PARMELA modeling and beam-based measurements in the JLab Upgrade FEL injector

Carlos Hernandez-Garcia and Kevin Beard

CASA Beam Physics Seminar

May 11th 2006

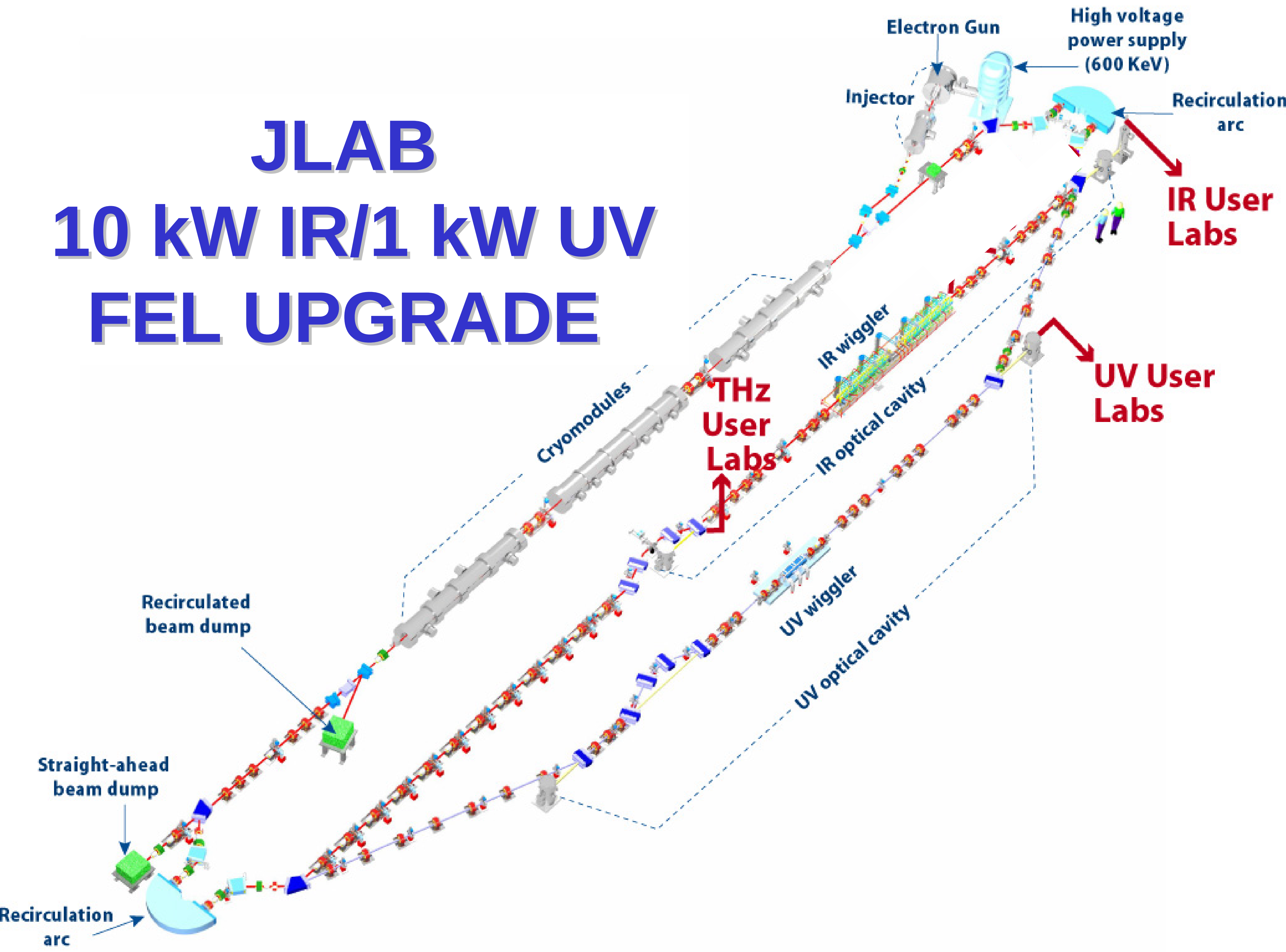
Overview

- This part of the talk is a storyline about trying to understand the injector behavior (i.e. physics?) by bringing closer together the PARMELA model to the actual machine

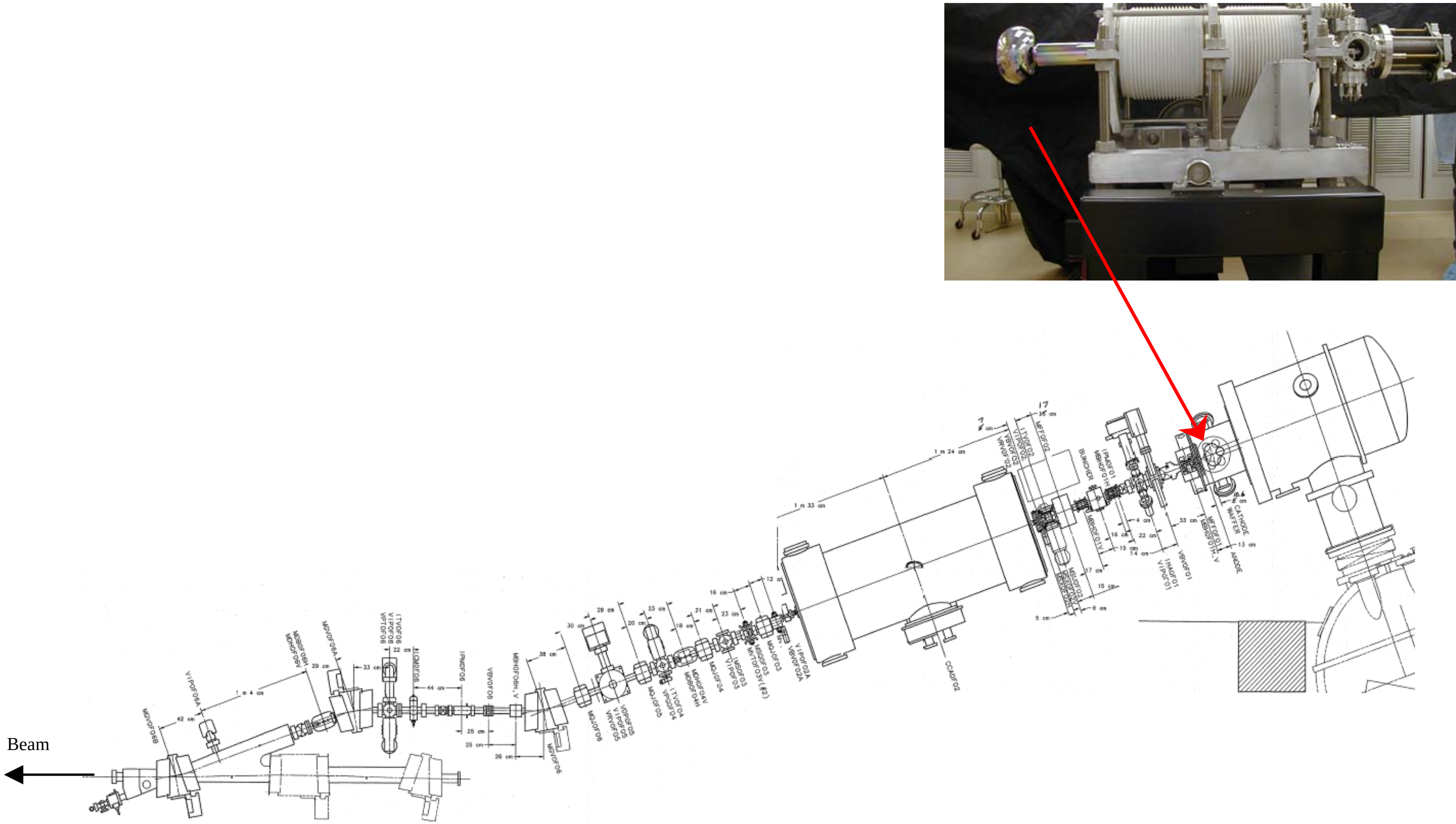
- We have a very limited set of measurements and beam-based observations that show the beam behavior is in very good agreement with the model

- We do not intent to say that the model predicts absolute values for any beam parameters. We look at the derivatives of a specific beam parameter as a function of a specific variable (AKA knob) to show that the model predicts quite well the ‘behavior’ of the beam

JLAB
10 kW IR/1 kW UV
FEL UPGRADE

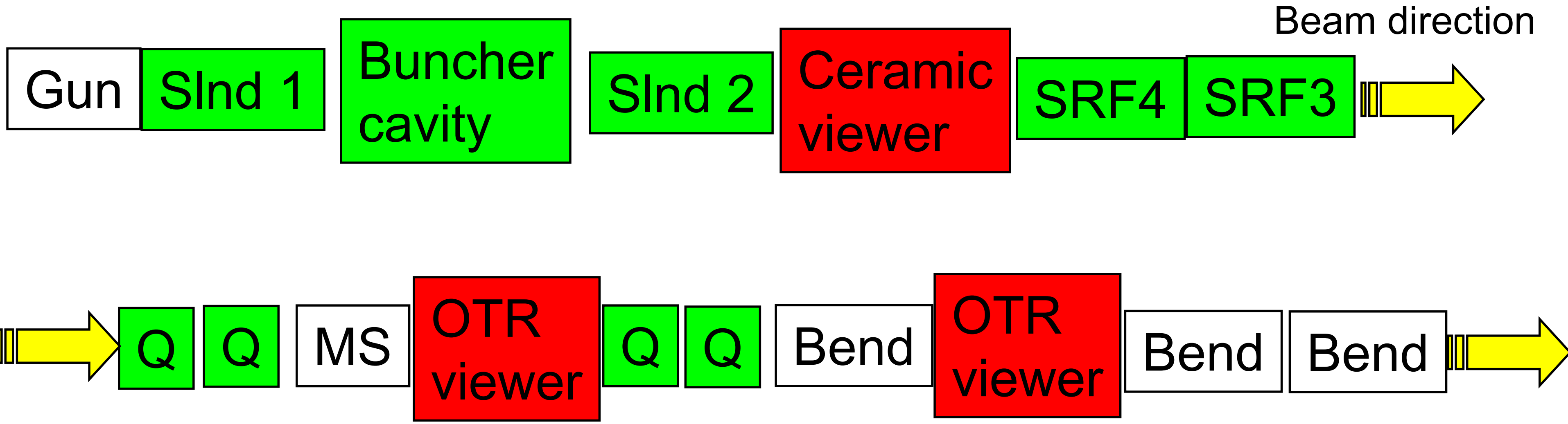


The injector is driven by a 350 kV DC GaAs Photocathode Gun generating 135pC bunches 50 ps FWHM long



