

A Matlab Solution of the 1D, 2D, and 3D Beam Envelope Equations.

Introduction: This Wakefield note describes how to use Matlab [1-3] to solve the beam envelope equation, including both space charge and emittance, in a drift for three different cases. In each case, the equation(s) to be solved is first presented and then the exact Matlab code for solving the equation(s) is given along with the corresponding output. No attempt is made to explain the physics behind the envelope equation since the literature by the beam physics experts already does this well [5-9]. The motivation for developing these routines is to eventually use them as part of a modified Quad Scan technique to measure the emittance of space-charge dominated beams.

The paper is divided into three main sections. In the first section, the simplest case is solved, that of a round (axisymmetric) beam of infinite length. In the second section, the case of an elliptical (non-axisymmetric) beam of infinite length is solved. And finally, in the third section, the case of most interest to the photoinjector/linac community is solved, the elliptical (non-axisymmetric), bunched beam.

[I.] The Continuous Round Beam Envelope Equation.

In this section, a numerical solution of the uniform beam envelope equation is obtained, with *Matlab*, for the case of a continuous (i.e. infinite length), cylindrical (i.e. a round cross-section or axisymmetric) beam.

THE 1D ENVELOPE EQUATION

The round beam envelope equation can be found in the reference [Ref. 5, Eqn. 4.112].

$$R''(s) + \mathbf{k}_0(s) R(s) - \frac{\mathbf{e}^2}{R(s)^3} - \frac{K}{R(s)} = 0 \quad (1.1)$$

where s is direction of propagation, the prime means a derivative was taken with respect to s , R is the envelope or radius of the beam, $\mathbf{k}_0(s)$ is the external focusing term, \mathbf{e}_r is the unnormalized transverse emittance, and K is the space charge parameter, or generalized perveance, which is given by

$$K = \frac{I}{I_0} \left(\frac{2}{\mathbf{b}^3 \mathbf{g}^3} \right) \quad (1.2)$$

where I is the peak current, $I_0 = 17000$ Amps is the *characteristic current*, and β and γ are the usual relativistic factors.

For future reference, I show how to convert the uniform beam envelope (of radius R) equation {1.1} to the RMS envelope (of RMS radius a) equation by substituting the relationships $R = 2 * a$ and $\mathbf{e} = 4 * \mathbf{e}_{rms}$ into {1.1}:

$$a''(s) + \mathbf{k}_0(s) a(s) - \frac{\mathbf{e}_{rms}^2}{a(s)^3} - \frac{K}{4a(s)} = 0$$

I am not sure, however, if I should also replace the peak current used in the definition of the space charge parameter {1.2} with the RMS current instead.

REDUCING THE ODE TO FUNDAMENTAL FORM

To solve a linear differential equation in Matlab [2] we need to first reduce the 2nd order ODE [1.1] to a system of first order equations [4] in the form of the $\{z'_i\} = \{f_i(t, z_j)\}$. Since we are interested in the evolution of the envelope in a drift, we set the focusing term to zero ($\mathbf{k}_0(s) = 0$) and drop the explicit s dependence of R . Now we define a 2-component **state vector** \mathbf{z} as,

$$z_1 = R \tag{1.3}$$

$$z_2 = R' \tag{1.4}$$

The three equations [1.1, 1.3, & 1.4] can be rearranged to give the derivative of the state vector, or **state derivative** \mathbf{z}' as a system of two, first-order equations,

$$z'_1 = z_2 \tag{1.5}$$

$$z'_2 = \frac{\mathbf{e}^2}{z_1^3} + \frac{K}{z_1} \tag{1.6}$$

or in matrix form,

$$\begin{Bmatrix} z'_1 \\ z'_2 \end{Bmatrix} = \begin{Bmatrix} z_2 \\ \frac{\mathbf{e}^2}{z_1^3} + \frac{K}{z_1} \end{Bmatrix} \tag{1.7}$$

THE MATLAB SOLUTION OF THE 1D ENVELOPE EQUATION

The Matlab code to solve Equations [1.7], for a particular case, is shown in Figure {1} and Figure {2}. The particular case considered here has: (1) initial conditions

$R(s=0) = 2 \text{ mm}$ and $R'(s=0) = -2.0 \text{ mrad}$; (2) $\epsilon_n = 4 * \epsilon_{r,n} = 1 \text{ mm mrad}$; (3) Energy = 7 MeV; (4) drift length = 1000 mm; (5) RMS bunch length = 1mm; and (6) Q = 1 nC.

