Who needs TWISS,DELTAP?

John Jowett

```
Date[]
```

```
{2001, 5, 11, 16, 38, 15}
```

Following a discussion with Eberhard Keil and Hans Grote yesterday, show how the existing TFS table outputs make it unnecessary to have the TWISS command looping over momentum deviations. The same functionality, plus a great deal more, is provided by Madtomma. The following is just one way to do it.

Setup

In[21]:=
```
{$MachineName, $OperatingSystem}
```

Out[21]=
```
{PCSLDS16, WindowsNT}
```

Packages needed

```
Needs["Madtomma`RunningMAD`"]
```

In[22]:=
```
Needs["Madtomma`Mfs`Mfs`"]
```

In[23]:=
```
SetDirectory[ThisNotebookDirectory[]]
```

Out[23]=
```
D:\My Documents\CLIC2001\DampingRings\NLC Damping rings
```

There is a MAD pool file containing the NLC Damping ring optics.

In[24]:=
```
edr = First[FileNames["*.pool"]]
```

Out[24]=
```
edr.pool
```

Functions for interacting with MAD

We already saved a pool file for an optics. Here's a function that specifies the calculations we want as a function of the momentum deviation δ. It uses the MadInput package.

In[25]:=
```
madin[δ_] := {
  MADpoolLoad["edr.pool"],
  "SELECT,OPTICS,class=QUADRUPOLE",
  MADoptics["optdp.tfs",
   columns → {"NAME", "S", "BETX", "BETY", "DX"}, DELTAP → δ]
 }
```

This returns a list of MadInput objects

In[26]:=
```
madin[.01]
```

Out[26]=
```
{MADpoolLoad[edr.pool], SELECT,OPTICS,class=QUADRUPOLE,
 MADoptics[optdp.tfs, columns → {NAME, S, BETX, BETY, DX}, DELTAP → 0.01]}
```

When they are finally fed to MAD they will look like this (but we don't need to know).

In[27]:=
```
madin[.01] // MADcommand // TableForm
```

Out[27]//TableForm=
```
poolload,"edr.pool"
SELECT,OPTICS,class=QUADRUPOLE

OPTICS,CENTRE,filename="optdp.tfs",columns=NAME, S, BETX, BETY, DX,DELTA
```

The new RunningMAD package provides a platform independent way to run MAD:

In[28]:=
```
? runMAD
```

```
runMAD[input] runs the MAD program and returns the console output from
  the MAD run as a list of strings.  The input object can be:
    (1) a string giving the name of a MAD input file
    (2) a list of MadInput objects
  recognised by the Madtomma`MadLanguage`MadInput` package
    (3) a list of strings containing MAD
  commands (a special case of (2)).
runMAD has a number of options which allow it to be customised
  for different computers or versions of the program.
```

In[29]:=
```
Options[runMAD]
```

Out[29]=
```
{MADprogram → C:\Progra~1\mad8dl\MAD8DL.BAT,
 runMADcleanup → False, runMADTemporaryFile → runMADLastFile.mad}
```

A quick test run:

In[30]:=
```
runMAD[madin[0.001]] // ColumnForm
```

Out[30]=
```
1"MAD" Version 8.23dl  Windows NT 4.0            Copyright (C) 1990 by
                                                 06/06/01     09.55.38

 Input stream and message log:


 MAD.     Reading standard input file.

              ! Temporary file created by runMAD at: {2001, 6, 6, 9, 55
              ASSIGN,print=TERM ! collect print and echo output togethe

 FLASSI.  PRINT stream rerouted to standard output.


              poolload,"edr.pool"

 FLLOAD.  Memory dump read on stream: edr.pool
          File name: edr.pool

         5    SELECT,OPTICS,class=QUADRUPOLE

              OPTICS,CENTRE,filename="optdp.tfs",columns=NAME, S, BETX,

 TMCLOR.  Searching for closed orbit, beam line: "EDR             ", ran
          Delta(p)/p =    0.001000, symmetry = F
 Iteration     x           px          y           py          t
         0    0.000000    0.000000    0.000000    0.000000    0.000000
         1   -0.000090    0.000000    0.000000    0.000000    0.000084
         2   -0.000090    0.000000    0.000000    0.000000    0.000098

 TWOPGO.  ## Warning ## OPTICS found transverse coupling for delta(p)/p
          results may be wrong.


 TWOPGO.  ## Warning ## OPTICS uses the RF system and synchrotron radiat
          for optical calculations it ignores both.


 TWOPTC.  Lattice functions written on file: optdp.tfs

              stop ! in case we forgot
```

```
ZEND.         2 Warning messages,
              0 Error messages.
                MAD terminated on 06/06/01 at 09.55.39

MZEND.   Usage statistics for  1 dynamic stores.
  Map of store  0 /MEMORY/
  ------------------------
  Division                                       Number of times
             Kind                 Max-size              Garb-coll.
           Mode     Position       used    allowed   Wiped  user   auto Pus
   1 QDIV1    0 1          9          0    4000006       0      0      0
   2 QDIV2    1 1    3999606     154515    4000006       1      0      0
  20 system   1 8    4000006        293    4000006       0      0      0
    del %fname%.ps
```