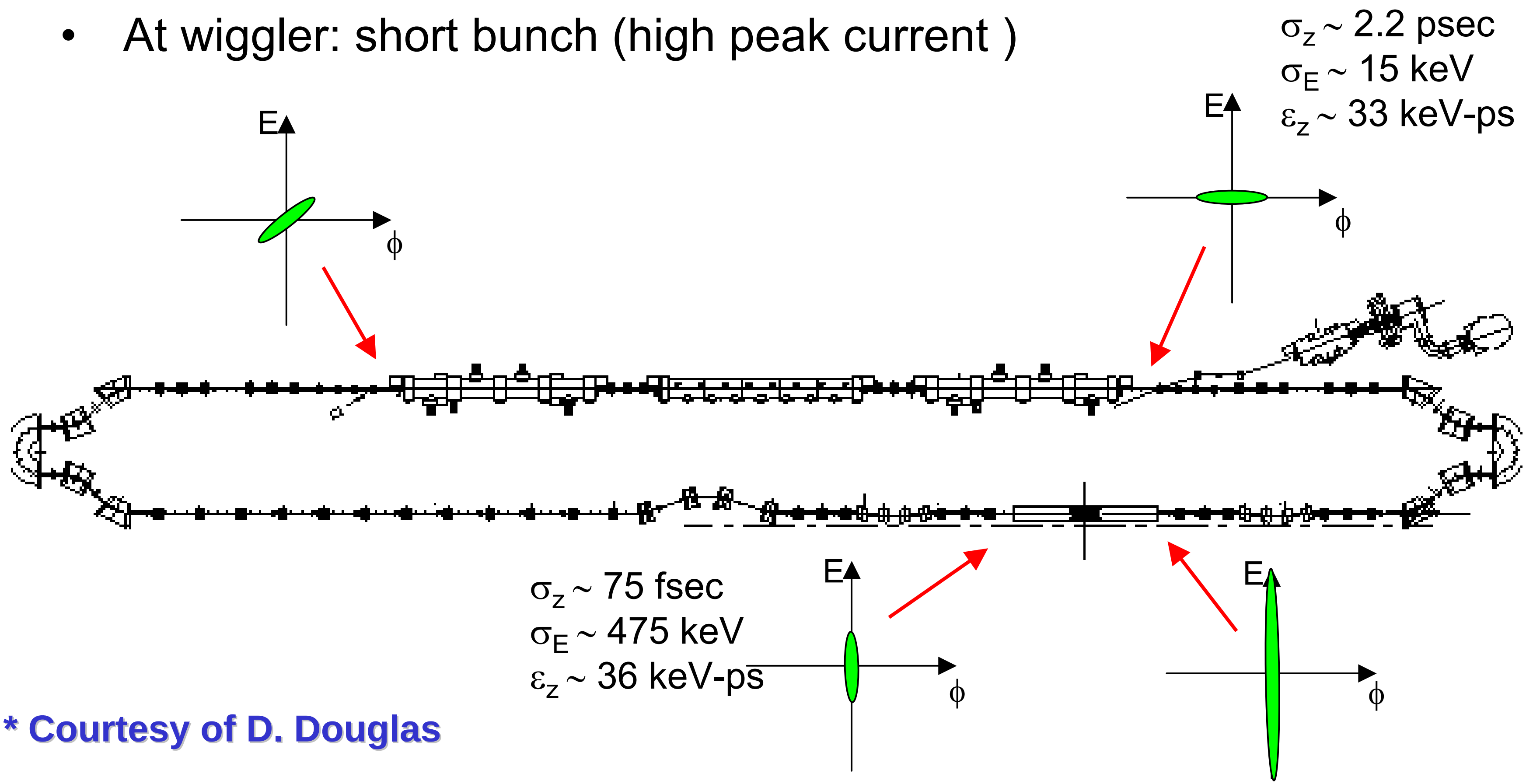
INJECTOR BLOCK DIAGRAM:

It's all about turning knobs and observing what happens to the beam where the viewers are. This is true for both, model and machine.



What does the FEL need from the injector?

- At injection: long bunch, low energy spread
- After acceleration: imposed phase/energy correlation
- At wiggler: short bunch (high peak current)



*** Courtesy of D. Douglas**

How do we setup the injector in the machine and in the model? (Kevin will discuss it later)

1: set the solenoid#2 after the gun

2: adjust cavity#4 for maximum particle energy

3: adjust cavity#3 for maximum particle energy

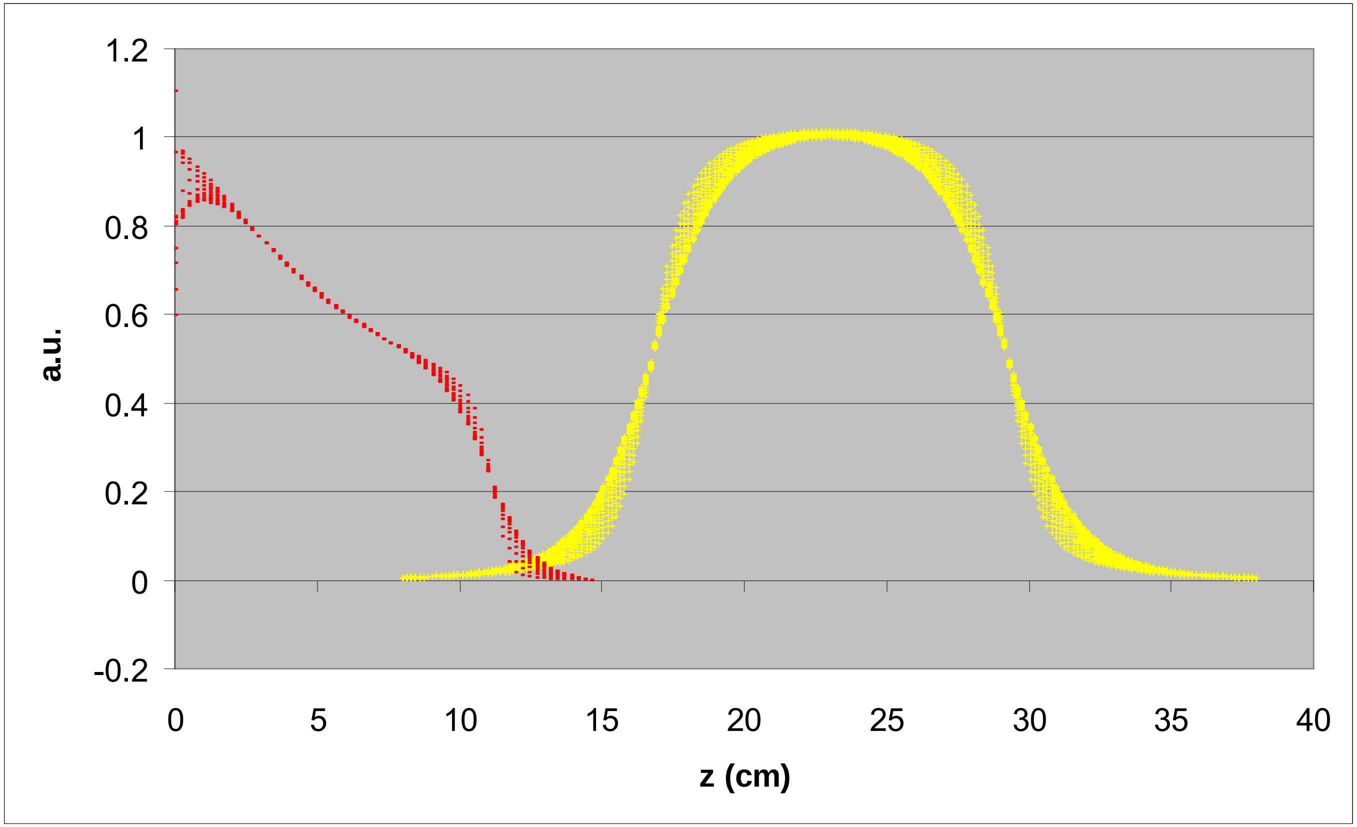
4: add 5° to cavity#4, subtract 10° from cavity#3

5: adjust cavity#3 to recover the gradient to step#3

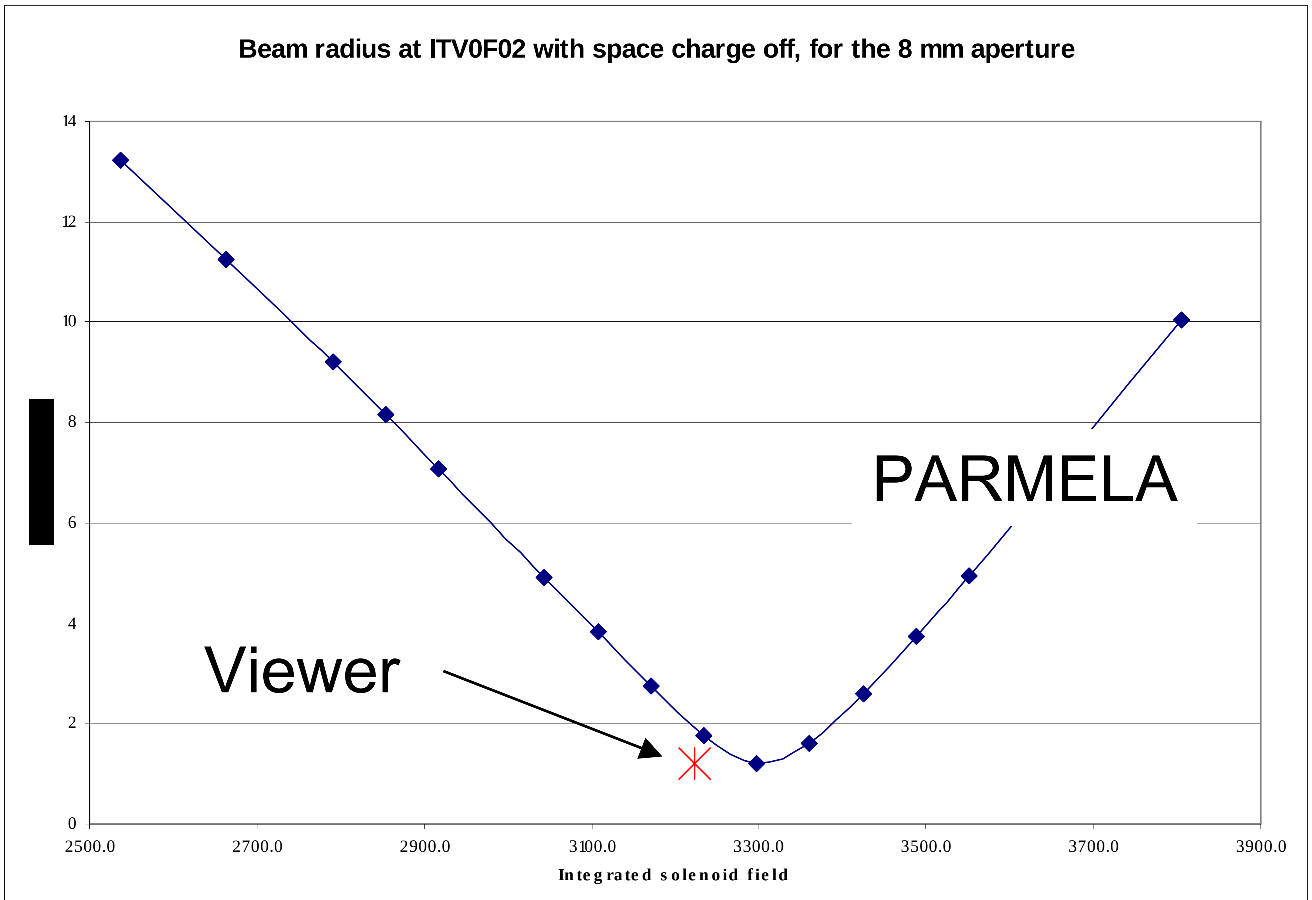
6: adjust the buncher to minimize energy spread at ITV0F06

But before setting up we try to 'connect' the beam behavior in the machine with that predicted by the model using some 'sort' of model calibration and making the source code a little closer representation of the actual machine configuration.

