A simple numerical algorithm for obtaining the longitudinal phase space density from the time projection of the current

Alvin V. Tollestrup

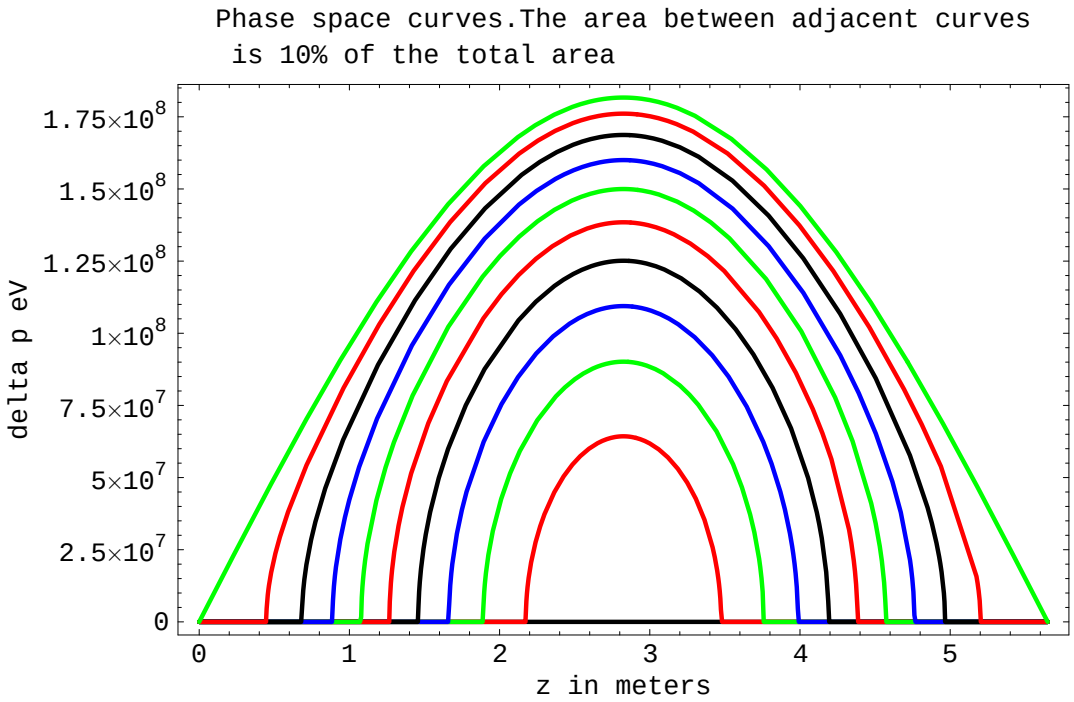
March 1, 2003

Abstract

The purpose of this note is to outline a simple numerical algorithm to obtain the phase space density from the observed time dependence of the proton bunch current. It applies only to equilibrium distributions. It is implemented using Mathematica, but can be easily transformed to other platforms.

Approach

Consider the longitudinal phase space plot shown in the figure below.



We know from Liouville's Theorem that the density along any of the trajectories shown will be constant. Thus if we wish the density at $\rho(z,p)$, it can be obtained from $\rho(z_0,p_0)$ where z_0 is at the synchronous phase, $\lambda/2$, and p_0 is on the same trajectory as (z,p). This is just a reflection of the fact that for an equilibrium distribution, there is only one variable on which the density

depends. The natural variable to pick is the "action" or the area of the associated phase space "ellipse".

The connection between the two points is easy to calculate because the Hamaltonian is constant:

$$p^2 + \frac{2v^2 E_s e V_{rf}}{\eta h c^2} Cos(2\pi z/\lambda) = p_0^2 + \frac{2v^2 E_s e V_{rf}}{\eta h c^2} Cos(2\pi z_0/\lambda) \tag{1.1}$$

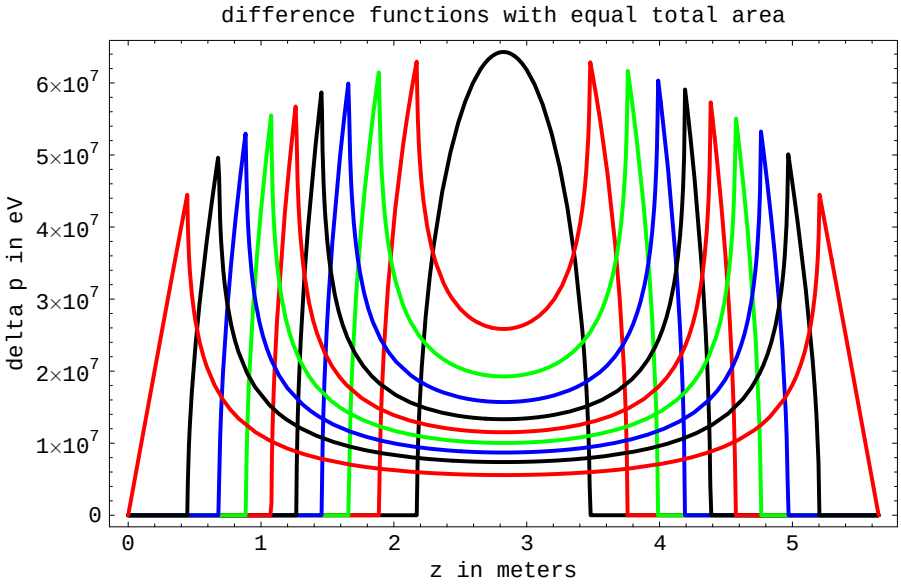
where p is the momentum deviation, λ is the wavelength, E_s is the energy, η is the slip factor, h is the harmonic number and V_{rf} is the cavity voltage. Applying this to the case where z_0 is on the center line:

$$p_0^2 = p^2 + \frac{2v^2 E_s e V_{rf}}{\eta h c^2} (Cos(2\pi z/\lambda) + 1) \tag{1.2}$$

Thus the problem of finding the density function becomes one of finding the density along the p axis at the synchronous phase.

The expansion functions

The figure above was drawn with the area difference between curves 10% of the total area or in steps of 10% of the action. Approximately the density will be constant in these bands. This leads us to propose using these difference functions as a set of functions in which to expand the proton current. Since the density is constant, the total number of protons in a slice of z is just proportional to the height of the individual curves. The difference functions shown below correspond to the first figure.



The figure shows the case for ten curves, but in fact we will use of the order of fifty. The curves have equal areas and so we need a recipe to calculate the curves, ie we need to associate each curve with its action which is the average value of the action for the two bounding phase space curves. We need a formula that connects the action S with its corresponding phase curve. But the phase space curve is completely determined by specifying the value of p_0 on the central axis. So the relation we seek is given by using eq.1.2 above:

(1.3) $$S = \int \sqrt{p_0^2 - \frac{2v^2 E_s e V_{rf}}{\eta h c^2}(Cos(2\pi z/\lambda)+1)}\,dz$$

This is an elliptic integral that is easily done and gives $S(p_0)$. In a practical sense it easiest to integrate over z from zero to λ and take the real part after the integral is done. However, the relation we have has to be inverted so that we have $p_0(S)$ so that we can generate the equally spaced curves inaction, S. This is done easily by making a numerical table of S vs p_0 , and from this table generate an interpolation function that gives $p_0(S)$. Once this function is known, the curves are easily generated.

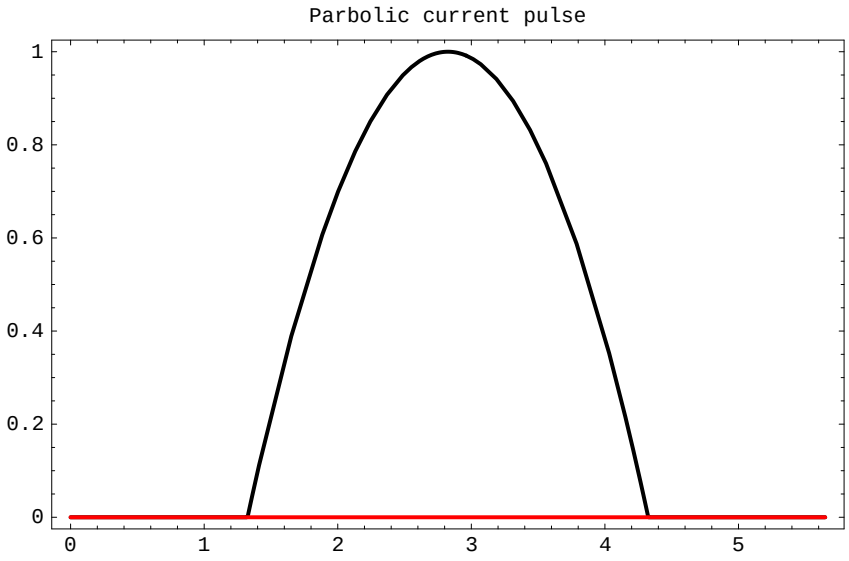
If we designate the difference functions by $F_i(z)$, then we have

(1.4) $$I_p(z) = \sum_i C_i F_i(z)$$

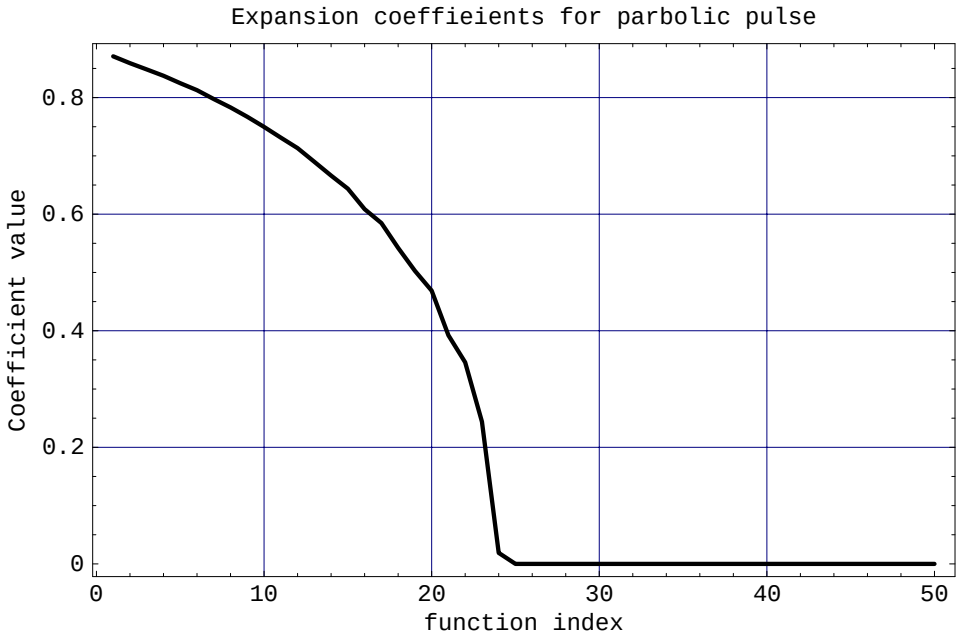
The C_i then give the density as a function of the index i which is directly related to the action and the density along the central axis. The constants are made by breaking up the proton current pulse into a number of discrete points and making a least squares fit using eq. 1.4.