In the main program (*Envelope1D.m*, see Fig. 1) Matlab's built-in ODE solver, *ode45* (a 4th/5th order Runge-Kutta solver) is used on line 29 to actually solve the reduced system. (Note that other built-in ODE solvers also work, like *ode23*, a 2nd/3rd order Runge-Kutta solver.) To use *ode45* we must pass it: (1) a function to compute the state derivative (*prime1D.m*, see Fig. 2); (2) the range of interest; (3) the initial value of the state vector (i.e. the initial conditions of the beam); and (4) a parameter list {p1, p2 ...}. From Eqn.'s [1.3] and [1.4] we see that the initial values of $R(s=0)$ and $R'(s=0)$ are also the initial values of the state vector $z_1(s=0) = R(s=0)$ and $z_2(s=0) = R'(s=0)$. Since the rest of the main program is well commented, and should be straightforward to most Matlab users, I won't describe it since it is probably unnecessary and I am too lazy!

```

1. % 1D Round beam envelope equation with Space Charge
2. %
3. %----constants
4. %
5. c=3e8; % speed of light (m/s)
6. %----Global beam parameters
7. %
8. Qbunch=1e-9; % beam charge (C)
9. Wkin=7; % kinetic energy (MeV)
10. distance=1.0; % length of drift
11. %----Transverse phase space
12. %
13. initR=0.0002; initRprime=-0.002; % initial size (m) and slope (rad)
14. emitn=1.0e-6; % normalized X emittance (m*rad)
15. %----Longitudinal phase space
16. %
17. dz=2.2*0.001; % FWHM bunch length (m) = 2.2 * RMS
18. %----Prepare data for ode solver
19. %
20. gamma=1+Wkin/0.511; % relativistic mass
21. beta=sqrt(1-1/gamma^2); % normalized velocity
22. dt=dz/(beta*c); % bunch length (s)
23. Ipeak=Qbunch/dt; % peak current ( Amps )
24. sRange=(0:0.1:distance);
25. Rvector0=[initR, initRprime];
26. %-----solve envelope equation
27. %
28. tic
29. [s, Rvector]=ode45('prime1D',sRange,Rvector0,[],emitn,Ipeak,Wkin);
30. toc % time to solve in seconds
31. %----prepare data for plots
32. %
33. Rsize=1000*Rvector(:,1);
34. Rslope=1000*Rvector(:,2);
35. s=1000*s;
36. %----plot results

```

```

37. %
38. subplot(2,1,1)
39. plot(s,Rsize,'o-')
40. xlabel('s [mm]'); ylabel(' R [mm]');
41. axis([-inf, inf, 0, inf])
42. text(50,5,'The Beam Envelope Size R ')
43. subplot(2,1,2)
44. plot(s,Rslope,'o-')
45. xlabel('s [mm]'); ylabel(' R'' [mrad]');
46. text(500,2,'The Beam Envelope Slope R'' ')

```

Figure (1). *Envelope1D.m* (Matlab m-file for Solving Eqn. 1.6)

The other piece of code needed to solve this problem is the Matlab function (*prime1D.m*, see Fig. 2) that returns the state derivative, z' , given the state vector as input, z . The exact way that these state derivatives are handled by *ode45* is not obvious (at least to me) but it need not concern us since the function *prime1D.m* is used by the ODE solver and not directly by the programmer. It is also interesting that to note that in our problem there is no explicit dependency on the propagation variable s , but we must pass it anyway from the main program as $sRange$ (line 29, Fig. 1) to the function as t (line 1, Fig. 2).

```

1. function zprime1D = prime1D(t,z,dummy,en,I,W);
2. %Calling syntax: zprime1D = prime1D(t,z,dummy,eX);
3. %
4. %Inputs:
5. %   t = distance[m];
6. %   z = [z(1); z(2)] = [size (m), slope (rad)]
7. %   dummy = this place holder is needed for 'options' in ode45
8. %   en = normalized emittance (m*rad)
9. %   I = peak current ( Amps )
10. %   W = kinetic energy (MeV)
11. %
12. %Outputs:
13. % zprime = [z(2); e^2/(z(1))^3 + K/z(1)];
14. %
15. IA = 17000;           % Alfven Current (Amps)
16. gamma=1+W/0.511;    % Relativistic mass
17. beta=sqrt(1-1/gamma^2); % normalized velocity
18. %
19. e=en/beta*gamma;    % unnormalized emittance; units = m*rad
20. K=(2*I)/(IA*(beta*gamma)^3); % generalized perveance [Reiser,
21. %
22. zprime1D=[z(2); (e^2)/(z(1))^3 + K/z(1)];

```

Figure (2). *prime1D.m* (Matlab Function to calculate the state derivative.)

Finally, we can run the above code by putting them into the same directory, somewhere in the Matlab path, and typing:

```
>>Envelope ID
>>
```

Figure (3). Text to type at the Matlab prompt.

The output results from executing the command of Figure 3 is shown in Figure 4. Here the top figure shows the beam radius while the bottom plot show the beam slope evolving starting at $s=0$, passing through a waist near $s=100$ mm, and diverging at the end of the 1000 mm drift. From the tic-toc functions on lines 28 & 30 of *Envelope1D.m* we that it takes 30 msec to solve for all 10 points shown in Figure 3.