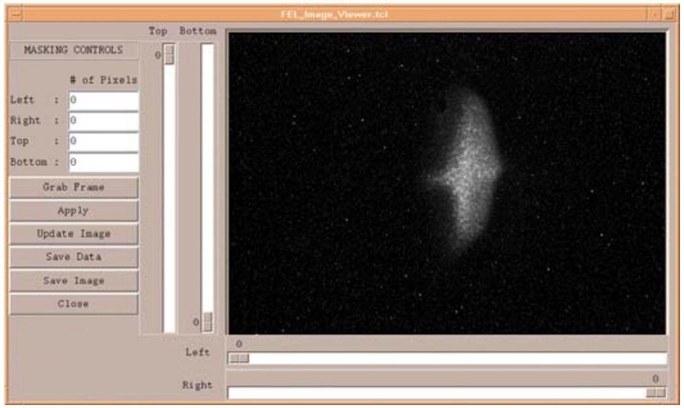
We started by modifying the PARMELA source code to incorporate the overlap between the gun and solenoid fields

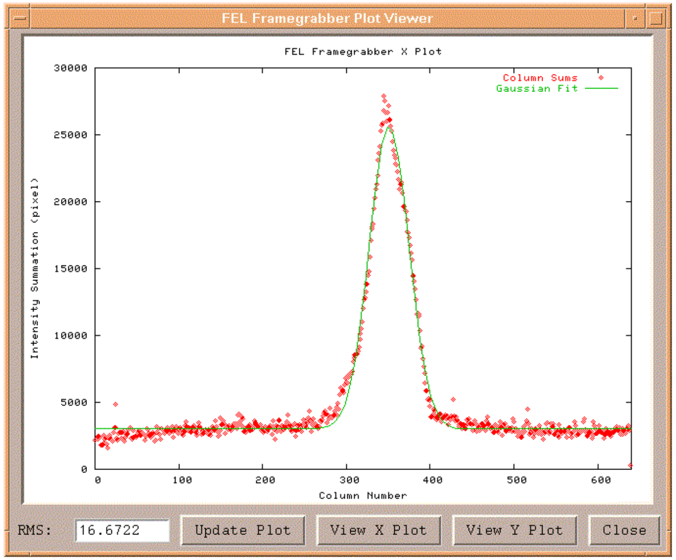


Then proceeded to ‘calibrate the model’ in incremental steps. First by comparing the solenoid strength required to focus the beam to the smallest spot on the ceramic viewer.

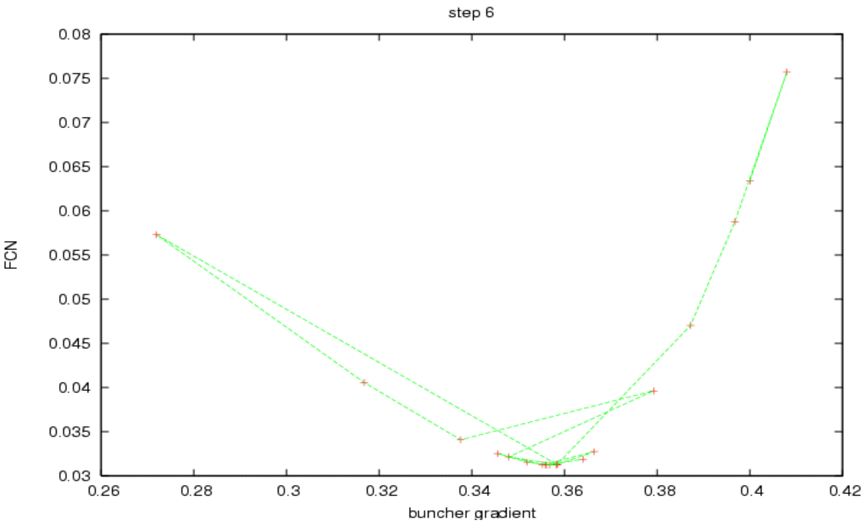


Later the buncher gradient was set to minimize the energy spread at the OTR viewer downstream of the first dispersion section in the injection chicane.

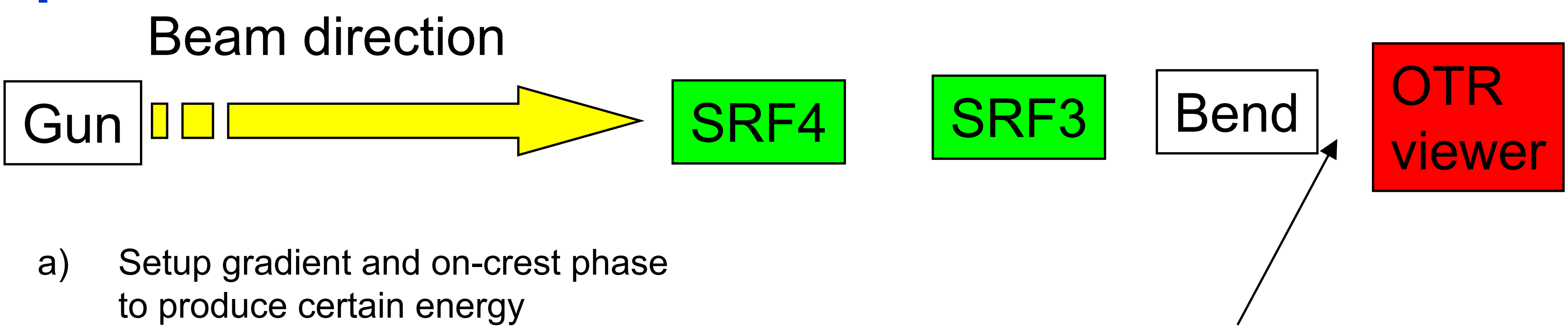




This action is taken in the machine and in PARMELA. The <u>gradient value</u> in both cases is <u>irrelevant</u>, what is important is that model and machine have a common set point



The last of our calibrations so far has been the gradient ratio between the two SRF cavities in the quarter cryounit. Again, the phase and gradient values are irrelevant, what matters is that code and machine have the same gradient ratio and same energy droop in energy for the same off-crest set point



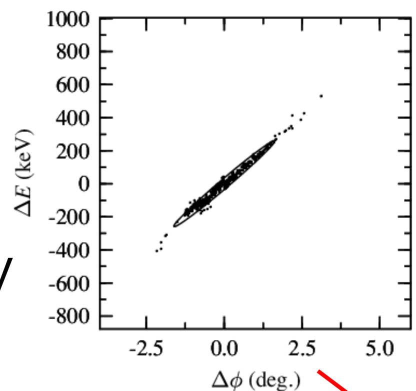
a) Setup gradient and on-crest phase to produce certain energy
b) Change gradients to produce same energy as in (a)
c) Solve two equations with two un-knows, including the transient time factor for SRF4 of course!

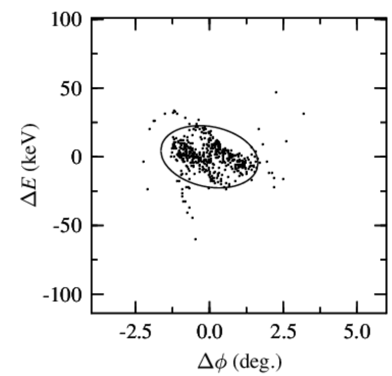
The energy out of the unit is well defined by the dispersion section in the machine and can be compared to the PARMELA value

And after all this calibrations, what's the outcome?

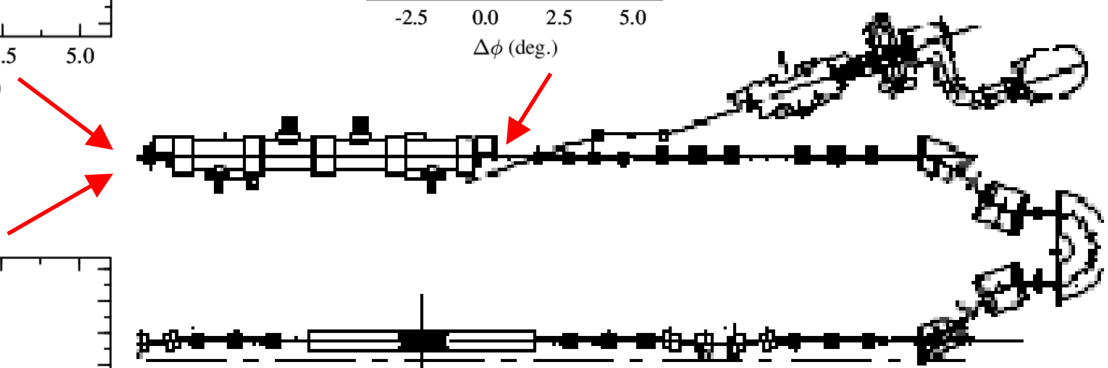
Longitudinal phase space: Good. The bunch length behavior shown by PARMELA as a function of both, buncher gradient and SRF3 phase follows beam-based observations of LSC such as energy spread asymmetry at either side of LINAC crest phase

Ahead of crest acceleration
$4\sigma_E$=520 keV
ε_{nzrms}=26 ps-keV

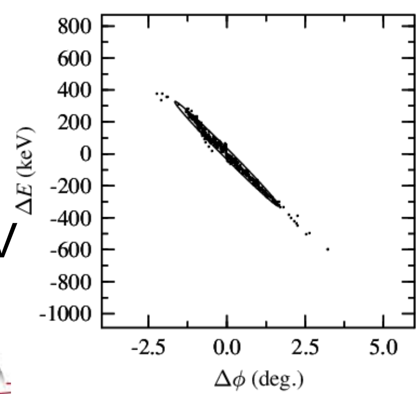




$4\sigma_E$=45 keV at injection
ε_{nzrms}=17 ps-keV

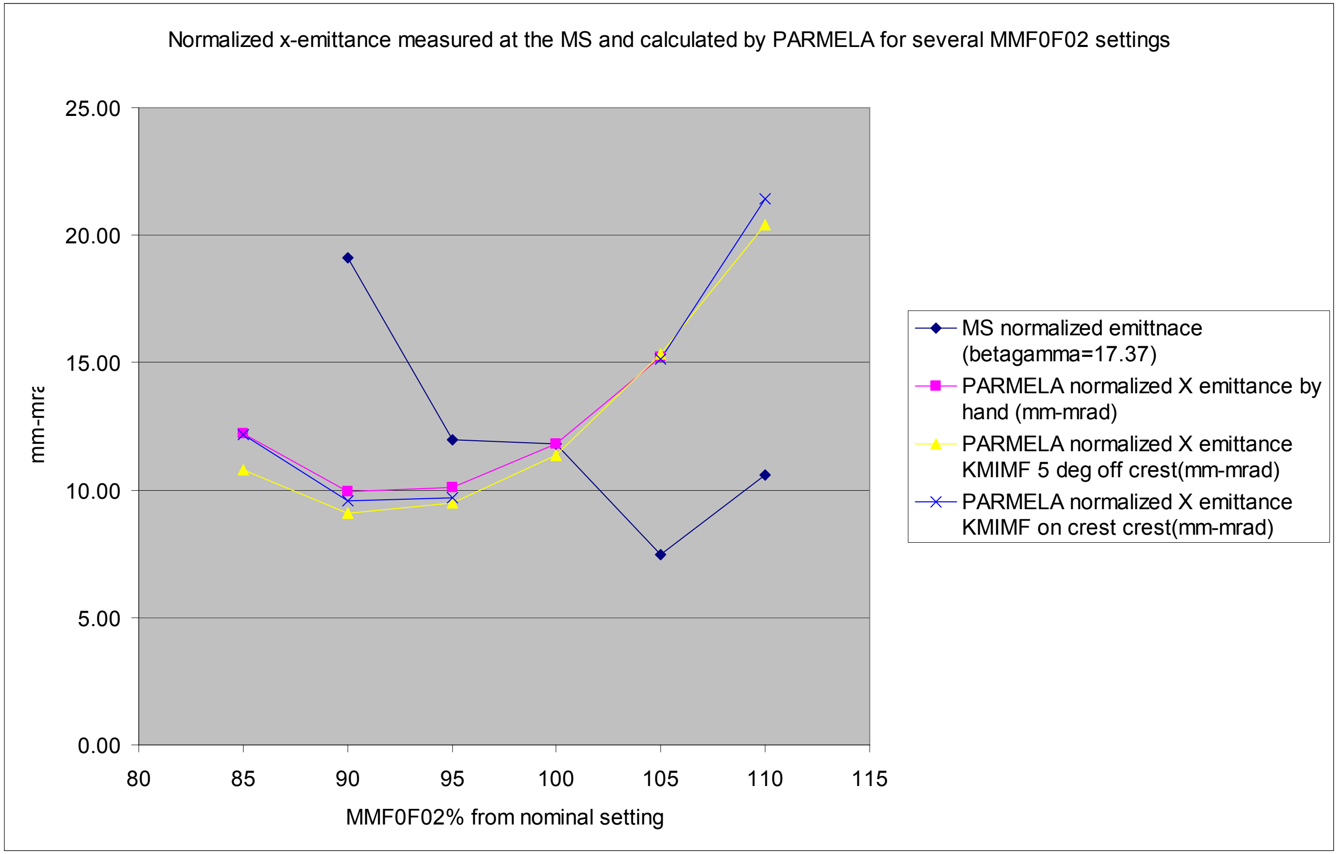


Behind crest acceleration
$4\sigma_E$=659 keV
ε_{nzrms}=27 ps-keV



And after all this calibrations, what's the outcome?