A toy example

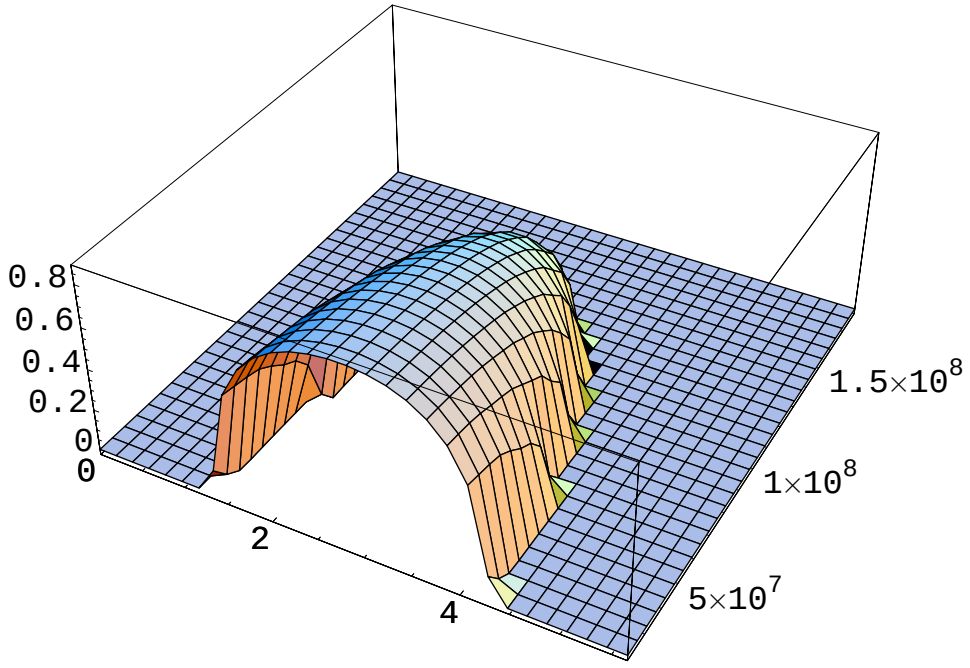
As an example, we take a parabolic pulse that has a base width of about half the bucket width. This is shown in the figure below. We use a set of expansion functions that are 2% of the total phase space area, and we have broken up the first half of the pulse into 50 points. Note that it is necessary to have more points than fitting functions. We will come back to this point later.

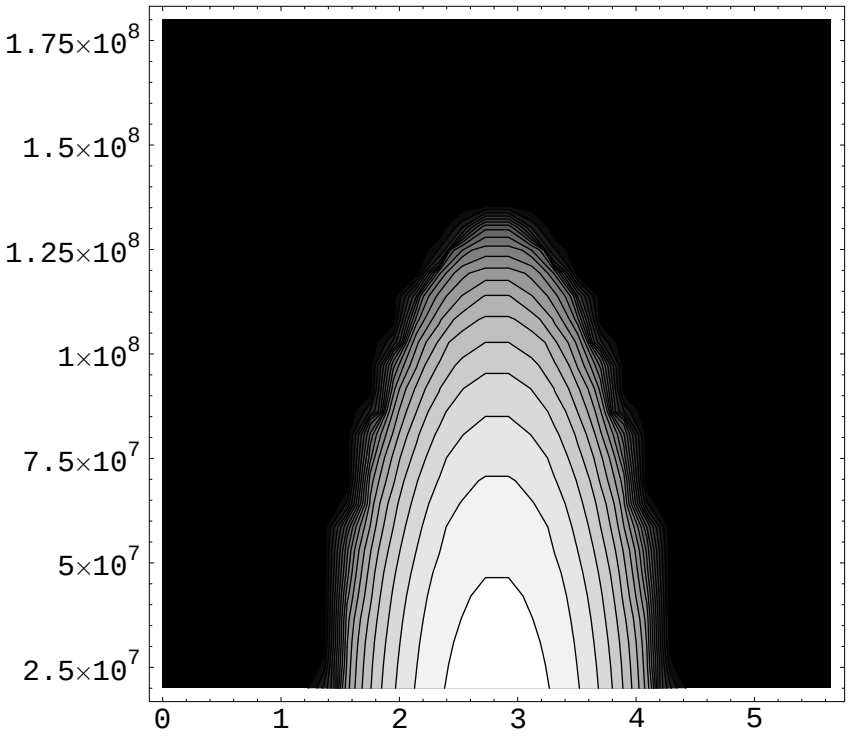


The expansion yields the following coefficients:



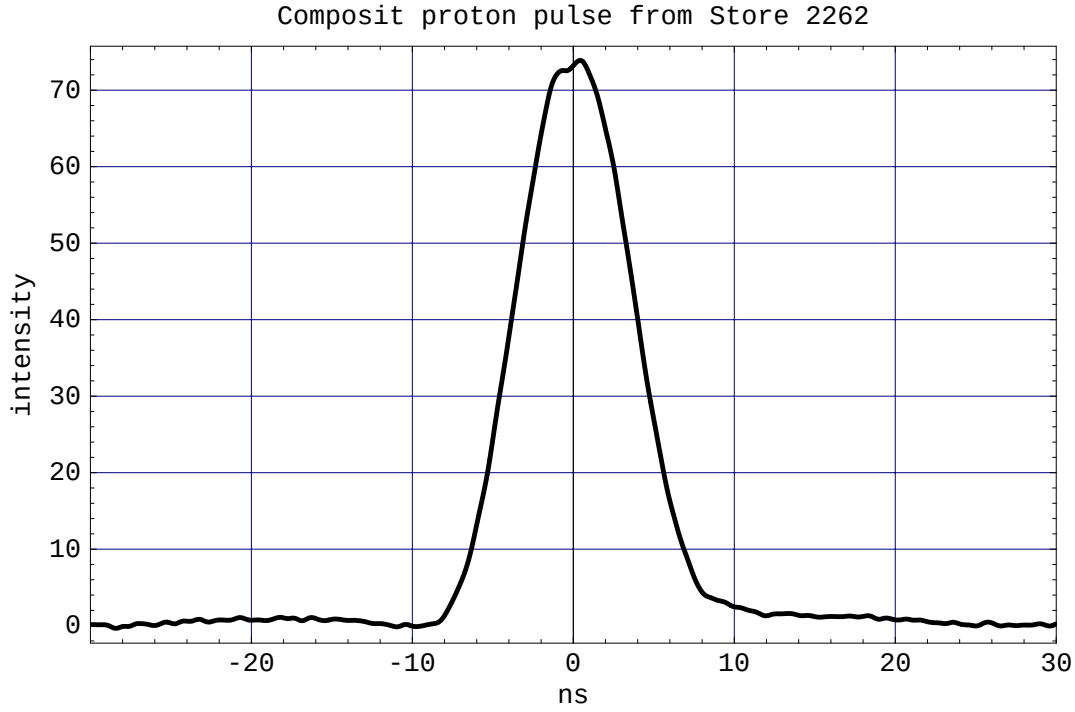
None of the higher order functions are necessary because there is no current in the region from z=0 to about 1.3 meters. The following two figures show the density as a 3D plot and as a contour plot. A plot of the input parabola and the output fitted function agree to about 1% in this case. In fact plotting the two curves on top of each other yields only a single line.


0.8
0.6
0.4
0.2
0
0
2
4
1.5×10^8
1×10^8
5×10^7


1.75×10^8
1.5×10^8
1.25×10^8
1×10^8
7.5×10^7
5×10^7
2.5×10^7
0
1
2
3
4
5

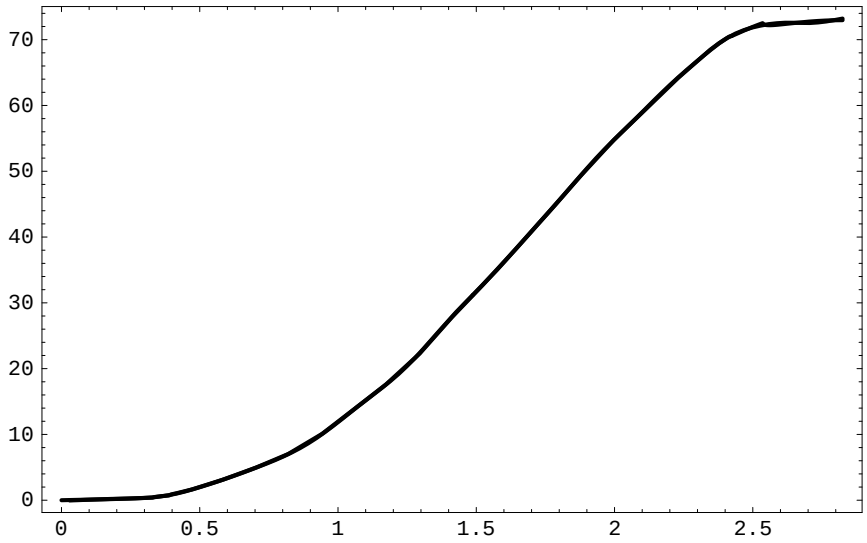
Composit Tevatron proton pulse at 150 GeV/c

As a second example, we use data from the Tevatron at injection. The pulses are wide, nearly filling the available bucket. The wall current monitor digitizes at 0.5 ns intervals which yields about 37 points per bucket. It is clear that none of the 36 proton pulses represent a phase space that is in complete equilibrium. There are local blobs that move around with the synchrotron frequency. An ensemble is generated by superimposing all 36 bunches on top of each other. The technique used was to fit each of the 36 bunches with a local Gaussian to get the centroid. Next, a local interpolation (3^{rd} order) function was generated for each of the 36 pulses. Using the time from the Gaussian fit, these bunches were superimposed to give the ensemble. (Note: this is not a superposition of Gaussians! The Gaussian fit is used only to get the center to an accuracy of about 50 ps or less. This time was then used in the interpolation functions to superpose the pulses.) The result is shown below.