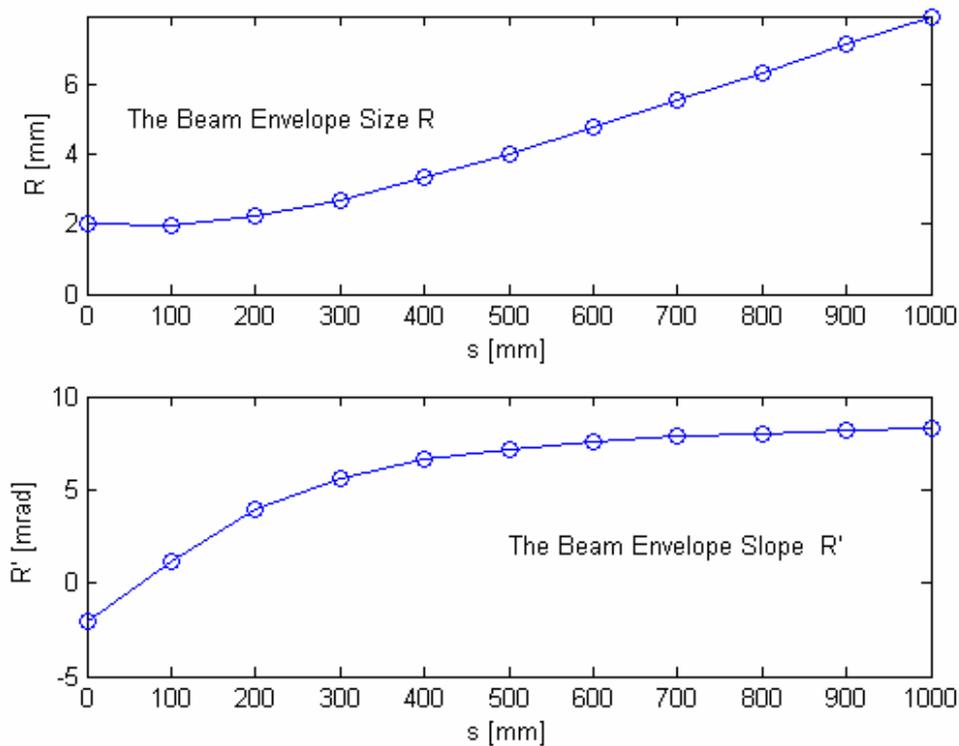


Figure (4). Numerical solution of the beam envelope equation [1.1]. The top trace is the beam radius (R) vs. distance (s) and the bottom trace is the beam slope (R') vs. distance (s).

[II] The Continuous Elliptical Beam Envelope Equation

In this section, a numerical solution of the uniform beam envelope equation is obtained, with *Matlab*, for the case of a continuous (i.e. infinite length), non-axisymmetric (i.e. an elliptical cross-section) beam.

THE 2D ENVELOPE EQUATION

The coupled envelope equations for a non-axisymmetric, unbunched beam can be found in the reference [Ref. 5, Eqn. 4.178].

$$X' + \mathbf{k}_{x0}(s)X - \frac{\mathbf{e}_x^2}{X^3} - \frac{2K}{X+Y} = 0 \quad (1.8)$$

$$Y' + \mathbf{k}_{y0}(s)Y - \frac{\mathbf{e}_y^2}{Y^3} - \frac{2K}{X+Y} = 0 \quad (1.9)$$

where the variables are defined as in section [I] except that $X(s)$ and $Y(s)$ are the beam envelopes in the x and y coordinates respectively and we allow for the emittances in the x and y planes to be unequal, $\mathbf{e}_x \neq \mathbf{e}_y$.

Again, for future reference, I show how to convert the 2D uniform beam envelope (of semi-axes X and Y) equations {1.8,9} to the RMS envelope (of RMS semi-axes a_x and a_y) equation by substituting the relationships $X = 2 * a_x$, $\mathbf{e}_x = 4 * \mathbf{e}_{xrms}$, $Y = 2 * a_y$ and $\mathbf{e}_y = 4 * \mathbf{e}_{yrms}$ into {1.8,9}:

$$a_x''(s) + \mathbf{k}_0(s)a_x(s) - \frac{\mathbf{e}_{xrms}^2}{a_x(s)^3} - \frac{2K}{4a_x(s)} = 0$$

$$a_y''(s) + \mathbf{k}_0(s)a_y(s) - \frac{\mathbf{e}_{yrms}^2}{a_y(s)^3} - \frac{2K}{4a_y(s)} = 0$$

And once again, I am not sure if I should also replace the peak current used in the definition of the space charge parameter {1.2} with the RMS current instead.