Transverse phase space: Not so good. We are just starting to look at parametric measurements with the Multi-Slit to calibrate the model in the transverse phase space. For example, the emittance below shows there's a "shift" in the model solenoid strength.



But many more parametric MS measurements and model calibrations are still needed:

- Solenoids strength
- Buncher gradient
- SRF cavities gradient and phase
- Drive Laser – Buncher ganged phase
- Quadrupoles strength

What are the uncertainties?

In the machine

- Solenoids, dipoles and quadrupoles, within 1%
- Energy, 0.1% set by injection chicane BDL
- Buncher phase, 5 deg
- Buncher gradient, totally dependent on accuracy of minimum energy spread
- SRFCAV4 phase, 2-3 deg
- SRFCAV3 phase, 0.2 deg

In the model

- POISSON model for solenoid not very accurate in the off-axis field, need very careful mapping
- Buncher RF focusing, actual buncher gradient might different from that in the model
- SRF cavities RF focusing, although gradient ratio is the same as in the machine, again actual gradient might different from that in the model

What tools do we need?

- **optics**
- **space charge**
- **RF cavities**
- optimization
- ...

"When the only tool you have is a hammer,
you tend to treat everything as if it were a nail."

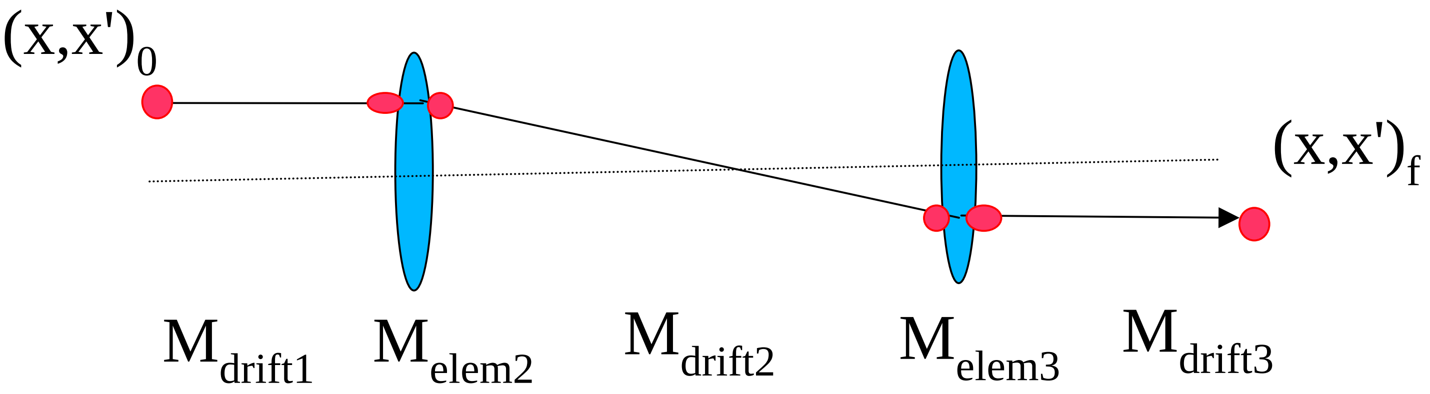
-- *Abraham Maslow*

What tools do we have?

- **`DIMAD`**
- **`parmela-fel`**
- **`kmimf`**
- **`retrack`**
- ...

The right tool for the job!
-- *Engineer Scott*

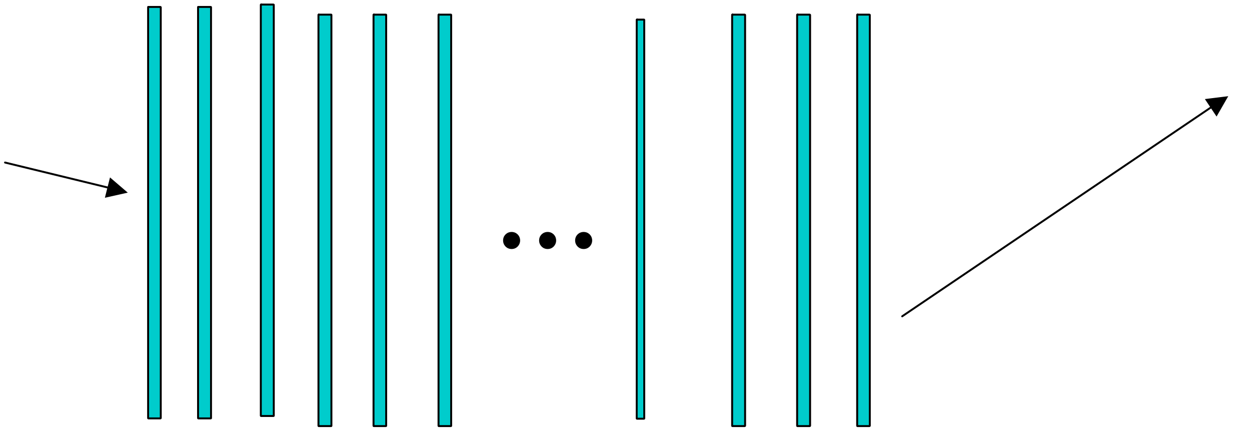
matrix approach



$$(x,x')_f = [\, M_{drift3}\ [\, M_{elem3}\ [\, M_{drift2}\ [\, M_{elem2}\ [\, M_{drift1}\ (x,x')_0\,]\,]\,]\,]\,]$$

$$M_{0f} = M_{drift3}\ M_{elem3}\ M_{drift2}\ M_{elem2}\ M_{drift1}$$

matrix approach continued

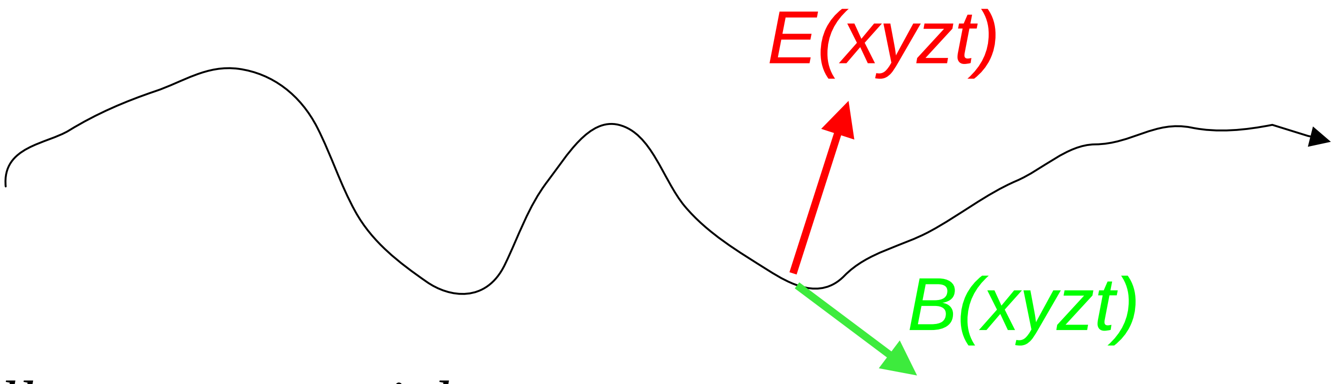


* very fast to calculate, tune, optimize - just matrix multiplication (some nonlinearities can be added in with extra work)

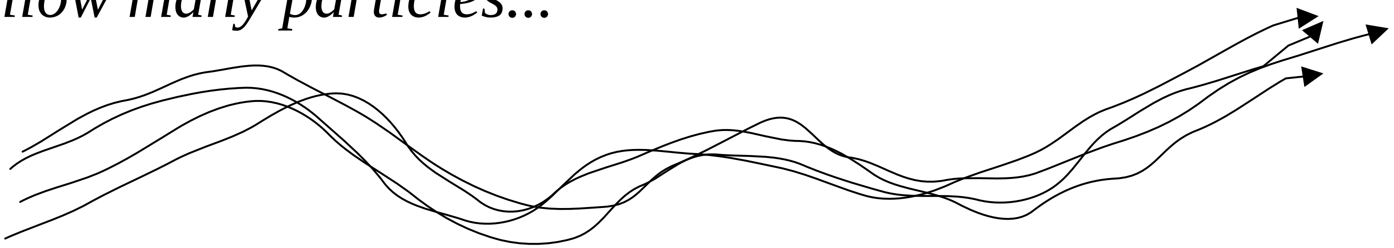
* no particle-particle interactions

particle pusher approach

Follow 1 particle... as a series of small steps

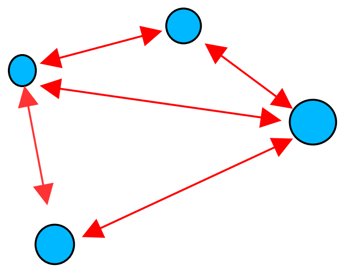


Follow many particles...



but particles can interact with each other - space charge *is important here*

particle pusher approach continued



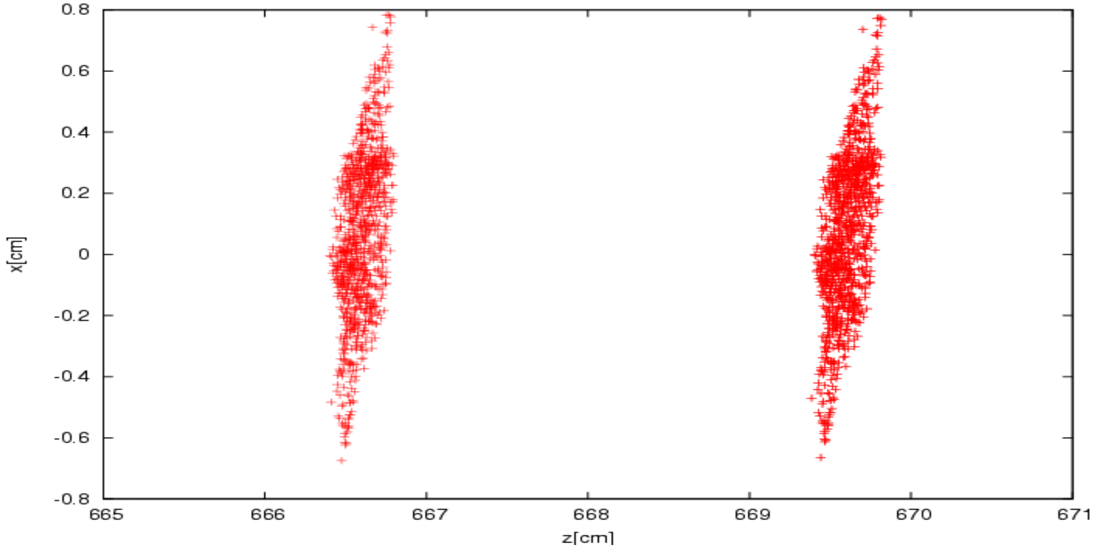
* 1 nC ~ 10^{10} particles -> 10^{20} interactions *(difficult to do)*

common approximations:

* 10^4 pseudoparticles (big, fuzzy balls)
* particle in cell – 10^5 – 10^6 particles and a grid

from injector to the rest of the FEL

particle pusher <------------> *matrix model*



|	0.946E+00	-0.946E+01	-0.190E-02	0.321E+00 |	x
|	-0.385E-03	0.110E+01	0.131E-04	-0.274E-02 |	x'
|	0.220E-02	0.265E+00	0.102E+01	0.882E+01 |	y
|	0.127E-04	0.209E-02	0.556E-04	0.102E+01 |	y'

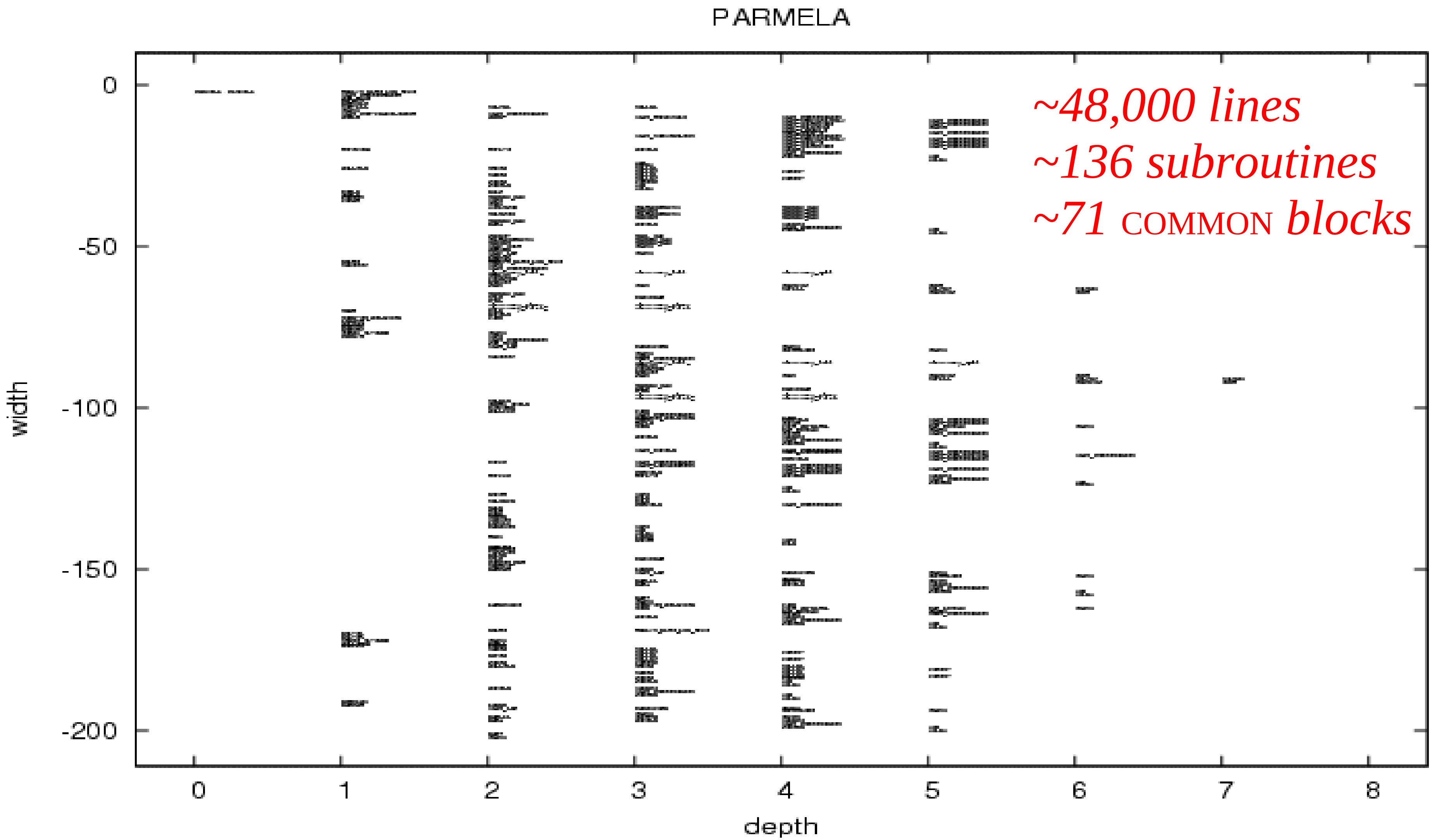
What our goals with the FEL injector?

* a not too short, not too long, bunch
* minimum energy spread
* no scraping

keep one quantity at a time in mind

How do we go about it? ***(in both the real machine and the model!)***

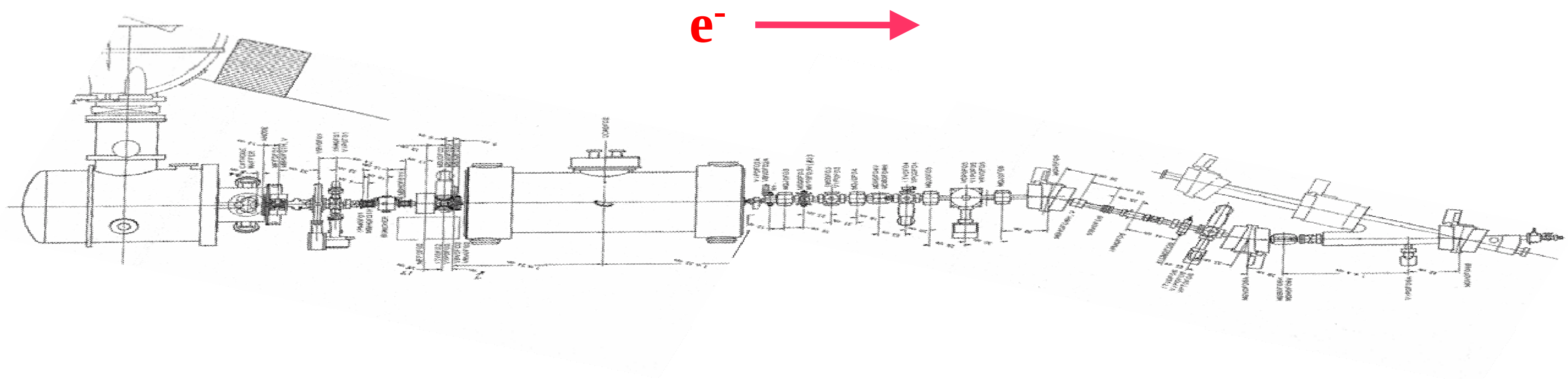
1: set the solenoid#2 after the gun
2: adjust cavity#4 for maximum particle energy
3: adjust cavity#3 for maximum particle energy
4: add 5° to cavity#4, subtract 10° from cavity#3
5: adjust cavity#3 to recover the gradient to step#3
6: adjust the buncher to minimize energy spread at ITV0F06