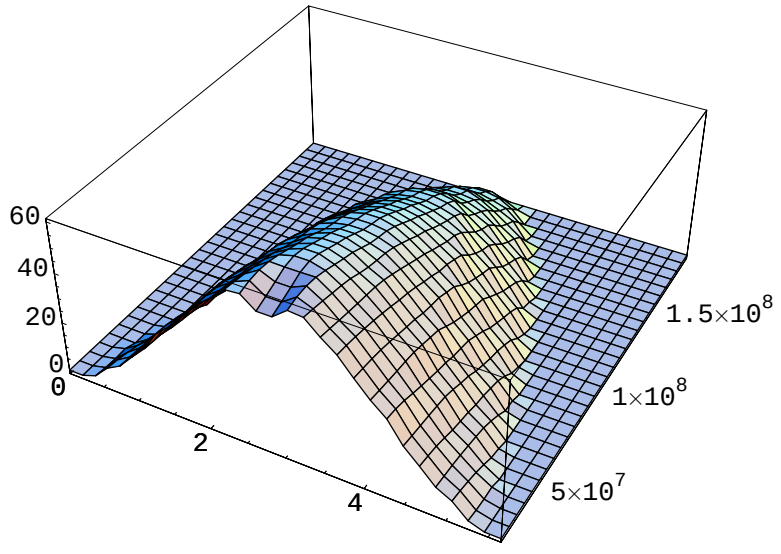


There is a small satellite at -19 ns,in the adjacent bucket, and a combination of a dispersive tail due to the cable from the tunnel to F0 and perhaps a small following satellite. The dispersion in the cable can be corrected, but for this study, I have taken just the leading half of the pulse. {It is clear that there is not complete statistical equilibrium because one sees a small bump on the right side of the peak). Since this pulse is generated by adding together 36 interpolation functions, it is continuous, and can be used to generate 50 data points in the region from -9.4 ns to 0.0. These points are then fit with our 50 F_i. The figure below shows the original half pulse and the fitted

function superimposed. The horizontal scale has been changed to meters from the ns scale in the above figure. The fit is quite good. The 3D plot is also shown.

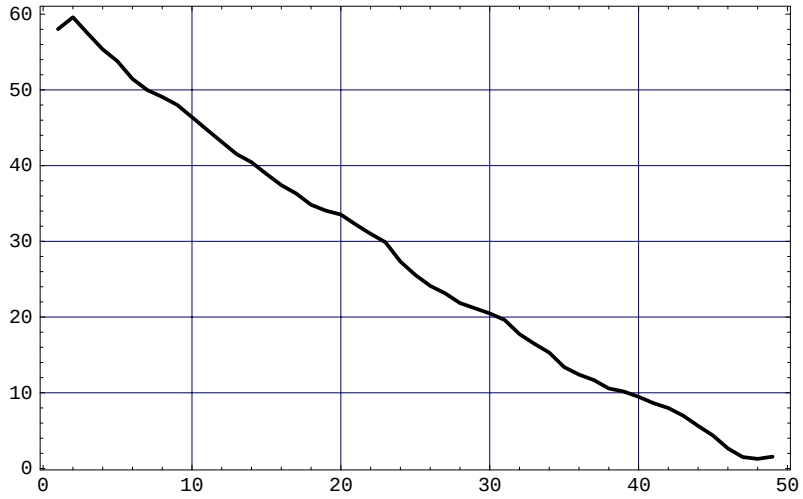


The original current (black) and the fit {blue) are shown in the figure above. The figure below is the reconstructed phase space.

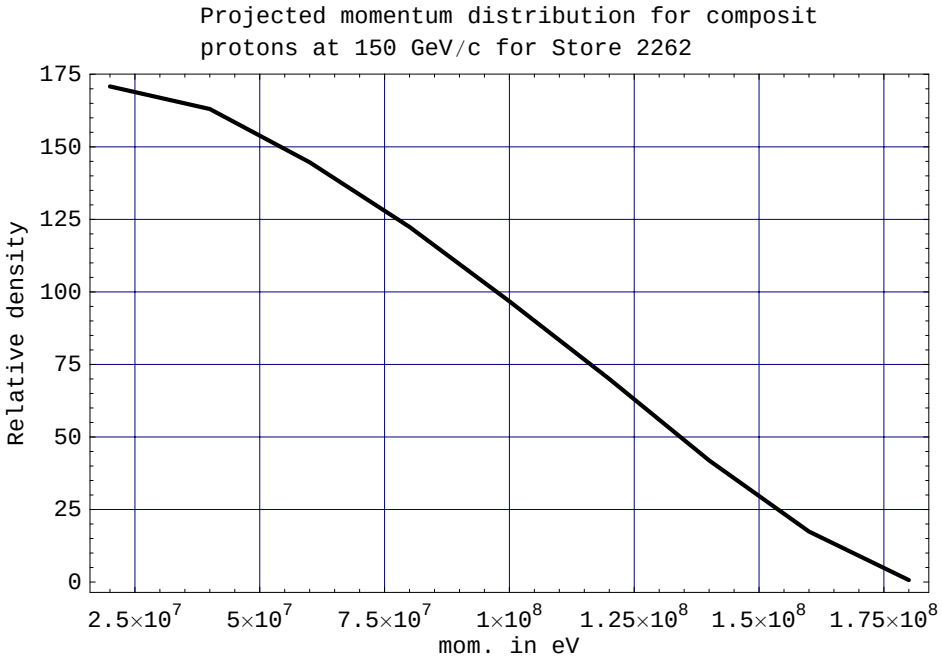


Notice the small dimple at the peak. This comes from injection errors. When a pulse is injected off center, it filaments into a ring leaving a hole in the center. The figure above indicates that on average there was an injection error for the ensemble. This is consistent with the current pulse which has a small dimple just after the center.

The coefficients are:



Given the coefficients, one can project the distribution along the z axis to get the momentum distribution.



This is less like a Gaussian than is the current pulse, and has a sigma that is smaller than would be predicted using a linear relation between z and p. Note that this reflects in the transverse emittance calculation where the longitudinal size is subtracted in quadrature from the measured total sigma.

Conclusion

After the expansion functions and the interpolation functions are set up, the method is very fast as it only requires one least squares fit to obtain the answer. It gives limited information about the momentum distribution from the pulse shape. Good information for the input is necessary which the wall monitor can supply. The technique will be to take 20 to 30 sweeps and superimpose these to give good information on each of the 36 bunches. The present scope gives a t_0 for the sweep that furnishes a precise way to superimpose multiple pulses without a lot of digital processing. (The use of a Gaussian fit as described above is not necessary when t_0 is available. It was not available for the measurements described here.)

The use of numerical methods and interpolation functions greatly simplifies the problem and makes its solution transparent. Leo Michelotti [2] and Cheung-Yang Tan [1] have both written notes on this subject which are more analytical and make useful reading. The first mentioned note has a rather complete list of references and some discussion of the historical background.

The Mathematica program is shown in the Appendix. An example using the data from Store 2328 is given. In this case the dispersion of the cable has been removed before the fitting.

References

1. C.-Y Tan A method for calculating longitudinal phase space distribution when given the time profile of the bunch. Technical Memo FERMILAB-TM-2155 July 2001.
2. Leo Michelotti. An integral for longitudinal phase space tomography on equilibrium distributions. FERMILAB-FN-0726 Oct. 2002.
3. There is an additional note being prepared by Leo Michelotti in draft dated March 31, 2003.

Appendix. Mathematica Program and example.

Example using data from Store 2328

```
In[21]:= Remove["Global`*"]
```

```
In[1]:= Date[]
```

```
Out[1]= {2003, 4, 7, 17, 54, 13}
```

```
In[2]:= Initialization
```

```
Out[2]= Initialization

        Initialization
```

```
In[3]:= so[tg_] :=
         {{SetOptions[Plot, Frame -> True, Axes -> False,
            GridLines -> None,
            PlotStyle -> {{Thickness[tg], RGBColor[0, 0, 0]},
              {Thickness[tg], RGBColor[1, 0, 0]}, {Thickness[tg], RGBColor[0, 1, 0]},
              {Thickness[tg], RGBColor[0, 0, 1]}}, ImageSize -> {400, 280}]},
          {SetOptions[ParametricPlot, Frame -> False, Axes -> True,
            GridLines -> None,
            PlotStyle -> {Hue[0], RGBColor[0, 0, 1]}, ImageSize -> {400, 280}]},
        {SetOptions[ListPlot, Frame -> True,
            PlotRange -> All, PlotJoined -> True, GridLines -> Automatic,
            PlotStyle -> {Thickness[tg], RGBColor[0, 0, 0]}, ImageSize -> {400, 300}]},
        {Off[General::spell]},
        {Off[General::spell1]}}
```

```
In[4]:= so[.005];
```

```
In[22]:= << Statistics`NonlinearFit`;
         << Statistics`DataManipulation`
         << NumericalMath`ListIntegrate`;

        SetDelayed::write : Tag ListIntegrate in ListIntegrate[cl_?VectorQ, h_, k_Integer : 3] is Protected.

        SetDelayed::write : Tag ListIntegrate in ListIntegrate[cl_?MatrixQ, k_Integer : 3] is Protected.
```

```
In[25]:= pl[f_, a_, b_, c_, d_, labl_, color_, tg_] := ListPlot[f, PlotRange → {{a, b}, {c, d}},
            PlotLabel → labl, PlotStyle → {Thickness[tg], colr[color]}];
         pl[f_, a_, b_, c_, d_, labl_, color_] := ListPlot[f, PlotRange → {{a, b}, {c, d}},
            PlotLabel → labl, PlotStyle → colr[color]];
         pl[f_, a_, b_, c_, d_, labl_] := ListPlot[f,
            PlotRange → {{a, b}, {c, d}}, PlotLabel → labl];
         pl[f_, a_, b_, c_, d_] := ListPlot[f, PlotRange → {{a, b}, {c, d}}];
         colr[m_] :=
           {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1], RGBColor[0, 0, 0]}[[m]];
         f[x_, n1_, n2_] := If[x[[1]] ≥ n1 && x[[1]] ≤ n2, True, False];
         sel[ff_, n1_, n2_] := Select[ff, f[#, n1, n2] &];
```

```
In[36]:= Co = 2.9979 10^8; h = 1113; eta = 1 / 18.7^2; tau = 2 * Pi * 1000. / Co;
         Frf = h / tau; Vrf = 1.1 10^6; lambar = 3 10^8 / (2 * Pi * Frf);
         CC[Edot_, z0_] := Cos[z0 / lambar] + z0 / lambar * Sin[Pi - Edot * tau / Vrf] ;
         Emax[Es_] := Sqrt[2 * Es * Vrf / (Pi * eta * h)];
         phiS[Edot_] := Edot * tau / Vrf; zs[Edot_] := lambar * phiS[SEdot];
         Area[Es_] := 16 * Sqrt[Vrf * Es / (2 * Pi * h * eta)] / (2 * Pi * Frf);
         krf[Es_] := Es * Vrf / (Pi * eta * h); nus[Es_] := Sqrt[eta * h * Vrf / (2 * Pi * Es)];

In[39]:= dir = "D:/new/Scratch/orbits/"; dir2 = "G:/Tev2003/StoresByNumber/2328/";
         dir3 = "G:/Tev2003/Results/z phase space/2DspaceEPS/";

       Export[dir3 <> "fig 1.EPS", pl1]

       Export::type : pl1 cannot be exported to the EPS format.

       $Failed
```