That's as close as we need to get to looking at an output.

A function to return an mfs object with the optics information as a function of deltap. Use dynamic programming here to avoid repeating MAD runs.

In[31]:=
```
edrOptics[δ_] := edrOptics[δ] =
  tfsRead[(runMAD[madin[δ], runMADcleanup → True]; "optdp.tfs"),
   mfsVerbose → False]
```

An example

In[32]:=
```
Short[
 edrOptics[-.00172],
 1]
```

Out[32]//Short=
```
mfs[{{GAMTR, 39.139}, «11», {TIME, 09.55.48}},
 {NAME, S, BETX, BETY, DX}, {«1»}]
```

Applications

Tune dependence on momentum

Some auxiliary functions to get the global optical parameters we are interested in

In[33]:=
```
edrQx[δ_] := mfsKeyValue[edrOptics[δ], "QX"];
edrQy[δ_] := mfsKeyValue[edrOptics[δ], "QY"];
edrxix[δ_] := mfsKeyValue[edrOptics[δ], "XIX"];
edrxiy[δ_] := mfsKeyValue[edrOptics[δ], "XIY"];
```

In[37]:=
```
edrQx[0.]
```

Out[37]=
```
23.8233
```

Just ask for a table of values. A number of MAD runs will take place behind the scenes the first time we evaluate this

In[38]:=
```
TableForm[
 Table[{δ, edrQx[δ], edrQy[δ]}, {δ, -.02, .02, .002}],
 TableHeadings → {{}, {"δp", "Qx", "Qy"}}
]
```

Out[38]//TableForm=

δp	Q_x	Q_y
−0.02	23.831	10.2803
−0.018	23.8296	10.2804
−0.016	23.8284	10.2808
−0.014	23.8273	10.2814
−0.012	23.8263	10.282
−0.01	23.8255	10.2826
−0.008	23.8248	10.283
−0.006	23.8242	10.2832
−0.004	23.8238	10.2831
−0.002	23.8235	10.2829
0.	23.8233	10.2826
0.002	23.8233	10.2821
0.004	23.8233	10.2816
0.006	23.8234	10.2811
0.008	23.8236	10.2808
0.01	23.8238	10.2806
0.012	23.8241	10.2806
0.014	23.8243	10.2808
0.016	23.8246	10.2811
0.018	23.8248	10.2815
0.02	23.8251	10.2818

Look at the deltap values for which we've made a MAD run.