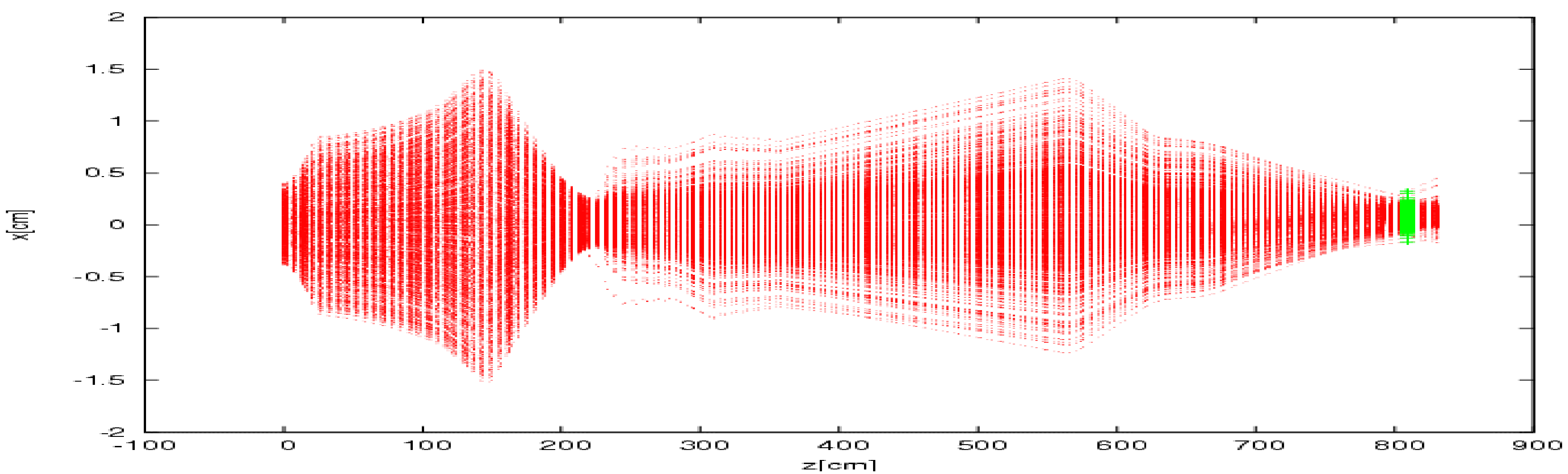


~48,000 lines
~136 subroutines
~71 COMMON *blocks*

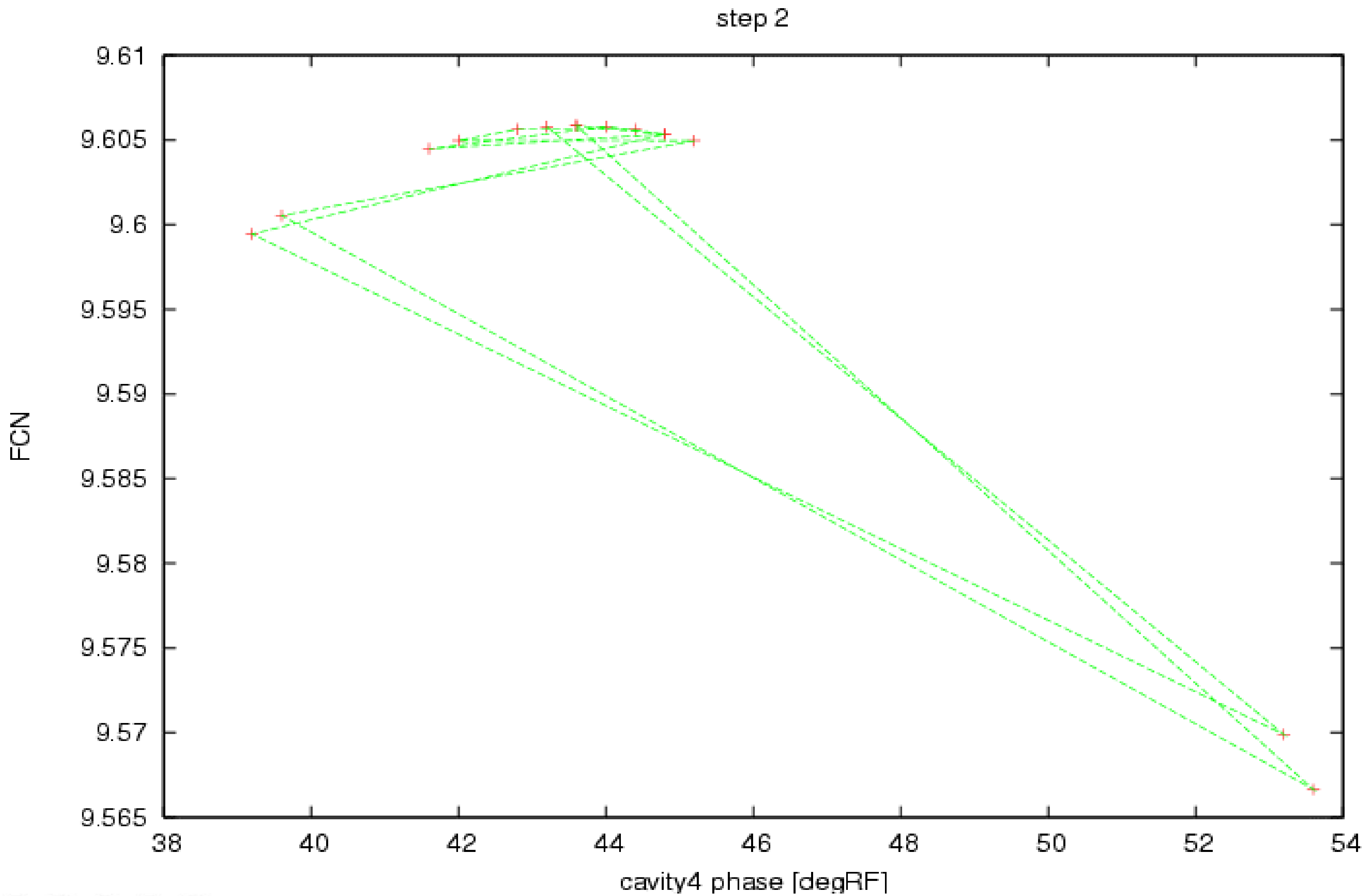
The evil that men do lives after them;
The good is oft interred with their bones;
Anthony and Cleopatra, Act 3, Scene 2