Note: In the equations above, z0 is in meters. Emax is the max energy variation on the separatrix. deltaPhi is measured from Pi. The following equation relates the oscillation energy to z for a particle strating at z0.

```
In[40]:= delE[z_, z0_, Edot_, Es_] :=
          Re[Sqrt[ krf[Es] * (CC[Edot, z0] - Cos[z * 2 Pi] - z * 2 * Pi * Sin[Pi - Edot * tau / Vrf])]]
```

Note: Above z0 still in meters, but z is in bucket fraction, 0 to 1. Below: delE2 gives the energy of a particle with initial coordinates zi, Ei that is now at z in meters. Edot has been turned off so we are dealing with static orbits.

```
In[41]:= delE2[z_, zi_, Ei_, Es_] := Sqrt[Ei^2 + krf[Es] (Cos[zi / lambar] - Cos[z / lambar])]
```

delE2 is used to relate two points in phase space. delE is used to find trajectory given starting energy is zero and one is at zi.

```
In[42]:= {krf[150 10^9]^.5, 1 / nus[150 10^9], 1 / nus[980 10^9], Emax[150 10^9], Emax[980 10^9]}

Out[42]= {1.28458×10^8, 518.84, 1326.17, 1.81667×10^8, 4.64348×10^8}
```

Energy vs Area at $\lambda/2$ at 980 and 150

■ 150 GeV

We now construct two functions at 150 and 980 GeV. The first relates the Area to the energy at z= 0.5, A=f[E]. A table of values is made and an interpolation function is generated. The table is reversed and and a function E=F[A] is generated which is the inverse of the first. These interpolation functions are continuous functions.

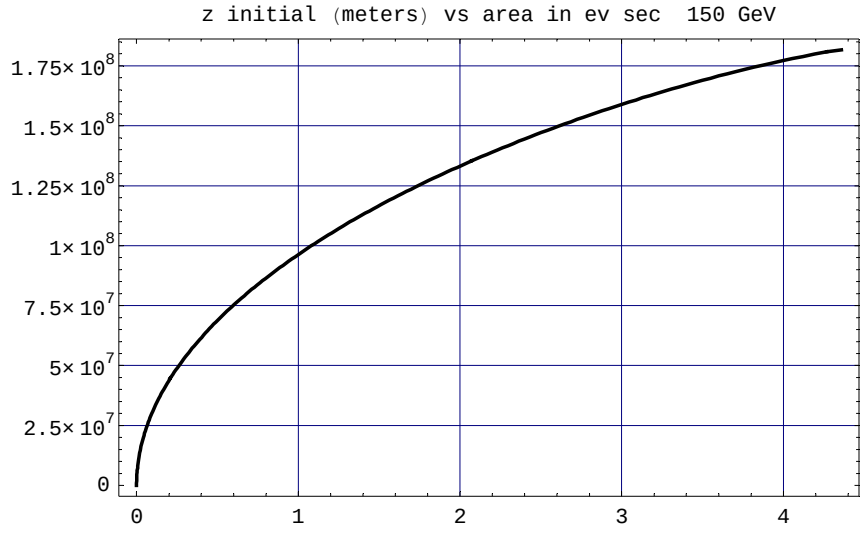
```
In[43]:= AreaEx[Ei_, Es_] := 4 * Integrate[delE2[z, Pi * lambar, Ei, Es], {z, 0, Pi lambar}] / Co

In[44]:= {AreaEx[Emax[150 10^9] - 1, 150 10^9], Area[150 10^9]}

Out[44]= {4.35871 - 3.76879×10^-8 i, 4.35566}
```

```
In[45]:= AreaList150 = Table[{Re[AreaEx[x, 150 10^9]], x},
           {x, 100, Emax[150 10^9] + 1000, (Emax[150 10^9]) / 200}];
```

The small lower limit is to avoid a complex infinity in the table and the 1 kv added to the upper limit gets the last interval included

```
In[46]:= Length[AreaList150]

Out[46]= 201
```

```
In[47]:= pl1 = ListPlot[AreaList150,
           PlotLabel -> "z initial (meters) vs area in ev sec  150 GeV"]
```




```
Out[47]= - Graphics -
```

```
In[49]:= E150[Area_] = Interpolation[AreaList150][Area]

Out[49]= InterpolatingFunction[{{1.03669×10^-12, 4.35873}}, <>][Area]
```

E2 is the Energy at Pi vs area in eV-sec

```
In[50]:= E150[4.3587]

Out[50]= 1.81667×10^8
```

```
In[51]:= inverAreaList150 = Table[{AreaList150[[m, 2]], AreaList150[[m, 1]]}, {m, 1, 200}];

In[52]:= Ar150[E3_] = Interpolation[inverAreaList150][E3]

Out[52]= InterpolatingFunction[{{100., 1.80759×10^8}}, <>][E3]
```

Ar[E] is the area associated with the energy at z =Pi lambar. The above two functions connect area and energy on the center line.

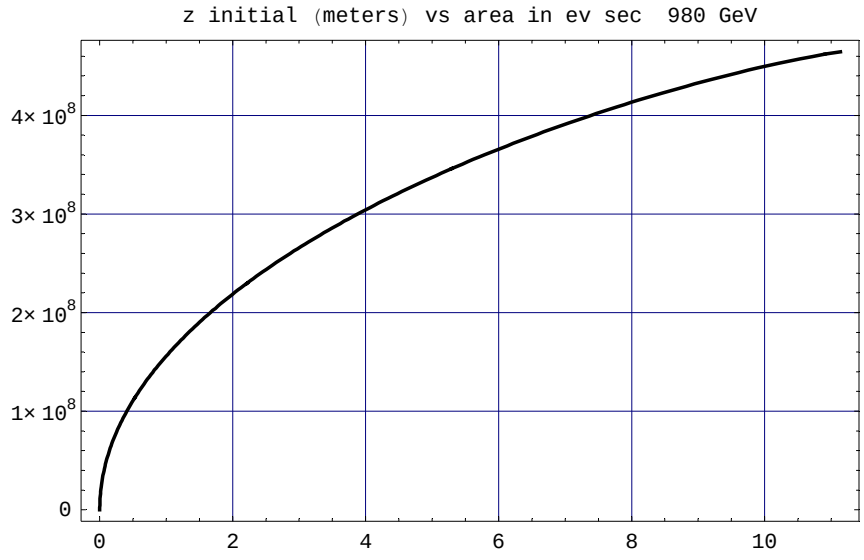
■ 980 GeV

```
In[53]:= AreaList980 = Table[{Re[AreaEx[x, 980 10^9]], x},
           {x, 100, Emax[980 10^9] + 1000, (Emax[980 10^9]) / 200}];
```

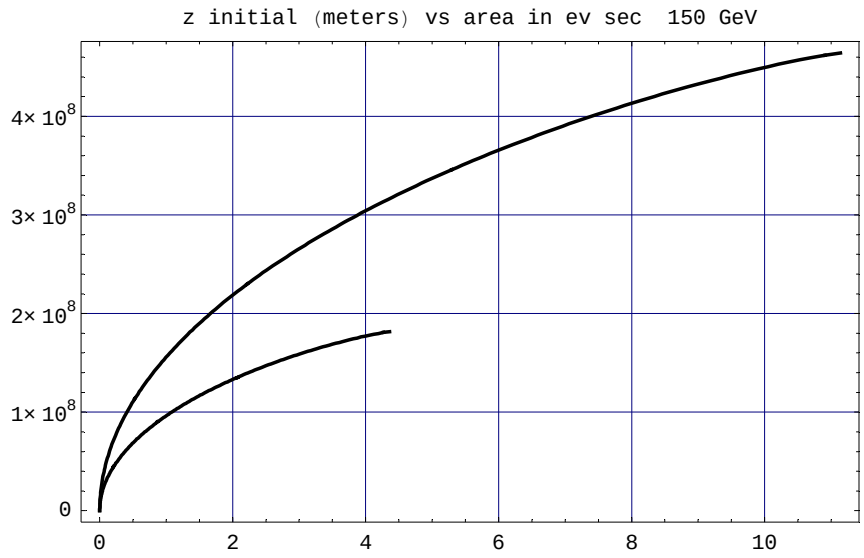
The small lower limit is to avoid a complex infinity in the table and the 1 kv added to the upper limit gets the last interval included

```
In[54]:= Length[AreaList980]
Out[54]= 201
```

```
In[55]:= pl2 =
           ListPlot[AreaList980, PlotLabel -> "z initial (meters) vs area in ev sec  980 GeV"];
```



In[56]:= **pl3 = Show[pl1, pl2]**



Out[56]= **-** Graphics **-**

In[57]:= **E980[Area_] = Interpolation[AreaList980][Area]**

Out[57]= InterpolatingFunction[{{4.06238×10^-13, 11.141}}, <>][Area]

In[58]:= **inverAreaList980 = Table[{AreaList980[[m, 2]], AreaList980[[m, 1]]}, {m, 1, 200}];**

In[59]:= **Ar980[E3_] = Interpolation[inverAreaList980][E3]**

Out[59]= InterpolatingFunction[{{100., 4.62026×10^8}}, <>][E3]

Ar[E] is the area associated with the energy at z =Pi lambar. The above two functions connect area and energy on the center line.

Fitting functions

■ 980 Gev

■ Set up 50 basis fubctions at 980

Note: The number of fitting functions can be set in the following tables. We will use 50 in for 150 and 980 GeV. The best number needs to be determined.