REDUCING THE COUPLED ODE'S TO FUNDAMENTAL FORM

It is surprisingly easy to solve the pair of coupled ODE [1.8, 1.9] with Matlab if you understood how to solve the single ODE of section I. Proceeding as before we define the **state vector z** as,

$$z_1 = X \quad (1.10)$$

$$z_2 = X' \quad (1.11)$$

$$z_3 = Y \quad (1.12)$$

$$z_4 = Y' \quad (1.13)$$

Equations [1.8-1.13] can be rearranged to give the **state derivative z'** as a system of four, first-order equations,

$$z_1' = z_2 \quad (1.14)$$

$$z_2' = \frac{\mathbf{e}_x^2}{z_1^3} + \frac{2K}{z_1 + z_3} \quad (1.15)$$

$$z'_3 = z_4 \quad (1.16)$$

$$z'_4 = \frac{e_y^2}{z_3^3} + \frac{2K}{z_1 + z_3} \quad (1.17)$$

or in matrix form,

$$\begin{Bmatrix} z'_1 \\ z'_2 \\ z'_3 \\ z'_4 \end{Bmatrix} = \begin{Bmatrix} z_2 \\ \frac{e_x^2}{z_1^3} + \frac{2K}{z_1 + z_3} \\ z_4 \\ \frac{e_y^2}{z_3^3} + \frac{2K}{z_1 + z_3} \end{Bmatrix} \quad (1.18)$$

THE MATLAB SOLUTION OF THE 2D ENVELOPE EQUATION

Proceeding in almost the exact same way as in section I, the code for both the main program (*Envelope2D.m*) and the function that returns the state derivative (*prime2D.m*) are given in figures 5 and 6 respectively.

```

1. % 2D Elliptical beam envelope equation with space charge.
2. %
3. %---constants
4. %
5. c=3e8; % speed of light (m/s)
6. %---Global beam parameters
7. %
8. Qbunch=1e-9; % beam charge (C)
9. Wkin=7; % kinetic energy (MeV)
10. %---Transverse phase space
11. %
12. initX=0.001; initXp=0.0; % initial X size (m) and X slope (rad)
13. initY=0.002; initYp=-0.0015; % initial Y size (m) and Y slope (rad)
14. emitXn=1.0e-6; emitYn=1.0e-6; % normalized X and Y emittances (m*rad)
15. %---Longitudinal phase space
16. %
17. dz=2.2*0.001; % FWHM bunch length (m) = 2.2 * RMS
18. %---Prepare data for ode solver
19. %
20. gamma=1+Wkin/0.511; % relativistic mass

```

```

21. beta=sqrt(1-1/gamma^2);           % normalized velocity
22. dt=dz/(beta*c);                 % bunch length (s)
23. Ipeak=Qbunch/dt;                 % peak current ( Amps )
24. distance=1.0;                    % length of drift
25. sRange=(0:0.1:distance);        % range for plots
26. XYvector0 =[initX, initXp, initY, initYp]; % I.C.'s [size; slope]
27. %---solve envelope equation
28. %
29. tic
30. [s, XYvector]=ode45('prime2D',sRange,XYvector0,[],emitXn,emitYn,Ipeak,Wkin);
31. toc                               % time to solve in seconds
32. %---prepare data for plots
33. %
34. Xsize=1000*XYvector(:,1);
35. Xslope=1000*XYvector(:,2);
36. Ysize=1000*XYvector(:,3);
37. Yslope=1000*XYvector(:,4);
38. s=1000*s;
39. %---plot results
40. %
41. subplot(2,1,1)
42. plot(s,Xsize,'o-',s,Ysize,'+');
43. xlabel('s [mm]'); ylabel(' Size [mm]');
44. legend('X envelope', 'Y envelope');
45. axis([-inf, inf, 0, inf])
46. text(5,0.27,'The Beam Envelope Size')
47. subplot(2,1,2)
48. plot(s,Xslope,'o-',s,Yslope,'+');
49. xlabel('s [mm]'); ylabel(' Slope [mrad]');
50. legend('X"', 'Y " ');
51. text(5,2,'The Beam Envelope Slope')

```

Figure (5). *Envelope2D.m* (Matlab m-file for Solving Eqn. 1.17)

```

1. function zprime2D = prime2D(t,z,dummy,enX,enY,I,W);
2. %Calling syntax: zprime2D = 2Dprime(t,z);
3. %
4. %Inputs:
5. %    t = distance[m];
6. %    z = [z(1); z(2);           [X size (m); X slope (rad);
7. %    z(3); z(4)]             Y size (m); Y slope (rad)]
8. %    dummy = this place holder is needed for 'options' in ode45
9. %    enX = normalized X emittance (m*rad)
10. %    enY = normalized Y emittance (m*rad)
11. %    I = peak current ( Amps )
12. %    W = kinetic energy (MeV)
13. %
14. %Output: zprime2=[z(2); (e^2)/(z(1))^3 + 2*K/(z(1)+z(3));
15. %    z(4); (e^2)/(z(3))^3 + 2*K/(z(1)+z(3));];