parmela-fel simulation of FEL injector

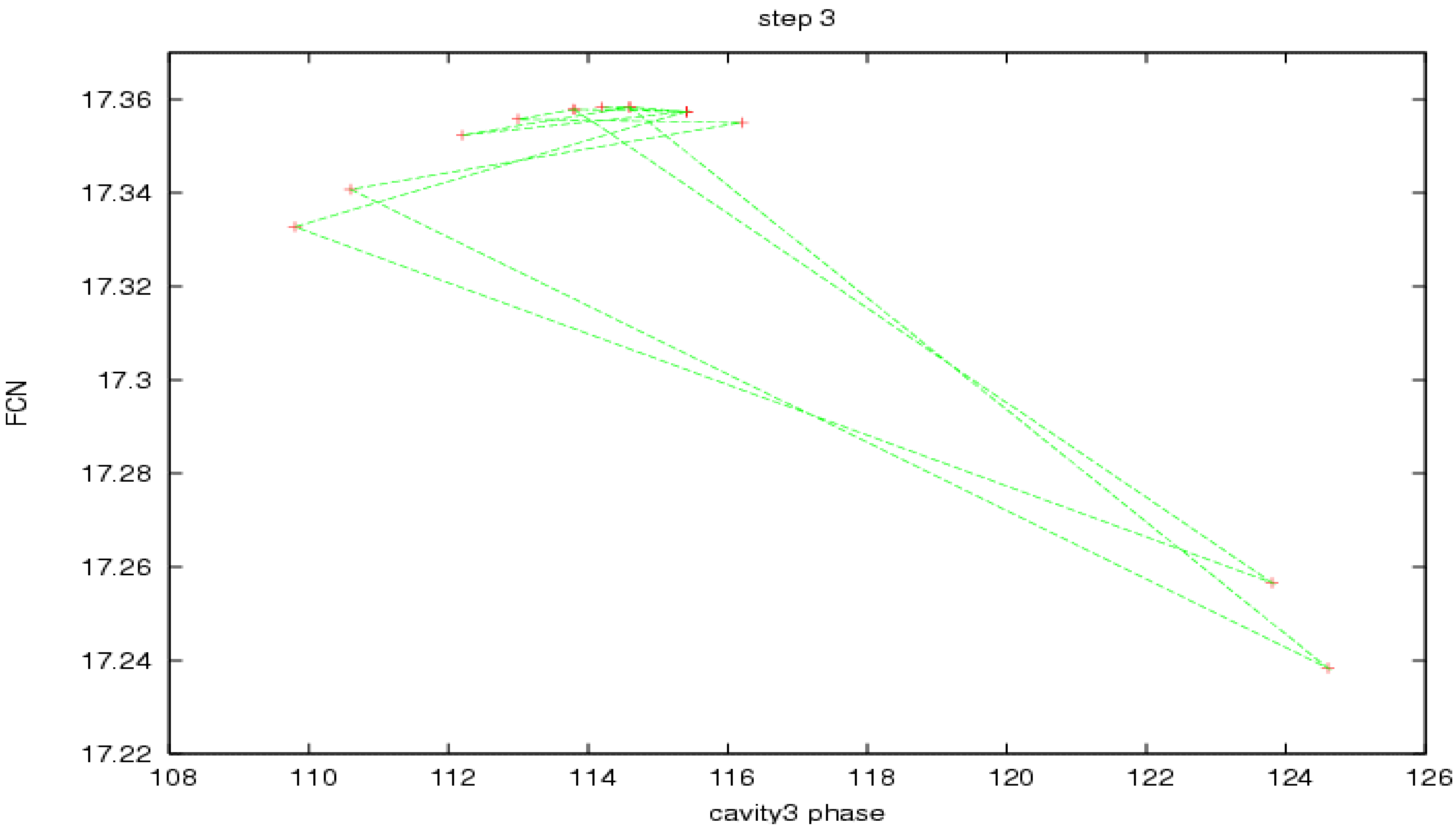




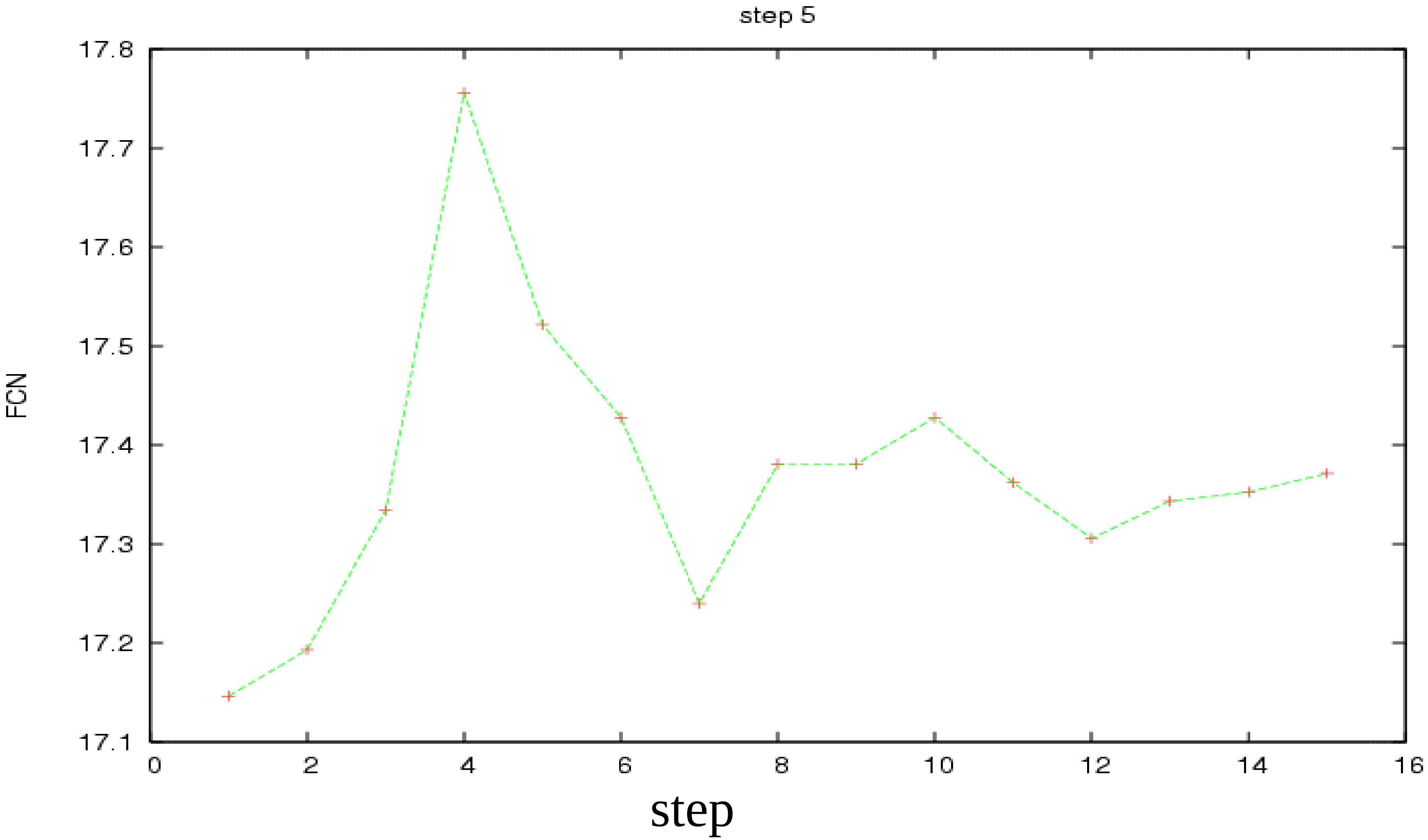
step2 - adjusting cavity#4 phase



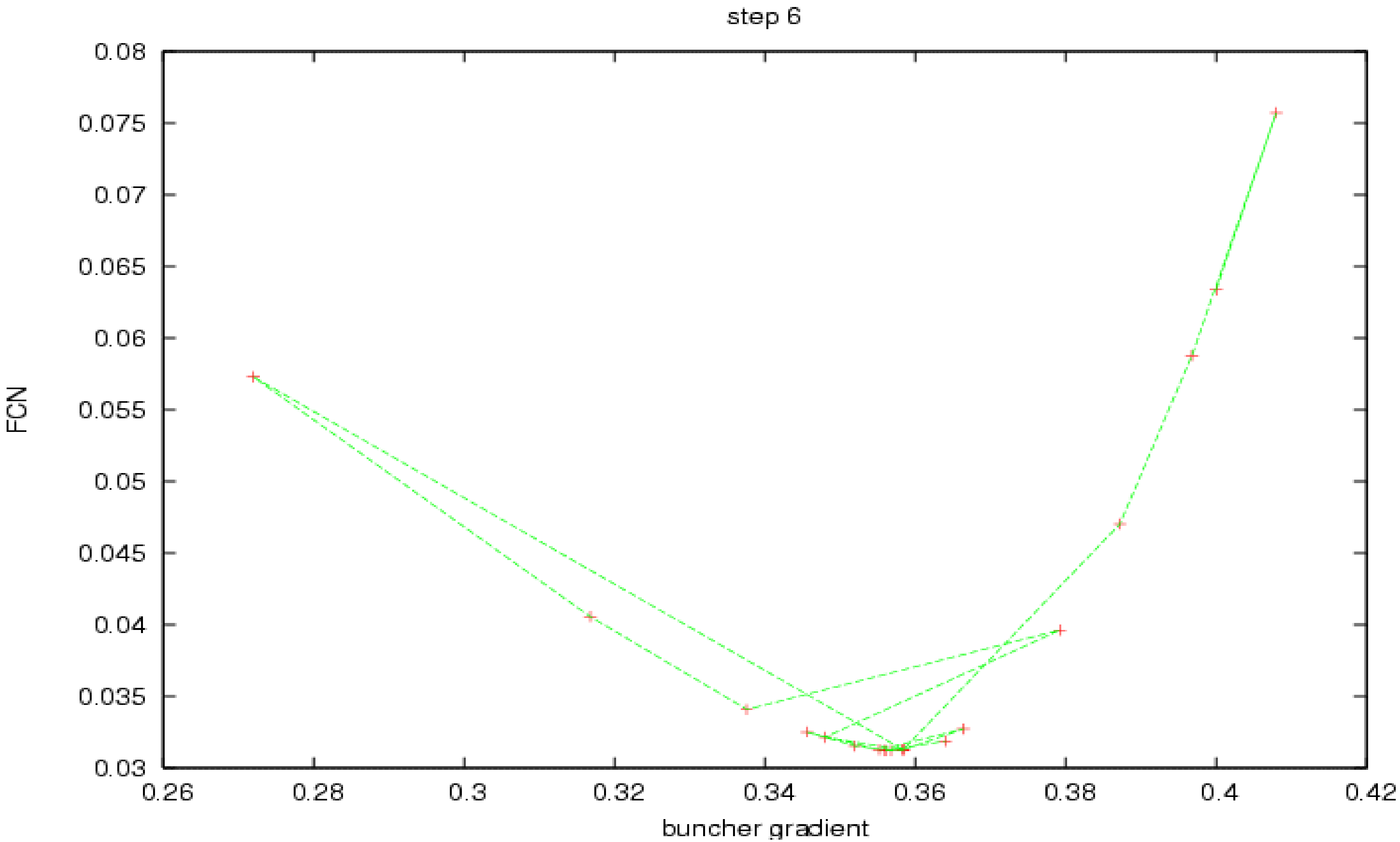
step#3 - adjust cavity#3 phase



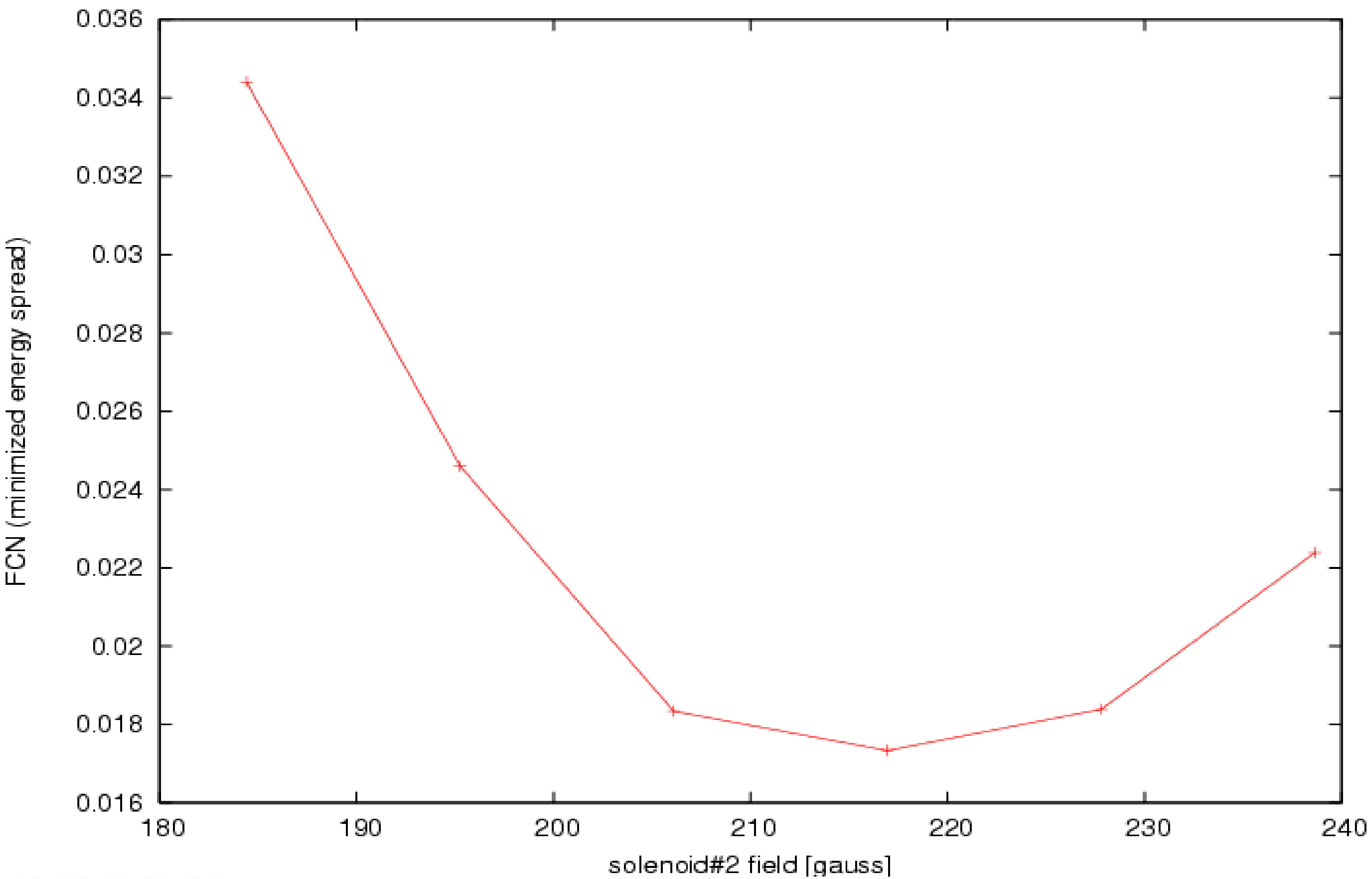
step#5 - adjust cavity#3 gradient to recover energy of step#3



step#6 - adjust buncher gradient to minimize energy spread



summary of minimized energy spread vs solenoid#2 field



Looked easy, didn't it?

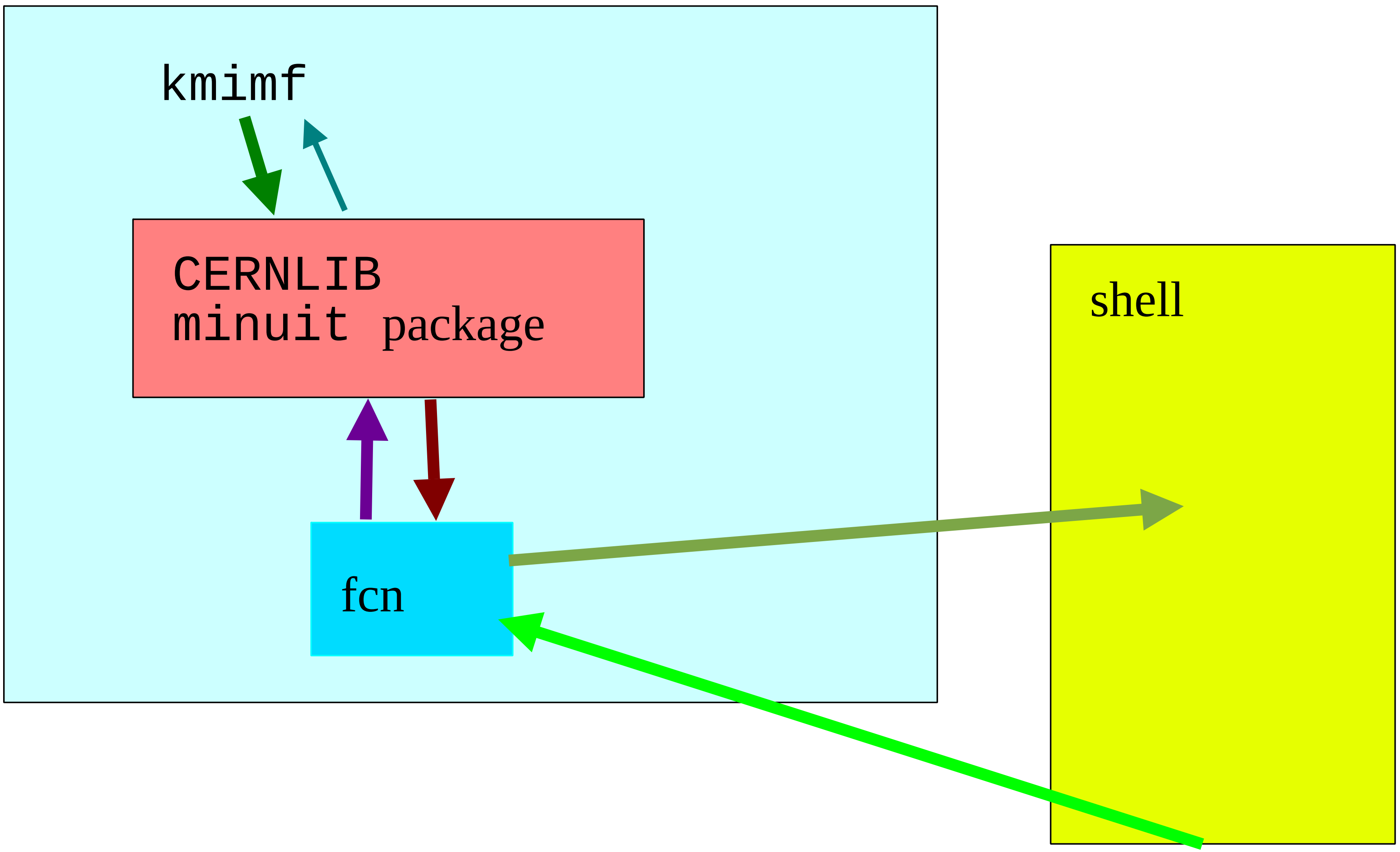
That required ~390 parmela runs; each run took ~40 minutes on a fast Linux box. Rather than have somebody do each by hand, we can use a software tool to make our lives easier. The computers still have to work hard, but we're free to go do something else....