In[62]:=
```
func1 = Table[{delE2[z, Pi * lambar, E980[Area], 980 10^9], E980[Area]},
    {Area, .0, AreaList980[[201, 1]], AreaList980[[201, 1]] / 50}];
```

```
In[63]:= Length[func1]

Out[63]= 51
```

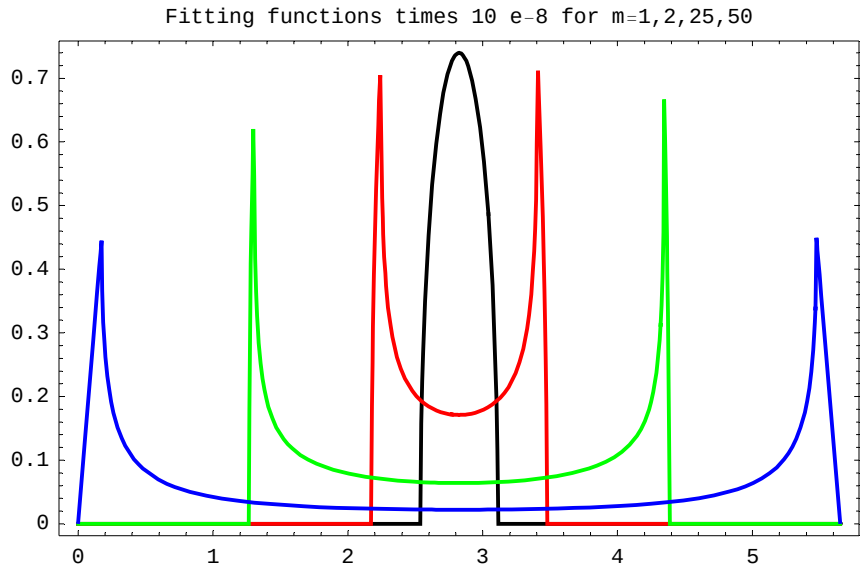
func1 is a set of 101 elliptical curves differing by 1% in area. func2 is the difference curve. .

```
In[64]:= func2 = Table[{Re[(func1[[m, 1]] - func1[[m - 1, 1]])] / (1. * 10^8),
            (func1[[m, 2]] + func1[[m - 1, 2]]) / 2}, {m, 2, 51}];

In[65]:= Length[func2]

Out[65]= 50

In[66]:= func2 = Join[func2, {{Area[980 10^9], Emax[980 10^9]}}];
```

Join a last point onto the energy and area to keep interpolation function from starting to diverge before Emax[Es]

func2 is a function of z which is in meters.

```
In[67]:= pl4 =
          Plot[{func2[[1, 1]], func2[[5, 1]], func2[[25, 1]], func2[[50, 1]]}, {z, 0, 2 Pi lambar},
           PlotLabel -> "Fitting functions times 10 e-8 for m=1,2,25,50", PlotRange → All]
```




```
Out[67]= - Graphics -
```

■ 150 GeV

■ Set up 50 basis fubctions

```
In[68]:= func3 = Table[{delE2[z, Pi * lambar, E150[Area], 150 10^9], E150[Area]},
            {Area, .0, AreaList150[[201, 1]], AreaList150[[201, 1]] / 50}];
```

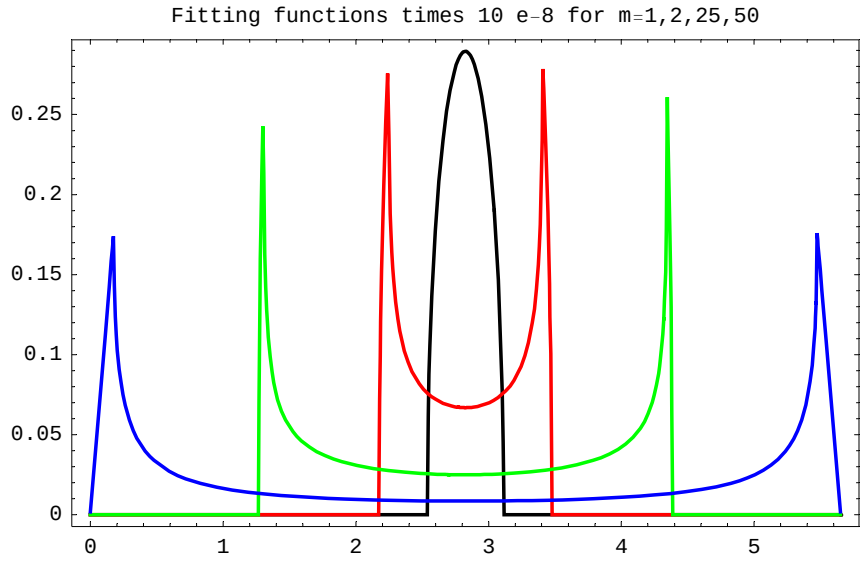
```
In[69]:= AreaList150[[201, 1]]

Out[69]= 4.35873

In[70]:= Length[func3]

Out[70]= 51
```

func3 is a set of 101 elliptically shaped curves differing by 1% in area. .

```
In[71]:= func4 = Table[{Re[(func3[[m, 1]] - func3[[m - 1, 1]])] / (1. * 10^8),
              (func3[[m, 2]] + func3[[m - 1, 2]]) / 2}, {m, 2, 51}];

In[72]:= func4 = Join[func4, {{Area[150 10^9], Emax[150 10^9]}}];
```

Join a last point onto the energy and area to keep interpolation function from starting to diverge before Emax[Es]

```
In[73]:= Length[func4]

Out[73]= 51

In[75]:= pl5 =
          Plot[{func4[[1, 1]], func4[[5, 1]], func4[[25, 1]], func4[[50, 1]]}, {z, 0, 2 Pi lambar},
           PlotLabel -> "Fitting functions times 10 e-8 for m=1,2,25,50", PlotRange → All]
```



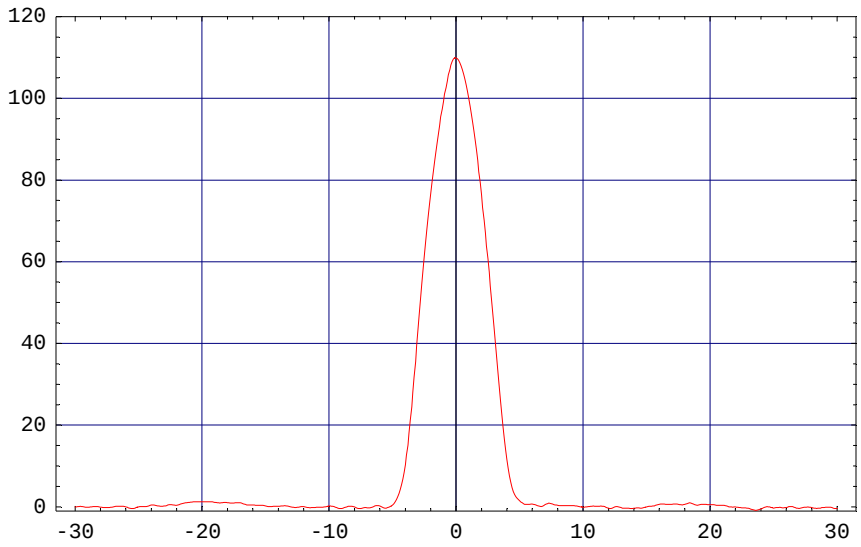

```
Out[75]= - Graphics -
```

Note that z is in meters.

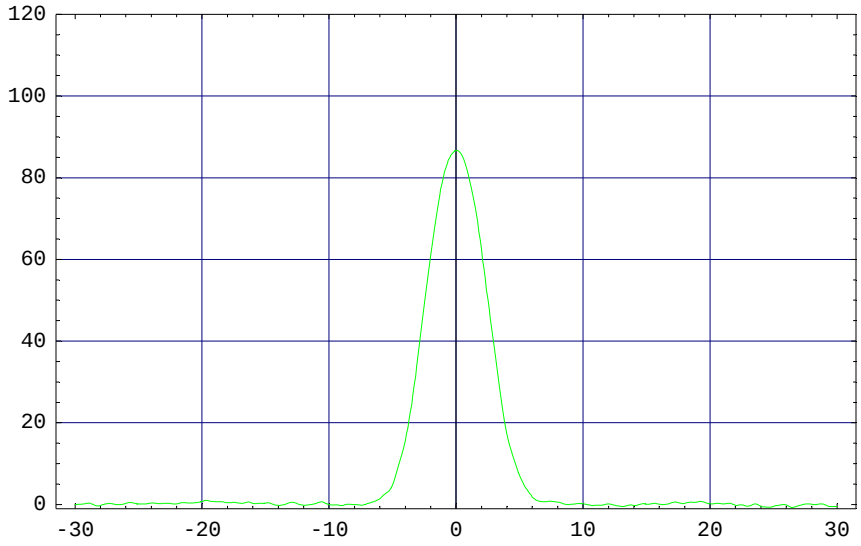
2328 at 980 protons. Three plots: after acceleration, after 9 hours and after 14 hours

We now apply these functions to fit some data in Store 2328. The data are an ensemble of all 36 proton pulses that have been superimposed. The base line has been correct as well as the dispersion in the cable.

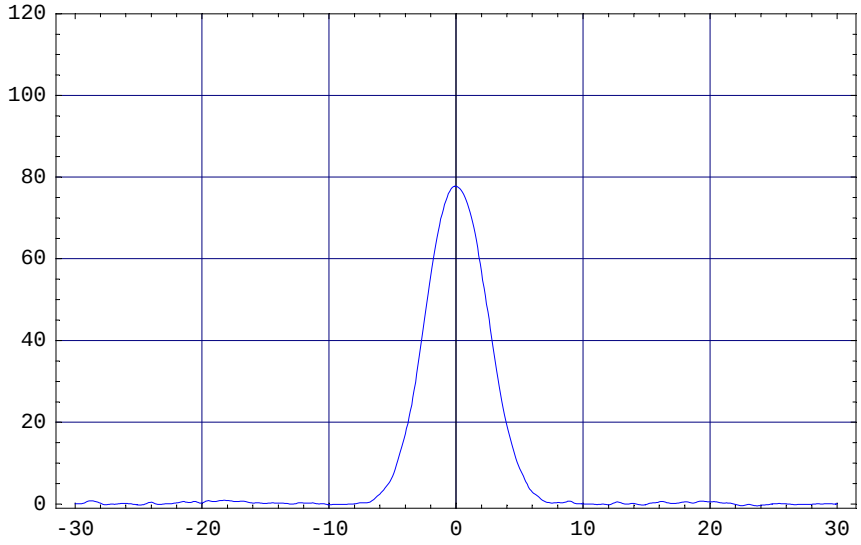
```
In[76]:= pa = << "G:/Tev2003/StoresByNumber/2328/p980a.dat";
         pb = << "G:/Tev2003/StoresByNumber/2328/p980b.dat";
         pc = << "G:/Tev2003/StoresByNumber/2328/p980c.dat";

In[77]:= pl6 = ListPlot[pa, PlotRange → {All, {-1, 120}}, PlotStyle → RGBColor[1, 0, 0]]
```



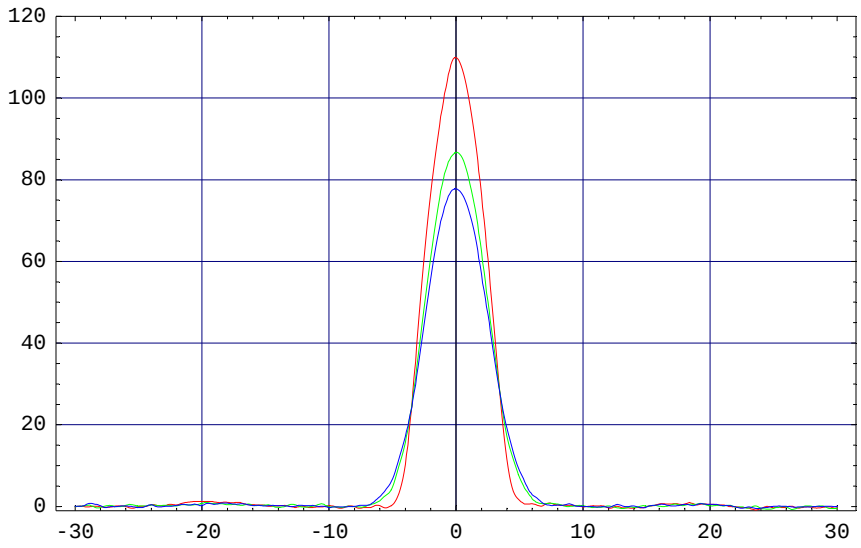

```
Out[77]= - Graphics -
```



Out[78]= - Graphics -



Out[79]= - Graphics -



Out[80]= **-** Graphics **-**

The three composit pulses are shown above.