```

```

16. %
17. IA = 17000;           % Alfven Current (Amps)
18. gamma=1+W/0.511;    % relativistic mass
19. beta=sqrt(1-1/gamma^2); % normalized velocity
20. %
21. eX=enX/gamma;       % unnormalized X emittance; units = m*rad
22. eY=enY/gamma;       % unnormalized Y emittance; units = m*rad
23. K=(2*I)/(IA*(beta*gamma)^3); % generalized perveance [Reiser,
24. %
25. zprime2D=[z(2); (eX^2)/(z(1))^3 + 2*K/(z(1)+z(3));
26.           z(4); (eY^2)/(z(3))^3 + 2*K/(z(1)+z(3));];

```

Figure (6). *prime2D.m* (Matlab function to calculate the state derivative.)

```

>>Envelope2D
>>

```

Figure (7). Text to type at the Matlab prompt.

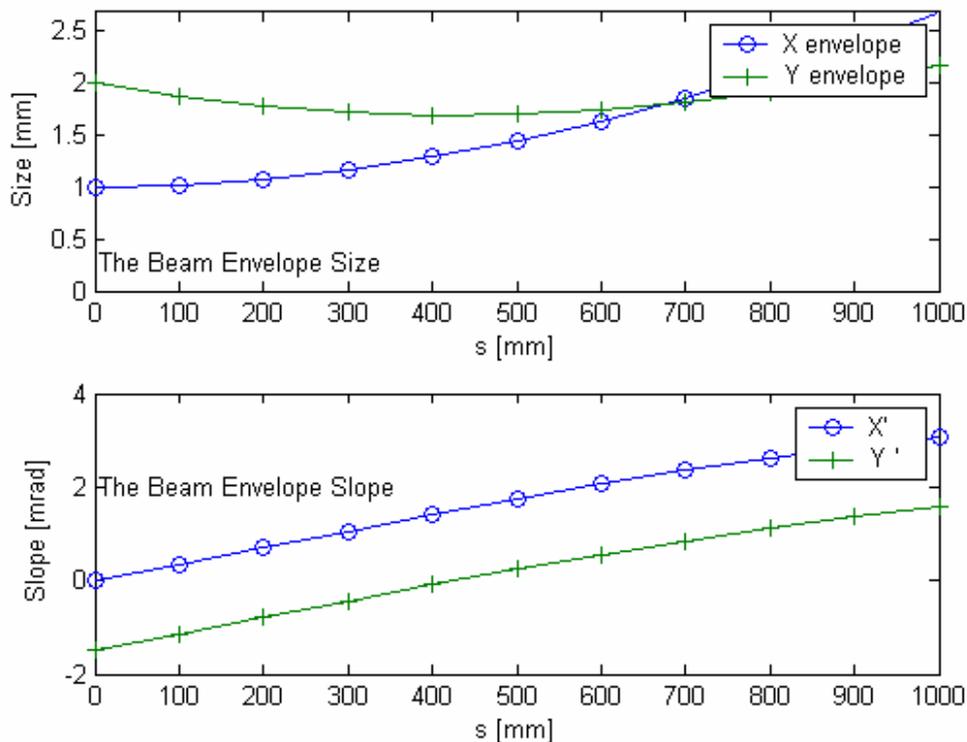


Figure (8). Numerical solution of the beam coupled beam envelope equations [1.8 & 1.9]. The top trace is the beam size vs. distance (s) and the bottom trace is the beam slope vs. distance (s). The tic-toc commands on lines 29 & 31 of *Envelope2D* show that the problem took 40 msec to solve.

[III] The Beam Envelope Equation of an Infinitely Long, Non-Axisymmetric Beam

In this section, a numerical solution of the 3D envelope equation is obtained, with *Matlab*, for the case of a bunched, non-axisymmetric beam. Of the three cases, this is the case of most interest for us, but also contains the most complications. The complications are not physics related, but arise from the confusion over the proper choice of longitudinal coordinates and the corresponding units. Fortunately, the Matlab solution is still straightforward and any mistakes in the units can be easily changed later after the Matlab routine is working.

THE 3D RMS ENVELOPE EQUATION

The set of three, coupled RMS envelope equations for a non-axisymmetric, bunched beam can be found in the reference [Ref. 6, Eqn. 9.41 – 9.43].