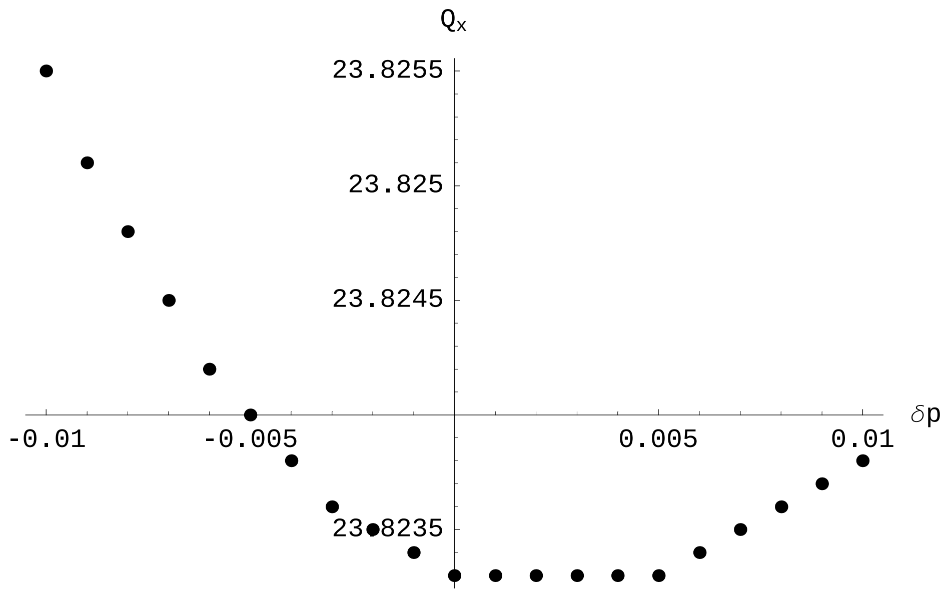
In[39]:=
```
domain[edrOptics]
```

Out[39]=
```
{-0.02, -0.018, -0.016, -0.014, -0.012, -0.01,
 -0.008, -0.006, -0.004, -0.002, -0.00172, 0., 0.002, 0.004,
 0.006, 0.008, 0.01, 0.012, 0.014, 0.016, 0.018, 0.02, δ_, δ}
```

In[40]:=
```
SetOptions[ListPlot, Prolog → PointSize[.015]];
```

If we now ask for more points, MAD runs willl only take place for the new values.

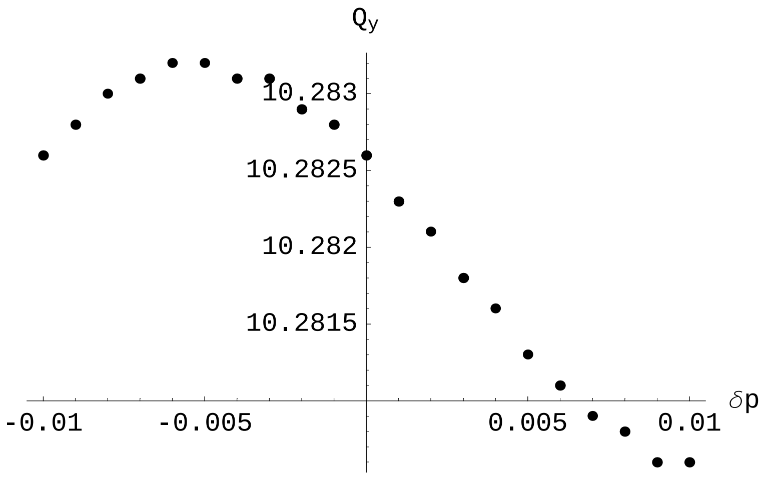
In[41]:=
```
edrQxplot = ListPlot[
  Table[{δ, edrQx[δ]}, {δ, -.01, .01, .001}], AxesLabel → {"δp", "Qx"}]
```



Out[41]=
```
- Graphics -
```

Now that all the optics are in *Mathematica*'s memory, other results are returned very quickly.

In[43]:=
```
edrQyplot = ListPlot[
  Table[{δ, edrQy[δ]}, {δ, -.01, .01, .001}], AxesLabel → {"δp", "Qy"}]
```



Out[43]=
```
- Graphics -
```

How many times did we run MAD there ? Or, for which values of δ have we already saved the optics?

In[44]:=
```
domain[edrOptics]
```

Out[44]=
```
{-0.02, -0.018, -0.016, -0.014, -0.012, -0.01, -0.009, -0.008,
 -0.007, -0.006, -0.005, -0.004, -0.003, -0.002, -0.00172,
 -0.001, 0., 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007,
 0.008, 0.009, 0.01, 0.012, 0.014, 0.016, 0.018, 0.02, δ_, δ}
```

Refrain from using `Plot` since it can take a lot of MAD runs to make a really smooth curve!

Find the inflection points in Q_x:

In[45]:=
```
FindMinimum[edrQx[δ], {δ, {0.002, 0.003}}]
```

Out[45]=
```
{23.8233, {δ → 0.00448091}}
```

In[46]:=
```
FindMinimum[-edrQy[δ], {δ, {-0.007, -0.002}}]
```

Out[46]=
```
{-10.2832, {δ → -0.00419499}}
```

Fit cubic polynomials to the tune dependences

In[47]:=
```
qx[δ_] = Fit[Table[{δ, edrQx[δ]}, {δ, -.01, .01, .001}], {1, δ, δ^2, δ^3}, δ]
```