In[81]:=
```
Length[pa]
```

Out[81]= 601

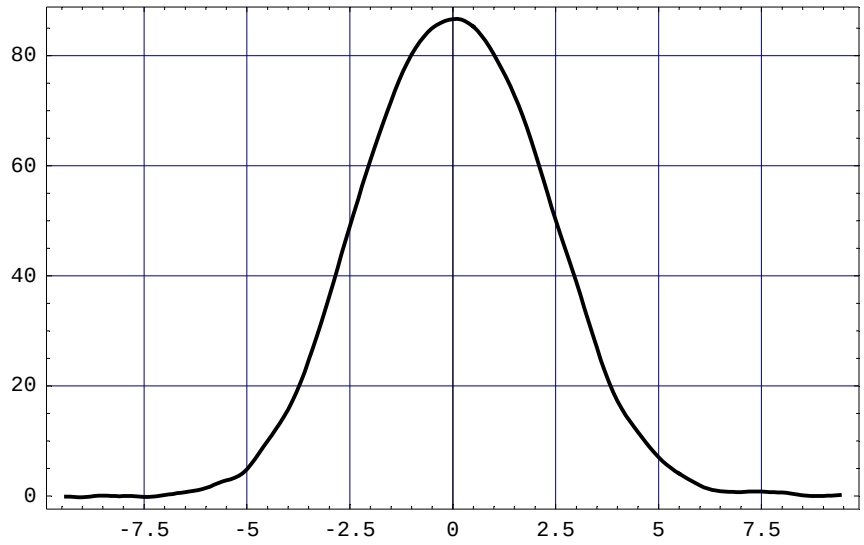
In[82]:=
```
dat1 = sel[pa, -9.41, 9.41]; dat2 = sel[pb, -9.41, 9.41]; dat3 = sel[pc, -9.41, 9.41];
```

In[83]:=
```
{Length[dat1], Apply[Plus, dat1[[All, 2]]],
 Apply[Plus, dat2[[All, 2]]], Apply[Plus, dat3[[All, 2]]]}
```

Out[83]= {189, 5742.52, 4958.17, 4654.61}

The areas decrease during the store. The selection has cut the data at the rf bucket boundries of +/- 9.4 ns. The horizintal scale is ns.

In[84]:=
```
pl7 = ListPlot[dat2]
```



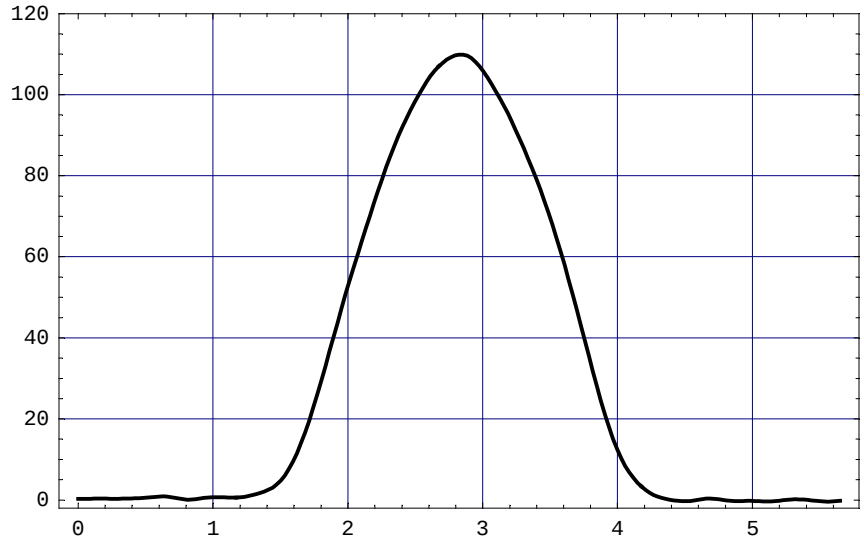
Out[84]= - Graphics -

In[85]:=
```
dat11 = Table[{(Pi lambar + dat1[[-n, 1]] * Pi * lambar / 9.4), dat1[[n, 2]]}, {n, 1, 189}];
```

In[86]:=
```
dat12 = Table[{(Pi lambar + dat2[[-n, 1]] * Pi * lambar / 9.4), dat2[[n, 2]]}, {n, 1, 189}];
```

In[87]:=
```
dat13 = Table[{(Pi lambar + dat3[[-n, 1]] * Pi * lambar / 9.4), dat3[[n, 2]]}, {n, 1, 189}];
```

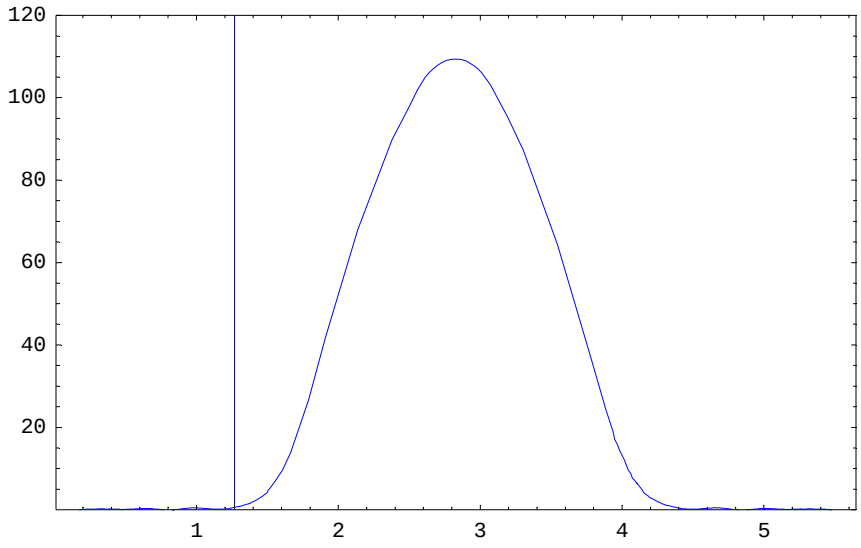
In[88]:=
```
pl8 = ListPlot[dat11, PlotRange → {All, {-2, 120}}]
```



Out[88]= - Graphics -

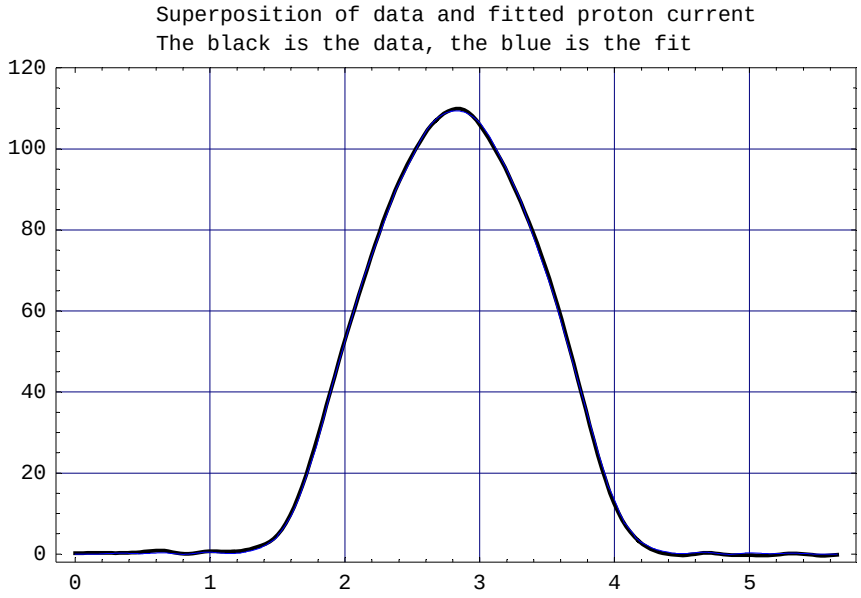
The pulses have been converted to using the z axis in meters so that they can be expanded in func2 functions.

```
In[89]:= Length[dat2]

Out[89]= 189

In[90]:= Fit[dat11, Table[func2[[m, 1]], {m, 1, 50}], z];

In[91]:= Fit[dat12, Table[func2[[m, 1]], {m, 1, 50}], z];
         Fit[dat13, Table[func2[[m, 1]], {m, 1, 50}], z];

In[93]:= pl9 = Plot[%90, {z, 0, 2 Pi lambar}, PlotStyle → RGBColor[0, 0, 1],
           PlotRange → {{0.01, 2 Pi lambar}, {0, 120}}, GridLines → {{1.26825}, None}]
```




```
Out[93]= - Graphics -
```

```
In[95]:= pl5a = Show[{pl8, pl9}, PlotLabel -> "Superposition of data and
            fitted proton current\nThe black is the data, the blue is the fit"]
```




```
Out[95]= - Graphics -
```

```
In[96]:= coef980a = Drop[Regress[dat11, Table[func2[[m, 1]], {m, 1, 50}],
             z, RegressionReport → {BestFitParameters}][[1, 2]], 1];
          coef980b = Drop[Regress[dat12, Table[func2[[m, 1]], {m, 1, 50}],
             z, RegressionReport → {BestFitParameters}][[1, 2]], 1];
```