$$a_x'' + \mathbf{k}_{x0}(s)a_x - \frac{\mathbf{e}_{x,rms}^2}{a_x^3} - \frac{3K_3(1-f)}{5\sqrt{5}(a_x + a_y)a_z} = 0 \quad (1.19)$$

$$a_y'' + \mathbf{k}_{y0}(s)a_y - \frac{\mathbf{e}_{y,rms}^2}{a_y^3} - \frac{3K_3(1-f)}{5\sqrt{5}(a_x + a_y)a_z} = 0 \quad (1.20)$$

$$a_z'' + \mathbf{k}_{z0}(s)a_z - \frac{\mathbf{e}_{z,rms}^2}{a_z^3} - \frac{3K_3f}{5\sqrt{5}a_x a_y} = 0 \quad (1.21)$$

where a_i are the RMS beam sizes, $\mathbf{e}_{i,rms}$ are the rms emittances, K_3 is the three dimensional space-charge parameter, and f is a form factor. The 3D space charge parameter is given by

$$K_3 = \left(\frac{cQ}{I_0} \right) \left(\frac{2}{\mathbf{b}^2 \mathbf{g}^3} \right) \quad (1.22)$$

where $I_0 = 17000$ Amps is the *characteristic current*, \mathbf{b} and \mathbf{g} are the usual relativistic factors, I is the average current over an RF period and λ is the RF wavelength. Note that the 3D definition of the generalized perveance, K_3 , differs from the 1D and 2D space charge parameter, K ; it has one less power of \mathbf{b} in the denominator, it has units of 'm', and the constant factor in front is due The ellipsoidal form factor f is a function of the parameter $p = \mathbf{g}r_z/r_x r_y$ where r_i are the semi-axes of the ellipsoid. The exact values for $f(p)$ are tabulated in Reference [9] and I used this as a lookup table.

It is important to note that I am not solving the uniform beam envelope equation this time but the RMS envelope equation. This means that the RMS emittance must be used and not the total emittance of the round beam as in sections I and II.

REDUCING THE COUPLED ODE'S TO FUNDAMENTAL FORM

Proceeding as in the last two sections we define the **state vector \mathbf{z}** as,

$$z_1 = a_x \quad (1.23)$$

$$z_2 = a'_x \quad (1.24)$$

$$z_3 = a_y \quad (1.25)$$

$$z_4 = a'_y \quad (1.26)$$

$$z_5 = a_z \quad (1.27)$$

$$z_6 = a'_z \quad (1.28)$$

Equations [1.19-1.28] can be rearranged to give the **state derivative \mathbf{z}'** as a system of six, first-order equations,

$$z'_1 = z_2 \quad (1.29)$$

$$z'_2 = \frac{\mathbf{e}_{xrms}^2}{z_1^3} + \frac{3K_3(1-f)}{(z_1 + z_3)z_5} \quad (1.30)$$

$$z'_3 = z_4 \quad (1.31)$$

$$z'_4 = \frac{\mathbf{e}_{yrms}^2}{z_3^3} + \frac{3K_3(1-f)}{(z_1 + z_3)z_5} \quad (1.32)$$

$$z'_5 = z_6 \quad (1.33)$$

$$z'_6 = \frac{\mathbf{e}_{zrms}^2}{z_5^3} + \frac{3K_3f}{z_1z_3} \quad (1.34)$$

or in matrix form,

$$\begin{Bmatrix} z_1' \\ z_2' \\ z_3' \\ z_4' \\ z_5' \\ z_6' \end{Bmatrix} = \begin{Bmatrix} \frac{z_2}{z_1^3} + \frac{3K_3(1-f)}{(z_1+z_3)z_5} \\ \frac{z_4}{z_3^3} + \frac{3K_3(1-f)}{(z_1+z_3)z_5} \\ \frac{z_6}{z_5^3} + \frac{3K_3 f}{z_1 z_3} \end{Bmatrix} \quad (1.35)$$

THE MATLAB SOLUTION OF THE 3D ENVELOPE EQUATION

Proceeding in almost the exact same way as in sections I and II, the code for both the main program (*Envelope3D.m*) and the function that returns the state derivative (*prime3D.m*) are given in Figures 9 and 10 respectively. Note that I have absorbed the factor of $5\sqrt{5}$ into the definition of K_3 in the Matlab code.

```

1. % 3D Elliptical bunched beam envelope equation with space charge.
2. %
3. %---constants
4. %
5. c=3e8; % speed of light (m/s)
6. mc2=511; % electron rest mass (keV)
7. %---Global beam parameters
8. %
9. f=2856e6; % frequency (Hz)
10. Qbunch=3e-9; % beam charge (C)
11. Wkin=7; % kinetic energy (MeV)
12. distance=1.0; % length of drift
13. %---Transverse phase space
14. %
15. initXr=0.001; initXrp=0.0; % initial RMS X size (m) and RMS X slope (rad)
16. initYr=0.002; initYrp=-0.0015; % initial RMS Y size (m) and RMS Y slope (rad)
17. emitXrn=1.0e-6; emitYrn=1.0e-6; % normalized X and Y RMS emittances (m*rad)
18. %---Longitudinal phase space
19. %
20. initZr=0.001; % initial RMS bunch length (m)
21. initdWbyWrParm=0.0115; % initial RMS energy spread dW/W (rad)
22. emitZrnParm = 68.457; % Parmela: normalized longitudinal emittance (deg-keV)
23. %---Prepare data for ode solver
24. %
25. gamma=1+Wkin/0.511; % relativistic mass
26. emitZrn=emitZrnParm*(c/(360*mc2*f)); % normalized RMS z emittance (m*rad)