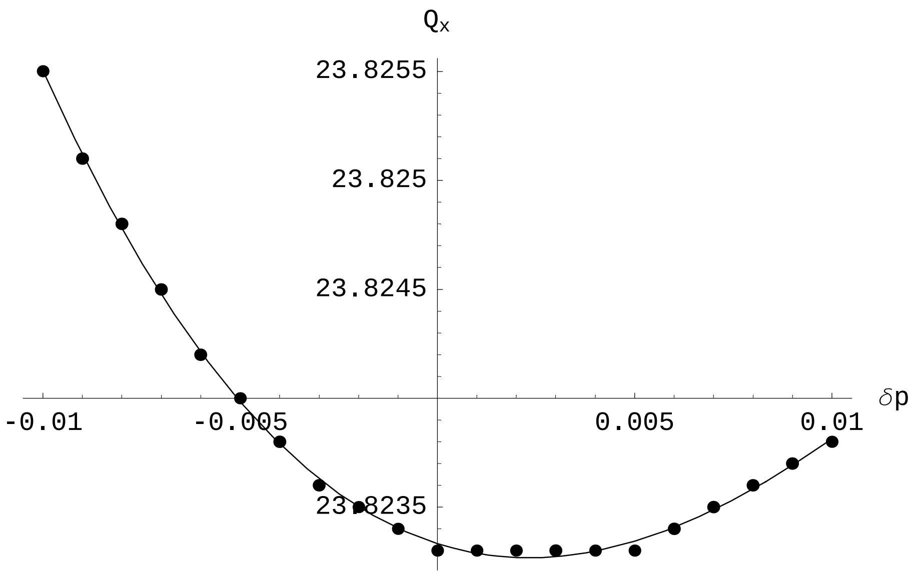
Out[47]= $23.8233 - 0.0572098\,\delta + 13.2589\,\delta^2 - 273.328\,\delta^3$

In[48]:=
```
qy[δ_] = Fit[Table[{δ, edrQy[δ]}, {δ, -.01, .01, .001}], {1, δ, δ^2, δ^3}, δ]
```

Out[48]= $10.2825 - 0.21489\,\delta - 9.89777\,\delta^2 + 1155.91\,\delta^3$

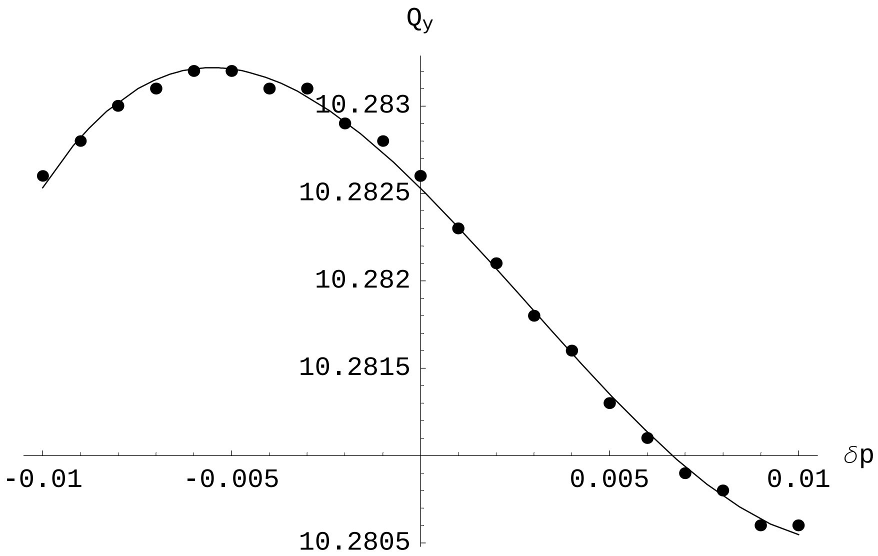
Compare the fits to the data

In[49]:=
```
Show[
 {edrQxplot, Plot[qx[δ], {δ, -0.01, .01}, DisplayFunction → Identity]}]
```



Out[49]= - Graphics -

In[50]:=
```
Show[
 {edrQyplot, Plot[qy[δ], {δ, -0.01, .01}, DisplayFunction → Identity]}]
```