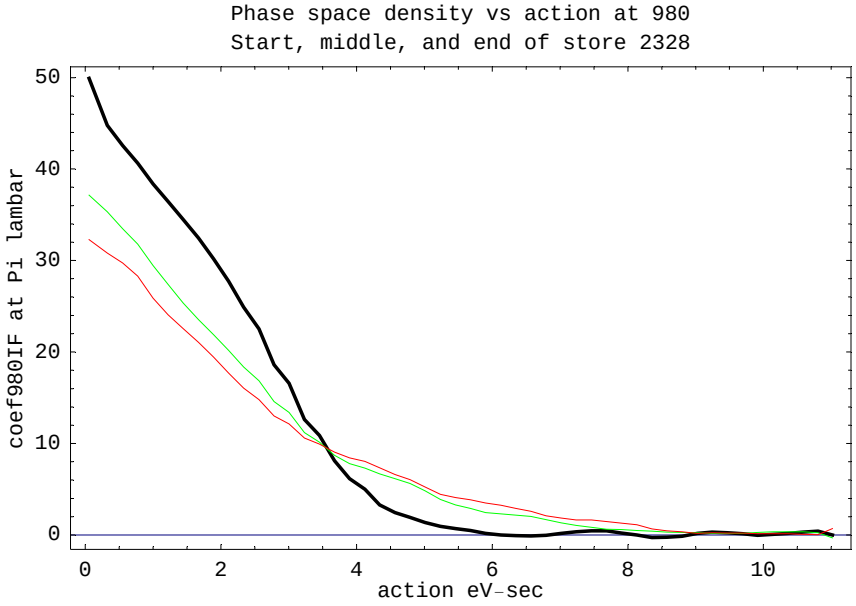
```
In[97]:= coef980c = Drop[Regress[dat13, Table[func2[[m, 1]], {m, 1, 50}],
              z, RegressionReport → {BestFitParameters}][[1, 2]], 1];
```

```
In[98]:= coef980aIF[x_] = Interpolation[Table[{func2[[m, 2]], coef980a[[m]]}, {m, 1, 50}]][x];
          coef980bIF[x_] = Interpolation[Table[{func2[[m, 2]], coef980b[[m]]}, {m, 1, 50}]][x];
          coef980cIF[x_] = Interpolation[Table[{func2[[m, 2]], coef980c[[m]]}, {m, 1, 50}]][x];
```

The above steps make a continuous function relating the density along the central axis and the action. this is a key step in the process. The step below relates the density at any point to the density along the central axis. The coef980IF[x] give the phase density along the central axis vs the oscillation energy.

```
In[99]:= densitya[zi_, Ei_] := coef980aIF[delE2[Pi lambar, zi, Ei, 980 10^9]] *
            UnitStep[Emax[980 10^9] - delE2[Pi lambar, zi, Ei, 980 10^9]];
          densityb[zi_, Ei_] := coef980bIF[delE2[Pi lambar, zi, Ei, 980 10^9]] *
            UnitStep[Emax[980 10^9] - delE2[Pi lambar, zi, Ei, 980 10^9]];
          densityc[zi_, Ei_] := coef980cIF[delE2[Pi lambar, zi, Ei, 980 10^9]] *
            UnitStep[Emax[980 10^9] - delE2[Pi lambar, zi, Ei, 980 10^9]]
```

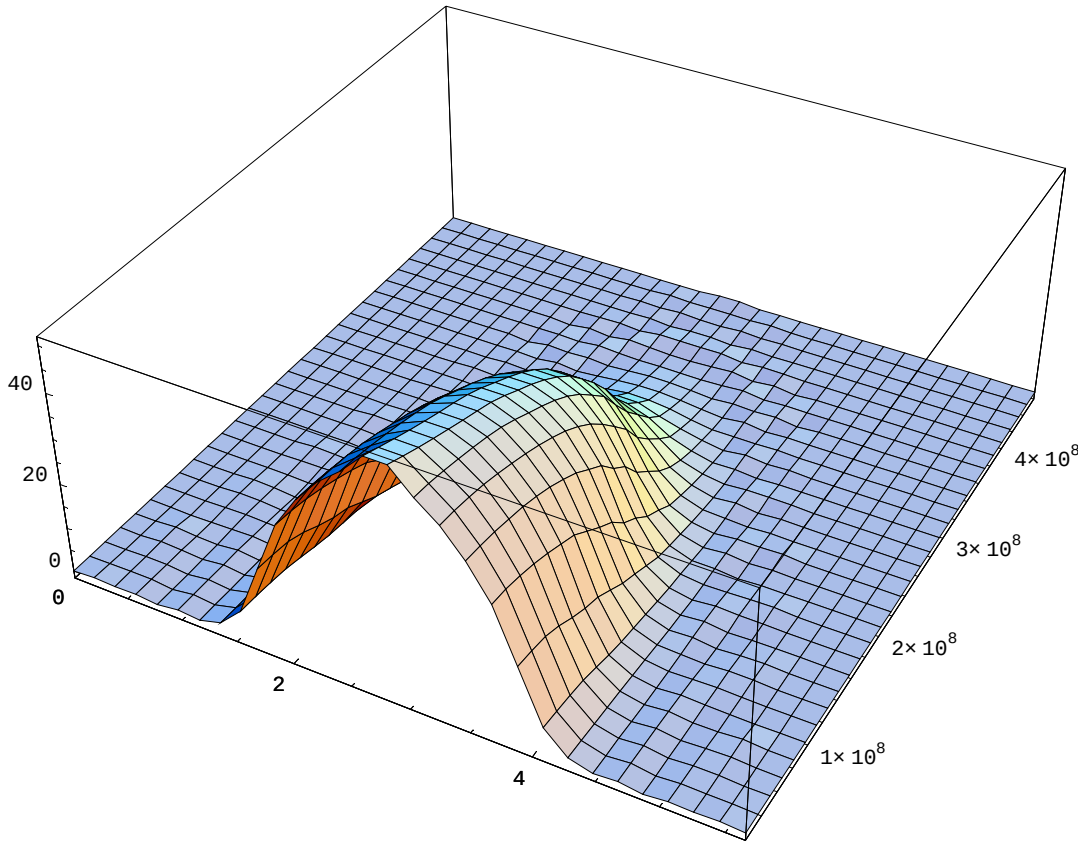
```
In[100]:= pl10 = ListPlot[
            Table[{Ar980[func2[[m, 2]]], coef980a[[m]]}, {m, 1, 50}], GridLines → {None, {0}},
            FrameLabel → {"action eV-sec", "coef980IF at Pi lambar", "Phase space
                density vs action at 980\nStart, middle, and end of store 2328", None}]

Out[101]= - Graphics -

In[102]:= pl10b = ListPlot[Table[{Ar980[func2[[m, 2]]], coef980c[[m]]}, {m, 1, 50}],
            PlotStyle → RGBColor[1, 0, 0]]

In[103]:= pl10c = Show[pl10, pl10a, pl10b]
```




```
Out[103]= - Graphics -
```

The plot shows the phase density vs action for the three different times in the store. The plot below shows a density plot vs z and E.

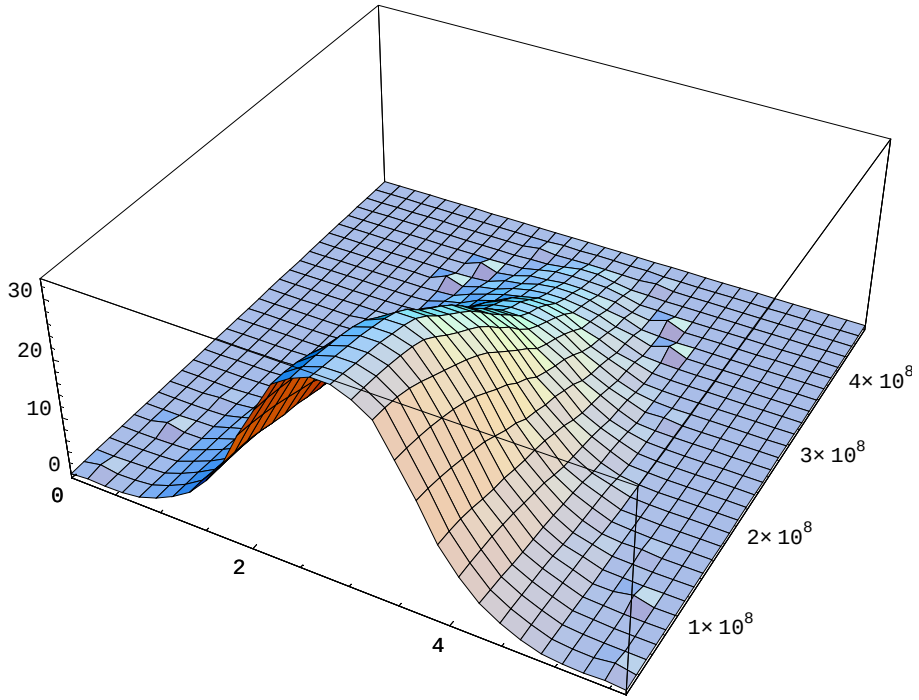
```
In[104]:= pl11 = Plot3D[densitya[zi, Ei], {zi, 0, 2 Pi lambar},
             {Ei, 2 10^7, 46 10^7}, PlotPoints → 30, PlotRange → All]
```




```
Out[104]= - SurfaceGraphics -
```

Density plot at the start of the store.

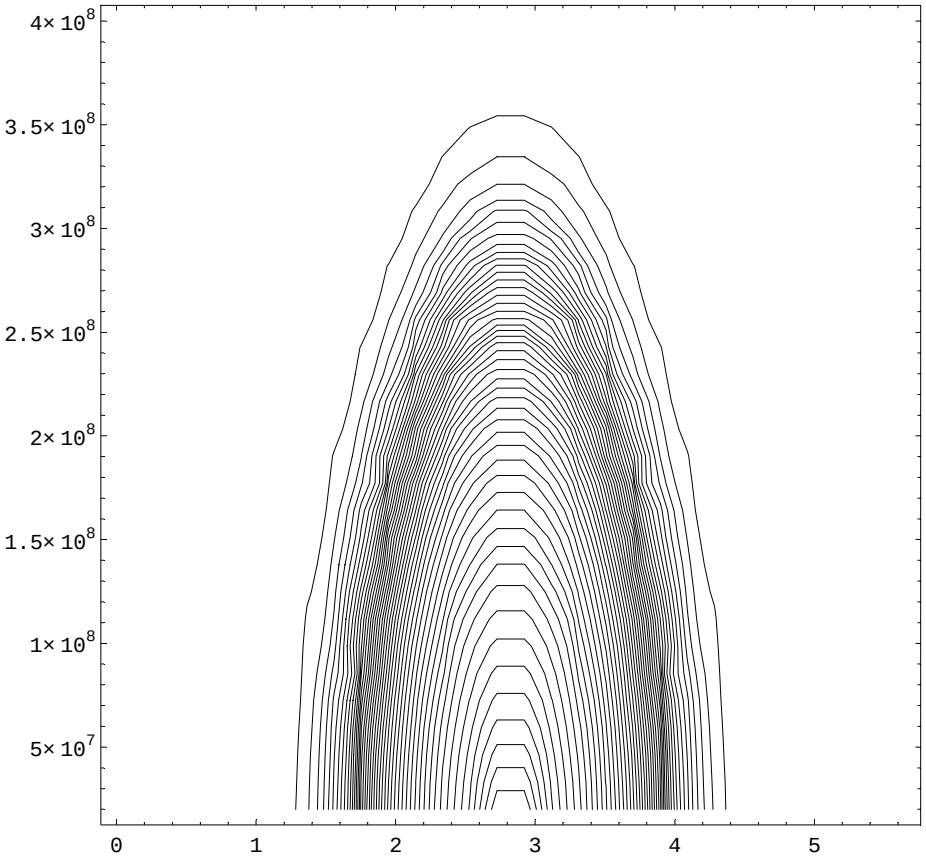
```
In[105]:= pl12 = Plot3D[densityc[zi, Ei], {zi, 0, 2 Pi lambar},
             {Ei, 2 10^7, 46 10^7}, PlotPoints → 30, PlotRange → All]
```




```
Out[105]= - SurfaceGraphics -
```