```

```

27. initZrp=initdWbyWrParm/(gamma*(1+gamma)); % init rms Z prime (rad) [Allen, Eqn. 6]
28. sRange=(0:0.1:distance); % range for plots
29. XYZvector0 =[initXr, initXrp, initYr, initYrp,initZr,initZrp]; %Init. Conditions
30. %---solve envelope equation
31. %
32. tic
33. [s,XYZvector]=ode45('prime3D',sRange,XYZvector0,[],...
                       emitXrn,emitYrn,emitZrn,Qbunch,Wkin);
34. toc % time to solve in seconds
35. %---prepare data for plots
36. %
37. Xsize=1000*XYZvector(:,1);
38. Xslope=1000*XYZvector(:,2);
39. Ysize=1000*XYZvector(:,3);
40. Yslope=1000*XYZvector(:,4);
41. Zsize=1000*XYZvector(:,5);
42. Zslope=1000*XYZvector(:,6);
43. s=1000*s;
44. %---plot results
45. %
46. subplot(2,1,1)
47. plot(s,Xsize,'o-',s,Ysize,'+',s,Zsize,'*-')
48. xlabel('s [mm]'); ylabel(' Size [mm]');
49. legend('X envelope', 'Y envelope', 'Z envelope');
50. axis([-inf, inf, 0, inf])
51. text(5,0.27,'The Beam Envelope Size')
52. subplot(2,1,2)
53. plot(s,Xslope,'o-',s,Yslope,'+',s,Zslope,'*-')
54. xlabel('s [mm]'); ylabel(' Slope [mrad]');
55. legend('X"', 'Y"', 'Z"');
56. text(5,2,'The Beam Envelope Slope')

```

Figure (9). *Envelope3D.m* (Matlab m-file for Solving Eqn. 1.17)

```

1. function zprime3D = prime3D(t,z,dummy,enXr,enYr,enZr,Q,W);
2. %Calling syntax: zprime3D = prime3D(t,z,dummy,enXr,enYr,enZr,l,lambda,W);
3. %
4. %Inputs:
5. % t = distance[m];
6. % z = [z(1); z(2); [X size (m); X slope (rad);
7. % z(3); z(4); Y size (m); Y slope (rad);
8. % z(5); z(6)] Z size (m); Z slope (rad)]
9. % dummy = this place holder is needed for 'options' in ode45
10. % enXr = normalized X RMS emittance (m*rad)
11. % enYr = normalized Y RMS emittance (m*rad)
12. % enZr = normalized Z RMS emittance (m*rad)
13. % Q = bunch charge (C)
14. % W = kinetic energy (MeV)
15. %

```

```

16. %
17. %Output: zprime3D=[z(2); (eXr^2)/(z(1))^3 + (3*K3*(1-f)) / ( (z(1)+z(3))*z(5) );
18. %                z(4); (eYr^2)/(z(3))^3 + (3*K3*(1-f)) / ( (z(1)+z(3))*z(5) )
19. %                z(6); (eZr^2)/(z(5))^3 + (3*K3*f) / ( z(1)+z(3) )
20. %
21. c=3e8;
22. IA = 17000;           % Alfven Current (Amps)
23. gamma=1+W/0.511;     % relativistic mass
24. beta=sqrt(1-1/gamma^2); % normalized velocity
25. bg=gamma*beta;
26. %
27. eXr=enXr/gamma*beta; % unnormalized X emittance; units = m*rad
28. eYr=enYr/gamma*beta; % unnormalized Y emittance; units = m*rad
29. eZr=enZr/gamma*beta; % unnormalized Z emittance; units = m*rad
30. K3=(Q*c)/(5*sqrt(5)*IA*beta^2*gamma^3); % generalized perveance [Allen,
31. %create a LookUp Table using Table III from the Trace-3D manual
32. %
33. pLUT=[0:0.05:0.95,1./[1:-0.05:0.05],1000];
34. fLUT=[1,0.926,0.861,0.803,0.750,0.704,0.661,0.623,0.588,0.556,0.527,0.500, ...
35. 0.476,0.453,0.432,0.413,0.394,0.378,0.362,0.347,0.333,0.320, ...
36. 0.306,0.291,0.276,0.260,0.244,0.227,0.210,0.192,0.174,0.155, ...
37. 0.135,0.115,0.095,0.075,0.056,0.037,0.020,0.007,0];
38. p=gamma*z(5)/sqrt(z(1)+z(3));
39. f=interp1(pLUT,fLUT,p);
40. %
41. zprime3D=[z(2); (eXr^2)/(z(1))^3 + (3*K3*(1-f)) / ( (z(1)+z(3))*z(5) );
42.           z(4); (eYr^2)/(z(3))^3 + (3*K3*(1-f)) / ( (z(1)+z(3))*z(5) );
43.           z(6); (eZr^2)/(z(5))^3 + (3*K3*f) / ( z(1)+z(3) )];

```

Figure (10). *prime3D.m* (Matlab function to calculate the state derivative.)

```

>>Envelope3D
>>

```

Figure (11). Text to type at the Matlab prompt.