Out[50]=
```
- Graphics -
```

Beta-beating

Define functions to return the lists of Twiss functions around the ring

In[51]:=
```
s[δ_] := mfsColumn[edrOptics[δ], "S"]
```

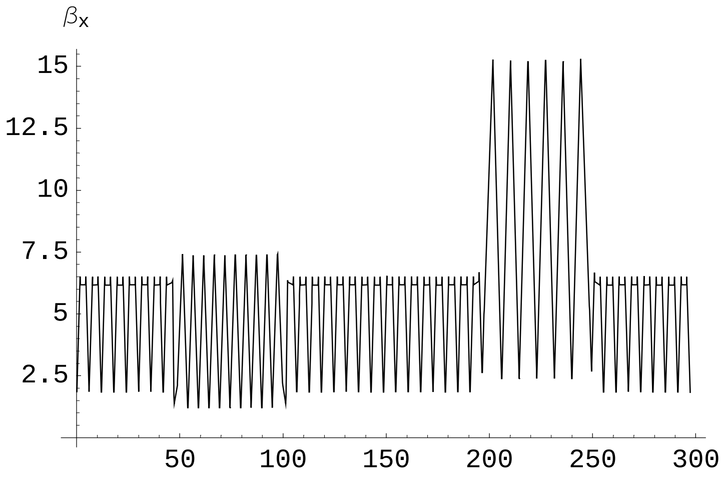
In[52]:=
```
betx[δ_] := mfsColumn[edrOptics[δ], "BETX"]
```

In[53]:=
```
bety[δ_] := mfsColumn[edrOptics[δ], "BETY"]
```

In[54]:=
```
dx[δ_] := mfsColumn[edrOptics[δ], "DX"]
```

In[55]:=
```
SetOptions[ListPlot, PlotJoined → True];
```

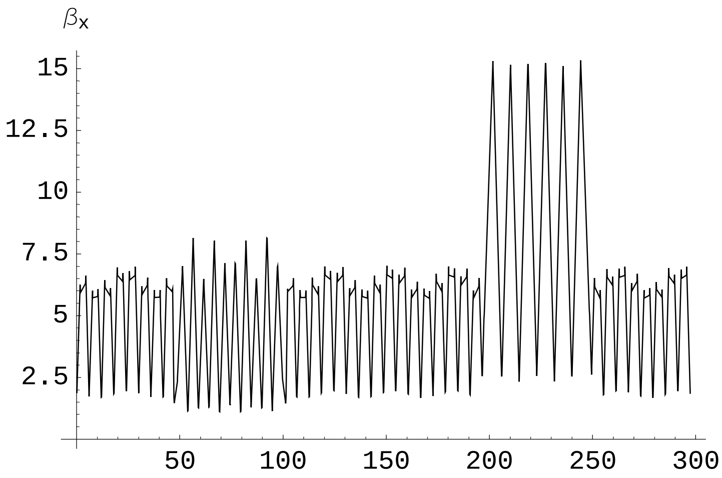
In[56]:=
```
ListPlot[Transpose[{s[0], betx[0]}], AxesLabel → "βx"]
```