Density plot at the end of the store.

```
pl13 = ContourPlot[densitya[zi, Ei], {zi, 0, 2 Pi lambar}, {Ei, 2 10^7, 4. 10^8},
  PlotRange → All, PlotPoints → 30, Contours → 50, ContourShading → False]
```




```
- ContourGraphics -
```

The momentum distribution

We can now integrate the density distribution over z and obtain the momentum distribution. This routine needs improvement as *Mathematica* complains loudly about the discontinuities in slope that it finds. The accuracy is plenty good enough, but it is an ugly approach! We make a table of the integral vs momentum.

```
In[106]:= pdista980 = Table[
            {Ei, NIntegrate[densitya[zi, Ei], {zi, 0, 2 Pi lambar}]}, {Ei, 2 10^7, 4.5 10^8, 10^7}]
```

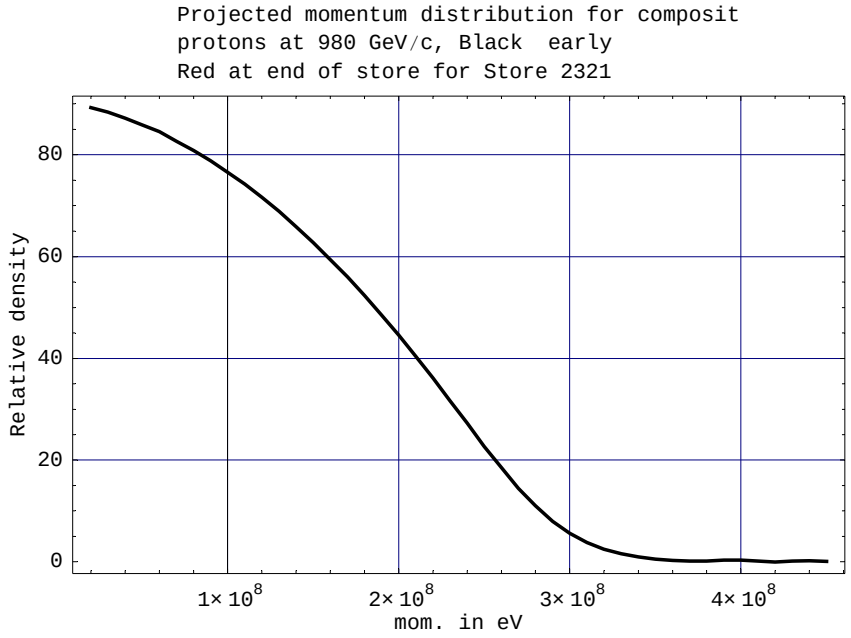
```
General::stop : Further output of NIntegrate::ncvb will be suppressed during this calculation.
```

```
Out[106]= {{20000000, 89.2845}, {30000000, 88.3545}, {40000000, 87.207}, {50000000, 85.8696},
           {60000000, 84.5602}, {70000000, 82.6766}, {80000000, 80.8244}, {90000000, 78.8005},
           {100000000, 76.5907}, {110000000, 74.2231}, {120000000, 71.6608},
           {130000000, 68.8821}, {140000000, 65.8757}, {150000000, 62.6914},
           {160000000, 59.3915}, {170000000, 55.9542}, {180000000, 52.3272},
           {190000000, 48.53}, {200000000, 44.5559}, {210000000, 40.4016},
           {220000000, 36.0972}, {230000000, 31.6591}, {240000000, 27.2847},
           {250000000, 22.6761}, {260000000, 18.5209}, {270000000, 14.4276},
           {280000000, 11.0625}, {290000000, 7.98185}, {300000000, 5.63022},
           {310000000, 3.8157}, {320000000, 2.45337}, {330000000, 1.61961},
           {340000000, 0.983175}, {350000000, 0.581958}, {360000000, 0.282393},
           {370000000, 0.175553}, {380000000, 0.188175}, {390000000, 0.354062},
           {400000000, 0.37321}, {410000000, 0.131521}, {420000000, -0.0287008},
           {430000000, 0.19054}, {440000000, 0.199}, {450000000, 0.11737}}
```

```
In[107]:= pdistc980 = Table[
            {Ei, NIntegrate[densityc[zi, Ei], {zi, 0, 2 Pi lambar}]}, {Ei, 2 10^7, 4.5 10^8, 10^7}]
```

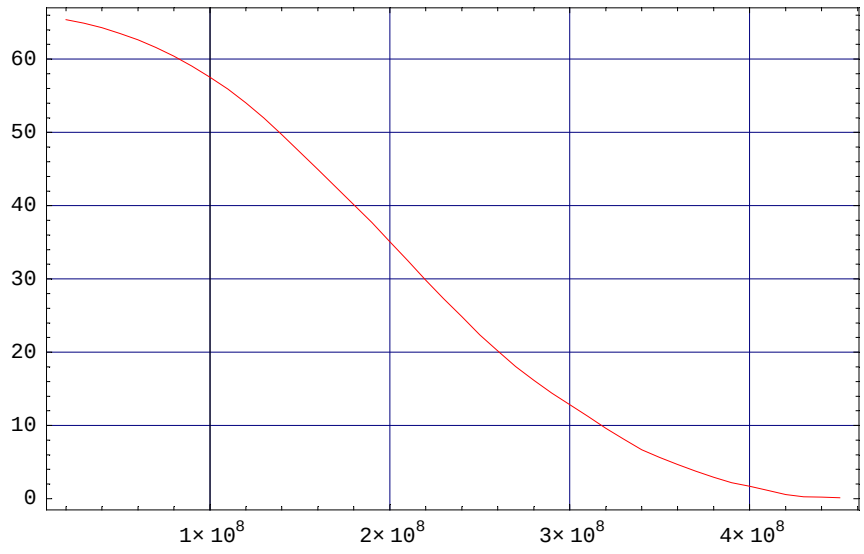
```
Out[107]= {{20000000, 65.3986}, {30000000, 64.8975}, {40000000, 64.2658}, {50000000, 63.5118},
           {60000000, 62.6193}, {70000000, 61.5873}, {80000000, 60.4066}, {90000000, 59.0527},
           {100000000, 57.5285}, {110000000, 55.9006}, {120000000, 54.0483},
           {130000000, 52.0061}, {140000000, 49.712}, {150000000, 47.3344},
           {160000000, 44.933}, {170000000, 42.5647}, {180000000, 40.1585},
           {190000000, 37.6739}, {200000000, 35.1044}, {210000000, 32.5052},
           {220000000, 29.8621}, {230000000, 27.2925}, {240000000, 24.8325},
           {250000000, 22.3764}, {260000000, 20.1756}, {270000000, 18.0216},
           {280000000, 16.1723}, {290000000, 14.4248}, {300000000, 12.8237},
           {310000000, 11.2607}, {320000000, 9.62092}, {330000000, 8.13902},
           {340000000, 6.67945}, {350000000, 5.62314}, {360000000, 4.68565},
           {370000000, 3.78125}, {380000000, 2.91176}, {390000000, 2.18746},
           {400000000, 1.68894}, {410000000, 1.14054}, {420000000, 0.555305},
           {430000000, 0.278002}, {440000000, 0.20669}, {450000000, 0.116942}}
```

In[114]:=
```
pl14 = ListPlot[pdista980, FrameLabel → {"mom. in eV", "Relative density",
    "Projected momentum distribution for composit\nprotons at 980
      GeV/c, Black  early\nRed at end of store for Store 2321", None}]
```



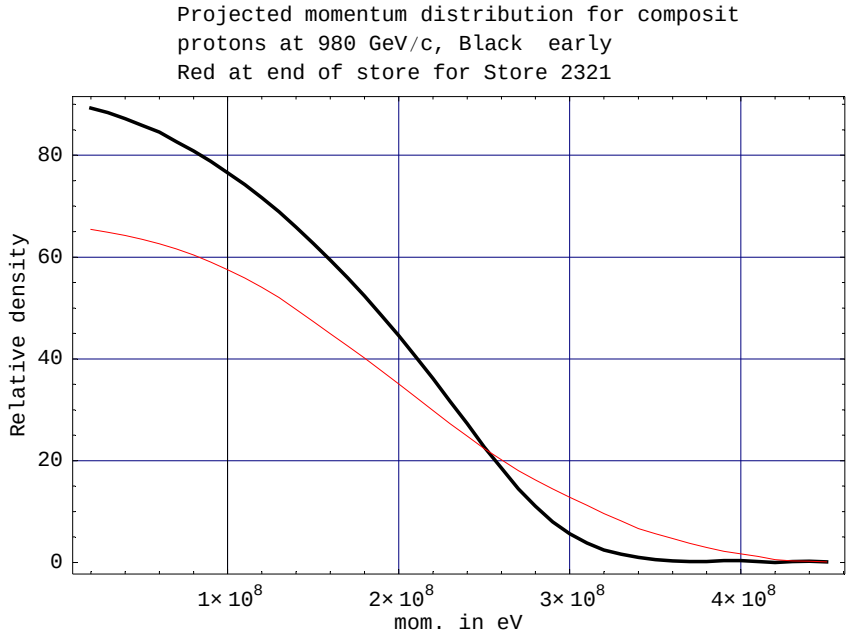
Out[114]= - Graphics -

In[115]:=
```
pl15 = ListPlot[pdistc980, PlotStyle → RGBColor[1, 0, 0]]
```



Out[115]= - Graphics -

In[116]:= **pl16 = Show[pl14, pl15]**



Out[116]= **-** Graphics **-**