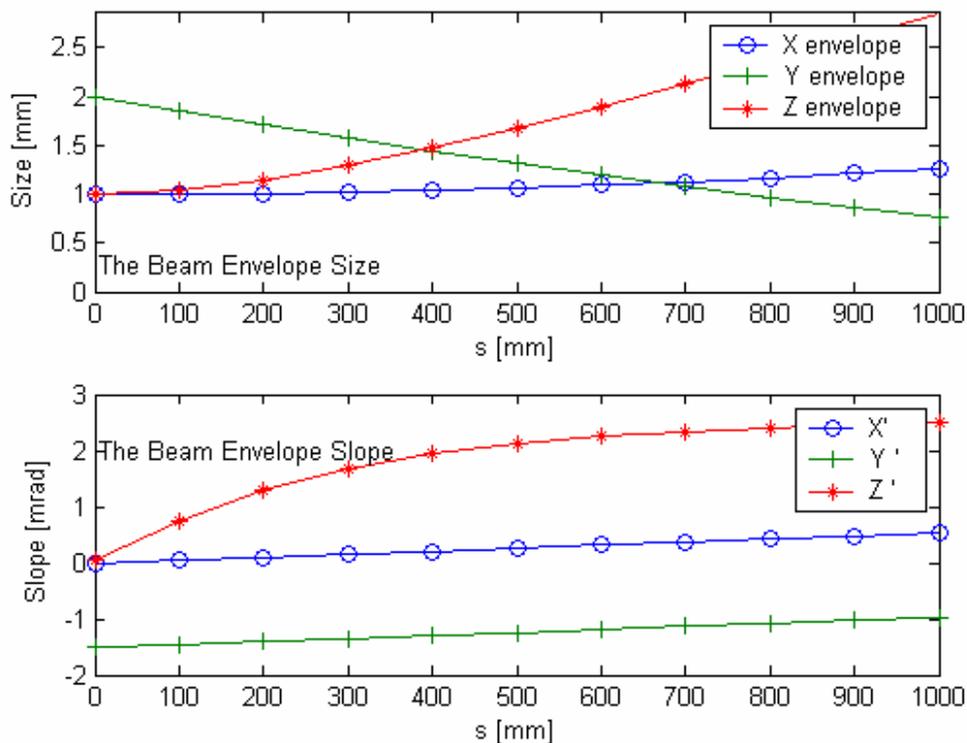


Figure (12). Numerical solution of the beam coupled beam envelope equation [1. 19-1.21]. The top trace is the beam size vs. distance (s) and the bottom trace is the beam slope vs. distance (s). The tic-toc commands on lines 32 & 34 of Envelope3D show that the problem took 80 msec to solve.

[IV] Future Development

With these envelope solvers in hand it is now possible to use them to do the Chi-squared fitting for a quad scan. The basic idea would be to call the line in the code that contains the ode45 solver, pass to it some number of fitting parameters, subtract the relevant values from the measured data, and then minimize this result.

[IV] Conclusion

The envelope equation for a beam in a drift for the cases of three cases has been solved. As stated in the introduction, these codes have been developed to eventually use as a part of a modified Quad Scan technique to measure the emittance of space charge dominated beams. However, the beam envelope equations in this paper have the limitation that they do not allow for non-zero correlations between different phase planes. It may turn out to be better for the Quad Scan problem to take a beam matrix approach (as in Reference 9) since it is easy to include correlations. However, this is the subject for a future work.

The next step would be to solve the envelope equation including the focusing term of a Quad to enable one to do the modified quad scan.

[V] References

- [1] MATLAB, The MathWorks inc., see www.mathworks.com
- [2] R. Pratap, “*Getting Started with Matlab 5*”[, Section 5.5, Ordinary Differential Equations
- [3] D. Hanselman and B. Littlefield, “*Mastering Matlab 6*”
- [4] Bronson, R., *Matrix Methods* Chpt. 7
- [5] Reiser, M., “*Theory and Design of Charged Particle Beams*”, Wiley
- [6] Wangler, T., “*RF Linear Accelerators*”, Wiley
- [7] Chao, A., “*Physics of collective Beam Instabilities in High Energy Accelerators*”
- [8] Trace-3D documentation
- [9] C.K. Allen, N.D. Pattengale, “Theory and Technique Of Beam Envelope Simulation: Simulation of bunched Particle Beams with Ellipsoidal Symmetry and Linear Space Charge Forces”, LA-UR-02-4979 (LANL report)