Out[56]= - Graphics -

Plot the off-momentum β-function

In[62]:=
```
ListPlot[Transpose[{s[0], betx[0.01]}], AxesLabel → "βx"]
```

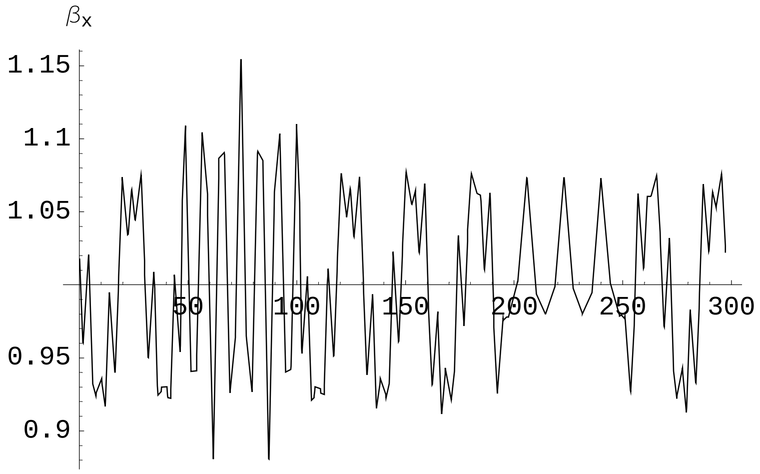


Out[62]= - Graphics -

Plot the beta-beating factor

In[63]:=

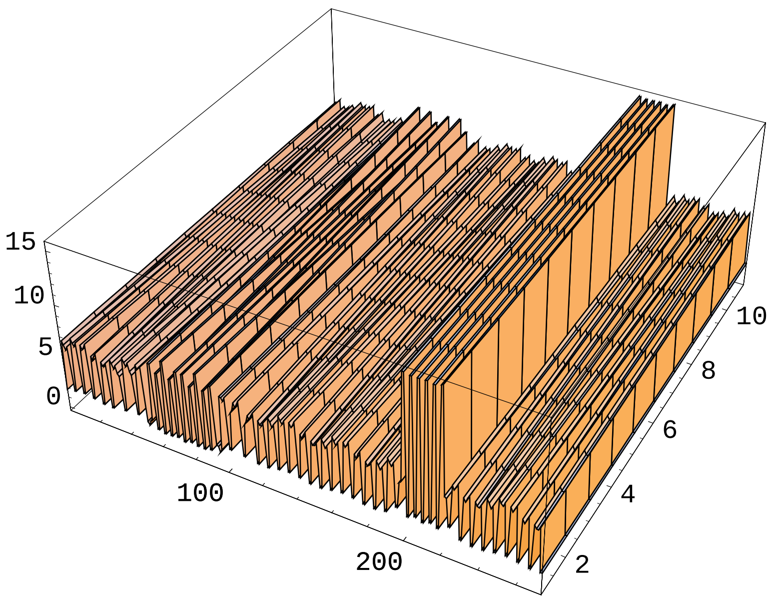
```
ListPlot[Transpose[{s[0], betx[0.01]/betx[0]}], AxesLabel → "βx"]
```



Out[63]= - Graphics -

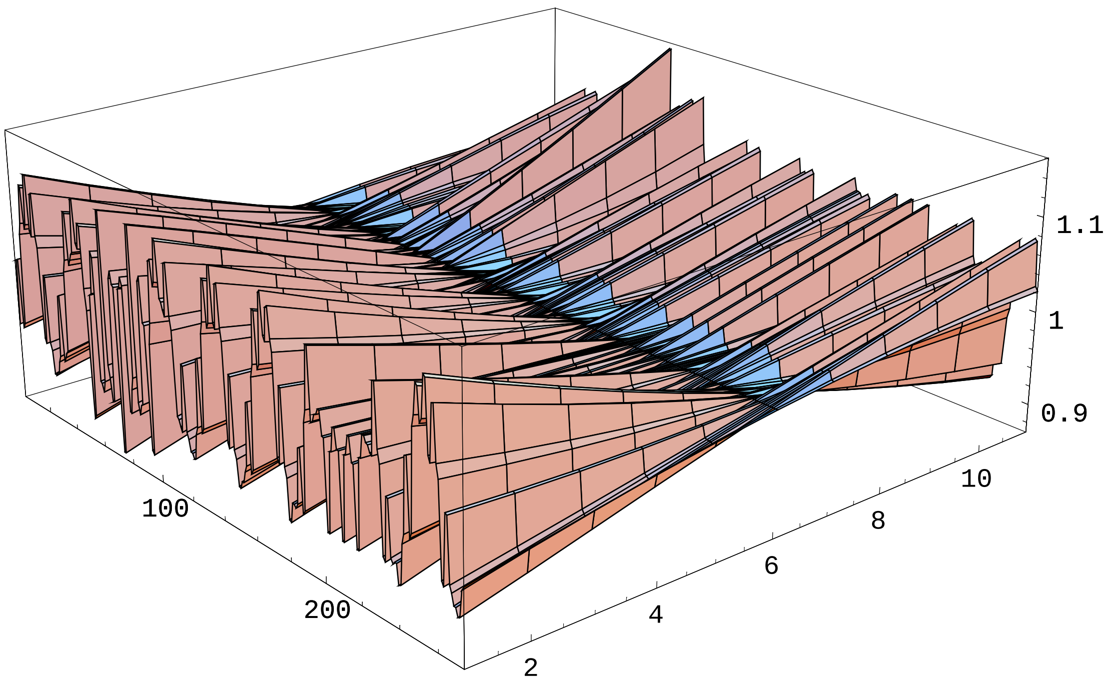
In[64]:=

```
ListPlot3D[Table[betx[δ], {δ, -.01, .01, .002}]]
```



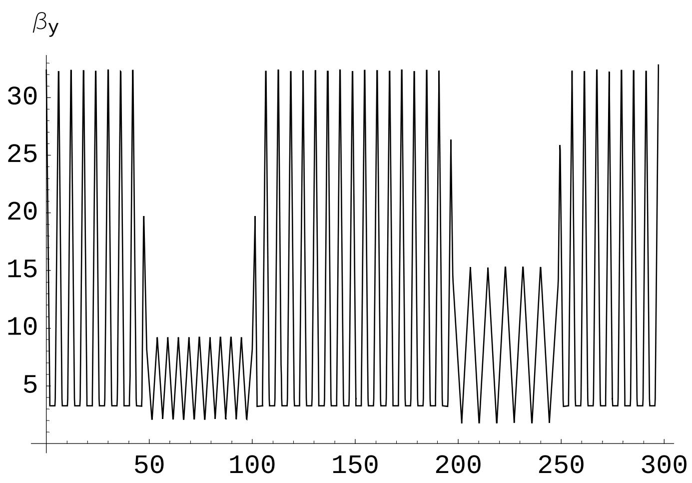
Out[64]= - SurfaceGraphics -

In[65]:=
```
ListPlot3D[Table[betx[δ]/betx[0], {δ, -.01, .01, .002}],
 ViewPoint -> {2.412, -2.029, 1.231}]
```



