=sigma11/epsillon;
 alpha0=-sigma12/epsillon;

 fprintf('Retrieved values: epsillon=%d, alpha=%d, beta=%d\n\n',…
    epsillon, alpha0, beta0)
```

(generateData.m)

```
%define initial beam
clear
alpha0=-0.1;
beta0=.8;
gamma0=(1+alpha0^2)/beta0;
epsillon=1e-5;


%define beamline matricies
fquad=[0.2:0.01:1.2];
L=0.5;


%Sigma matrix formalism. { sigma(1,1), sigma(1,2), sigma(2,2) }

for iquad=1:length(fquad)
    f=fquad(iquad);
    Mdsig=[1 L;0 1];
```

```
    Mqsig=[1 0;-1/f 1]; %[C1 S1; C1’ S1’]
    M=Mdsig*Mqsig;
    sig11=epsillon*beta0;
    sig22=epsillon*gamma0;
    sig12=-epsillon*alpha0;
    Sig=[sig11 sig12;sig12 sig22];
    SigFinal=M*Sig*M’;
    R_screen_matrix(iquad)=sqrt(SigFinal(1,1)); (Generating Data with
noise includes multiplication by a random (Guassian) number at this
step.)
end
%

%plot data point
plot(fquad, 1e3*R_screen_matrix,’+’)

MquadScan=[fquad’,R_screen_matrix’];
dlmwrite(‘quadScanData.txt’, MquadScan);

fprintf(‘Initial values: epsillon=%d, alpha=%d, beta=%d\n\n’,…
    epsillon, alpha0, beta0)
```

Appendix C

This MATLAB code analyzes data with noise in order to return initial conditions and perform a data fit using equation (23) from section 4.

```
clear

%generate data for line
quadScandata =dlmread('quadScanData.txt');  %col1=f; col2=spot size
f_xdata = quadScandata(:,1);
spotSizescreen_ydata=quadScandata(:,2);
L=.5;
plot(f_xdata,spotSizescreen_ydata, '^'); hold on


%find the best fit to equation 548 Wiedermann
alpha0=-0.2; beta0=.7; gamma0=(1+alpha0^2)/beta0; epsillon=2e-5;
paramsin0=[epsillon*beta0; -epsillon*alpha0; epsillon*gamma0;];
%paramsin0=[8e-6, 1.2625e-5, 1e-6];
options=optimset('TolX',1e-8);
sseHWn = @(x) sseHW(x, L, f_xdata, spotSizescreen_ydata);
paramsout = fminsearch(sseHWn,paramsin0,options);
% alpha0=-0.1; beta0=.8; gamma0=(1+alpha0^2)/beta0; epsillon=1e-5;
% paramsout=[epsillon*beta0; -epsillon*alpha0; epsillon*gamma0;]

%extract data
sigma011=paramsout(1);
sigma012=paramsout(2);
sigma022=paramsout(3);


%generate model fit
for iquad=1:length(f_xdata)
    f=f_xdata(iquad);
    C=1-L/f; S=L;
    sigma11screen=…
        C^2*sigma011 + 2*C*S*sigma012 + S^2*sigma022;
    spotSizeScreen_model(iquad)=sqrt(sigma11screen);
end
plot(f_xdata,spotSizeScreen_model,'r','LineWidth',4); hold off

epsillon0jp=sqrt(sigma011*sigma022-sigma012^2);
beta0jp=sigma011/epsillon0jp;
alpha0jp=-sigma012/epsillon0jp;

fprintf('Retrieved values: epsillon=%d, alpha=%d, beta=%d\n\n',…
    epsillon0jp, alpha0jp, beta0jp)

% plot results
% plot(f_xdata, sigma11scdata, '^',f_xdata,sigma111,'-')
% legend('data', 'model')
% mstring = ['best fit m = ', num2str(m), ' b = ', num2str(b)]
% text(1, 20, mstring)
```

(sseHW.m)

```
function sumsq = sseHW(paramsin, L, f_xdata, spotSizescreen_ydata)
%eqn 5.48
%
sigma011=paramsin(1);
sigma012=paramsin(2);
sigma022=paramsin(3);

for iquad=1:length(f_xdata)
    f=f_xdata(iquad);
    C=1-L/f; S=L;
    sigma11screen=…
        C^2*sigma011 + 2*C*S*sigma012 + S^2*sigma022;
    spotSizeScreen_model(iquad)=sqrt(sigma11screen);
end
%plot(f_xdata,spotSizeScreen_model)
sumsq= sum((spotSizeScreen_model’-spotSizescreen_ydata).^2);
```