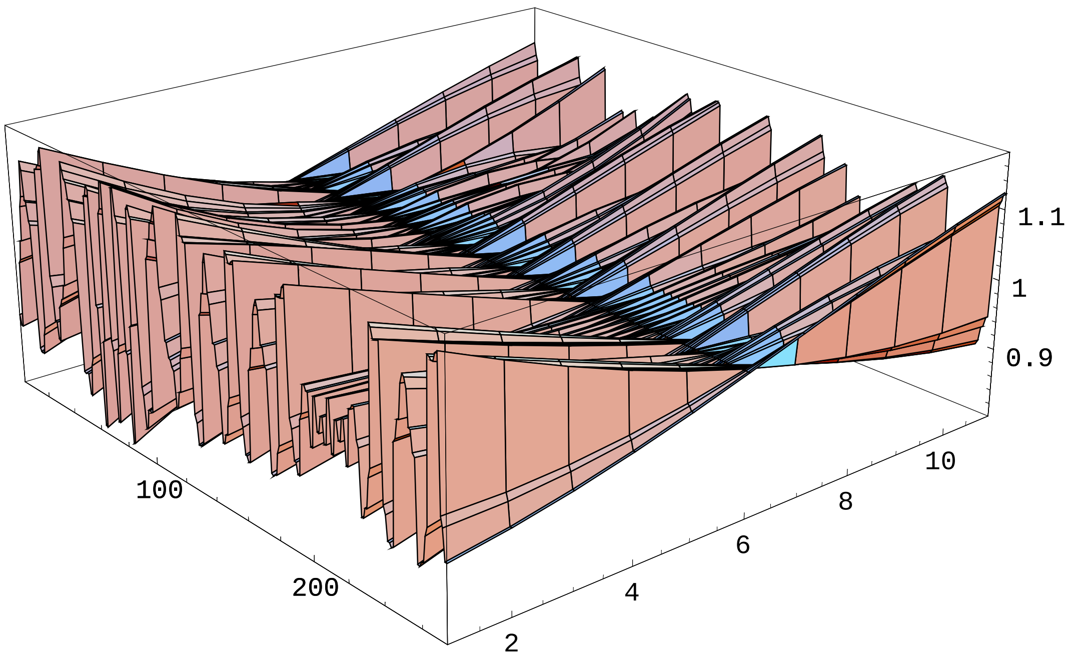
Out[65]=
```
- SurfaceGraphics -
```

In[66]:=
```
ListPlot[Transpose[{s[0], bety[0]}], AxesLabel → "βy"]
```



Out[66]=
```
- Graphics -
```

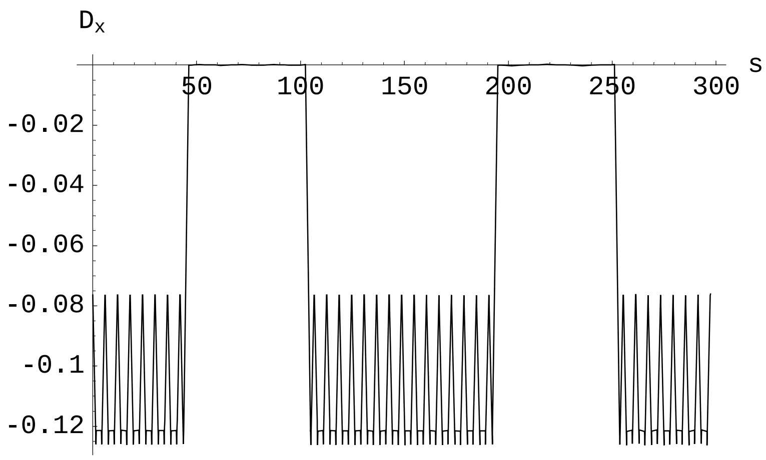
In[67]:=
```
ListPlot3D[Table[bety[δ]/bety[0], {δ, -.01, .01, .002}],
 ViewPoint -> {2.412, -2.029, 1.231}]
```



Out[67]= - SurfaceGraphics -

Dispersion function

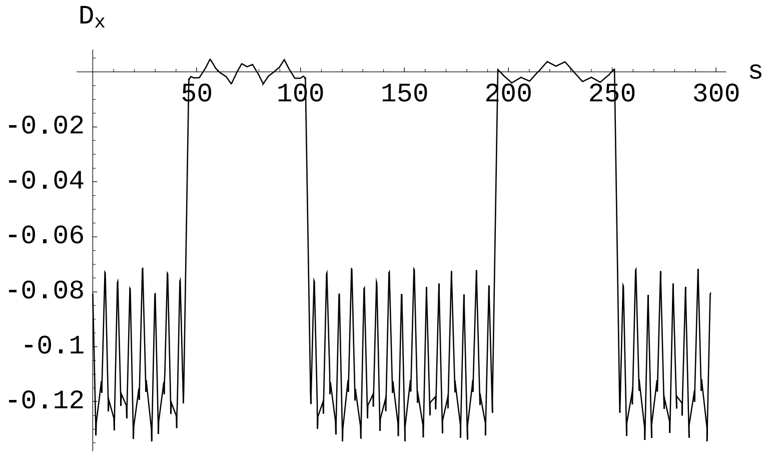
In[68]:=
```
ListPlot[Transpose[{s[0], dx[0]}], AxesLabel → {"s", "Dx"}]
```



Out[68]= - Graphics -

In[69]:=

```
ListPlot[Transpose[{s[0], dx[0.01]}], AxesLabel → {"s", "Dx"}]
```



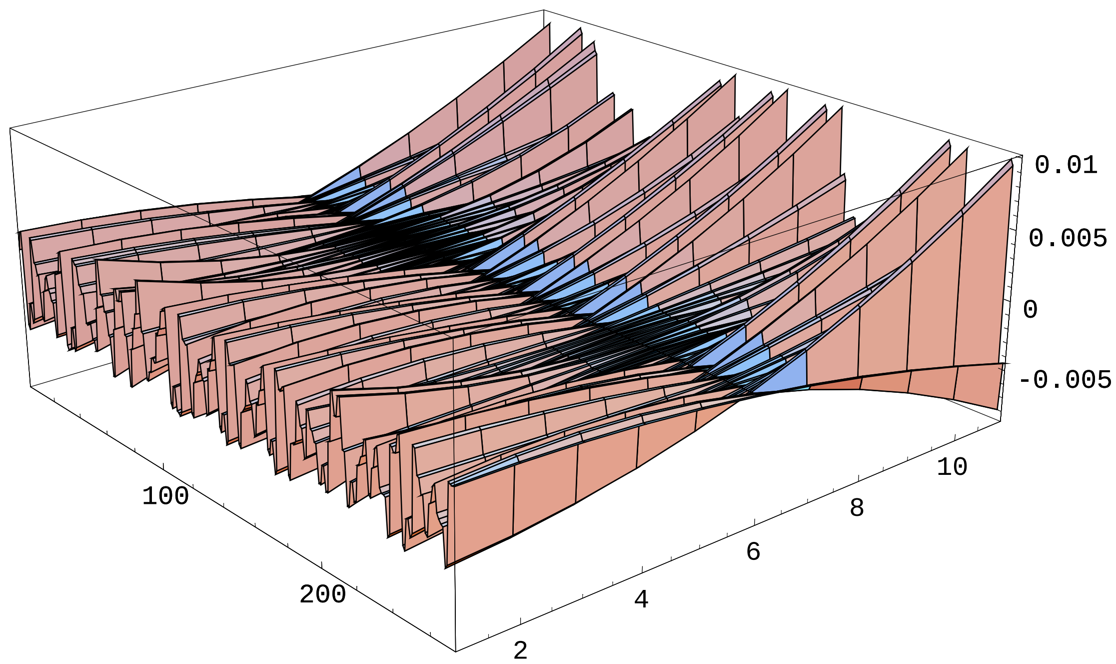
Out[69]=

```
- Graphics -
```

Show the change in dispersion with δ

In[70]:=

```
ListPlot3D[Table[dx[δ] - dx[0], {δ, -.01, .01, .002}],
 ViewPoint -> {2.412, -2.029, 1.231}, PlotRange → All]
```



Out[70]=

```
- SurfaceGraphics -
```