The are a number of generic tools available; recently released ones include `tune`[1], `gminuit`[2], and `kmimf`[3]

[1,2] **http://www.muonsinc.com/index.html#programs**

[3] **file:///u/group/casa/acc_phys/6Dcooling/TOOLS/KMIMF/DOC/index.html**

kmimf - kevin's minimization function



`kmimf` - *continued*

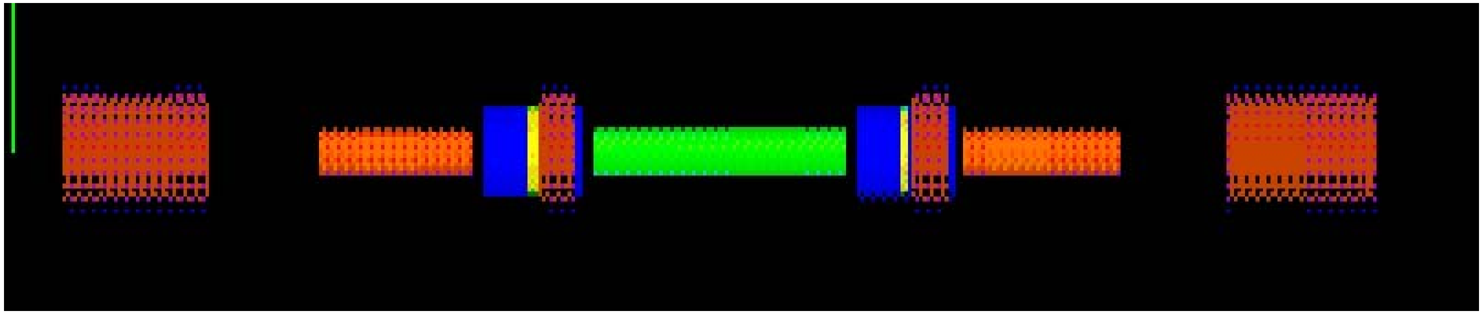
$> **kmimf --help**

```
kmimf -  attempts to minimize the value returned from a shell command
kmimf {options}

 -h  --help                                  - print short help and quit [default value]
 +h  ++help                                  - print longer help and quit
 -V  --version                               - print short version info and quit
 +V  ++version                               - print longer version info and quit
 -v  --verbose                               - display informational messages
 -i  --input          STRING                 - specify command string
 -if --inputfile      FILE                   - read command string from a file
 -M  --MINUIT         STRING                 - specify MINUIT commands (seperated by |)
 -Mf --MINUITfile     FILE                   - read MINUIT commands from a file
 -p  --parameter      NAME:X{:dX{:lo:hi}}    - set parameter start,step, & limits
 -pf --parameterfile  FILE                   - set parameter settings from a file
 -F  --Final          FILE                   - write final best results to a file
     --format         STRING                 - specify a F77 insertion format [(f20.10)]
     --hints                                 - print additional guidance & quit

kmimf 0.1f5, 20mar2006
         http://www.jlab.org/~beard/index.html#muons
```

`kmimf` - *script for a μ cooling channel*



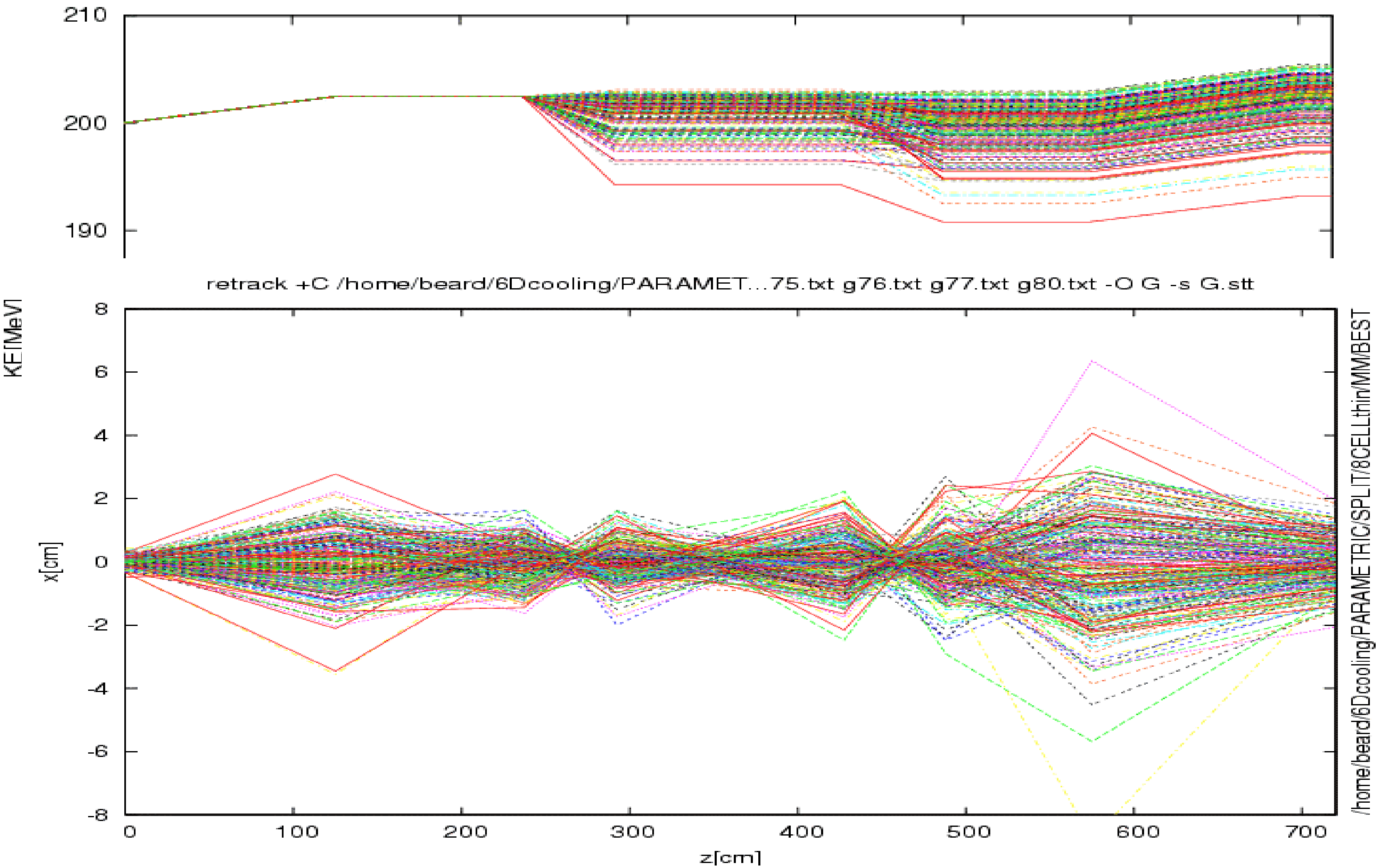
```
$> cat 3c.if
# SCAN4 script for kmimf minimization -
# passed to shell, must return a single #

export TWRK=`autoname -b 1 -f G%04d`;
mkdir $TWRK; cd $TWRK;
echo "g4beamline ../km14.in j1={j1} j2a={j2a} j2b={j2b} \
            t0a={t0a} t0b={t0b} t1a={t1a} t1b={t1b} \
            t2a={t2a} t2b={t2b} t3a={t3a} t3b={t3b}" > in.log;
g4beamline ../km14.in j1={j1} j2a={j2a} j2b={j2b} \
            t0a={t0a} t0b={t0b} t1a={t1a} t1b={t1b} \
            t2a={t2a} t2b={t2b} t3a={t3a} t3b={t3b} >& out.log;
txt2rms -NS +o -i g01.txt; txt2rms -NS +o -i g40.txt;
grep " Pz " g40.txt.rms | beshuffled -I "-#4 #t0a={t0a} $TWRK" | \
                tail -1 | sdup -o g.out
```

`kmimf` - *parameter file*

```
# KM12 search starting points:
# name start step {low limit} {high limit}
   t0a  2.2525894   0.05 #nS timeOffset
   t0b  23.6972198  0.05 #nS timeOffset
   t1a  28.2023987  0.0  #nS timeOffset
   t1b  49.6470276  0.0  #nS timeOffset
   t2a  54.1522095  0.0  #nS timeOffset
   t2b  75.5968384  0.0  #nS timeOffset
   t3a  80.1020142  0.0  #nS timeOffset
   t3b  101.5466492 0.0  #nS timeOffset
   j1 -2509.78          #A/mm2 in center coil
   j2a 2919.675         #A/mm2 in leading coil
   j2b 2919.675         #A/mm2 in trailing coil
```

first cell of the μ cooling channel



back to the FEL injector...

To use the Jlab physics farm, first you need a script:

```
PROJECT: casa

COMMAND: /u/scratch/beard/FEL/MAY06/CMD/sol2_228.com

OS: linux

OPTIONS:

INPUT_FILES:

SINGLE_JOB: true
```

Then you need a command file:

```
...
# maximum PZavg
kmimf +m -pf cav4.pf -if cav4.if -F cav4.finalout -o cav4.log -SN -v -a 15
#
##### step#3: now adjust cavity3 phase
#
echo "#--cav3.pf--" > cav3.pf
echo "# using the previous results, adjust the cavity4 phase" >> cav3.pf
echo "# to minimize the energy spread at z=294cm" >> cav3.pf
#
#  search on cavity#3 phase
grep cav3_phase cav4.finalout | beshuffled -I "#0 #1 10." >> cav3.pf
#  copy everything else excluding comments
cat cav4.finalout | no_bs -nc | grep -v cav3_phase | beshuffled -I "#0 #1" >> cav3.pf
#
# maximum PZavg
kmimf +m -pf cav3.pf -if cav3.if -F cav3.finalout -o cav3.log -SN -v -a 15
...
mail -s `pwd` beard < step6.finalout
#
```

parmela input deck

```
...
!
! Center Line of future BPM distance to be verified
DRIFT /L=23.0 /APER=2.54 /IOUT=1 /TUNE=0.0  !one to find zero crossing phase
! Buncher entrance
DRIFT /L=10.3 /APER=2.54 /IOUT=1
! Distance from wafer surface to CL Buncher is 112.09 cm
!   KMIMF
!   KMIMF                                    bnch_phase=257.5  bnch_grad=0.40
!   KMIMF                                            |             |
CELL V_BUNCHER /L=16.0 CM /APER=2.5 CM /IOUT=1 [bnch_phase] [bnch_grad]
 /NC=2 /DWTMAX=1.0 /SYM=0 /CFREQ=1497 MHZ /CTYPE=1 /BZ=0.0 /NFC=14 /VV=1
 0.1755805E+01,0.7670868E+00,-.8749583E-02,-.1692334E+00,
 -.6601132E-01,0.1124237E-01,0.1969989E-01,0.5417380E-02,
 -.2206802E-02,-.2076432E-02,-.3329378E-03,0.3107045E-03,
 0.2088789E-03,0.2500894E-04 15 71
DRIFT /L=10.84 /APER=2.54 /IOUT=1
!   KMIMF
!   KMIMF                                           sol2_fld=-216.95
!   KMIMF                                                  |
POISSON V_MMF0F02 /L=30.0 /APER=2.54 /IOUT=1 /NC=2 /WR=0.1 [sol2_fld]
       /sym=0 /bftype=0 /dwtmax=1.0
BFIELD 2
sln80
! ITV0F02
...
```

scripts...

```
# adjust cav4 phase to maximize energy
# at exit (z=294cm)

export TWRK=`autoname -b 1 -f cav4__%04d`;
mkdir $TWRK; cp *fld pwig.dat $TWRK; cd $TWRK;

cat ../FELmar06_template.IN | no_bs  \
         -s '[gbl_Npart_]' "{gbl_Npart_}" \
         -s '[bnch_phase]' "{bnch_phase}" \
         -s '[bnch_grad]' "{bnch_grad}" \
         -s '[sol2_fld]' "{sol2_fld}" \
         -s '[cav4_phase]' "{cav4_phase}" \
         -s '[cav4_grad]' "{cav4_grad}" \
         -s '[cav3_phase]' "{cav3_phase}" \
         -s '[cav3_grad]' "{cav3_grad}" \
                            > $TWRK.in;

printenv|grep PML > out.log;
pwd >> out.log;
/u/scratch/beard/FEL/MAY06/PML/pml $TWRK >> out.log;

retrack -F P -i *.KBB +zL 293. +zH 298. -oP out.kbb >> out.log;
txt2rms -F -i out.kbb -o out.kbb.rms -NS >> out.log;

gzip -9v *.KBB >> out.log;
rm -rf ../cav3.out.kbb.rms; cp out.kbb.rms ../cav3.out.kbb.rms;
# 0:index 1:name 2:low 3:high 4:mean 5:RMS
grep zp out.kbb.rms | tail -1 | beshuffled -I 4;
```

retrack – a tool to handle simulation output

What can retrack do?

* convert simulation data formats
* select subsets of the data
* gather statistical information
* generate cannonical data
* generate plotting command files
* fit 4D Twiss functions to the data
* fit 4x4 matricies to the data

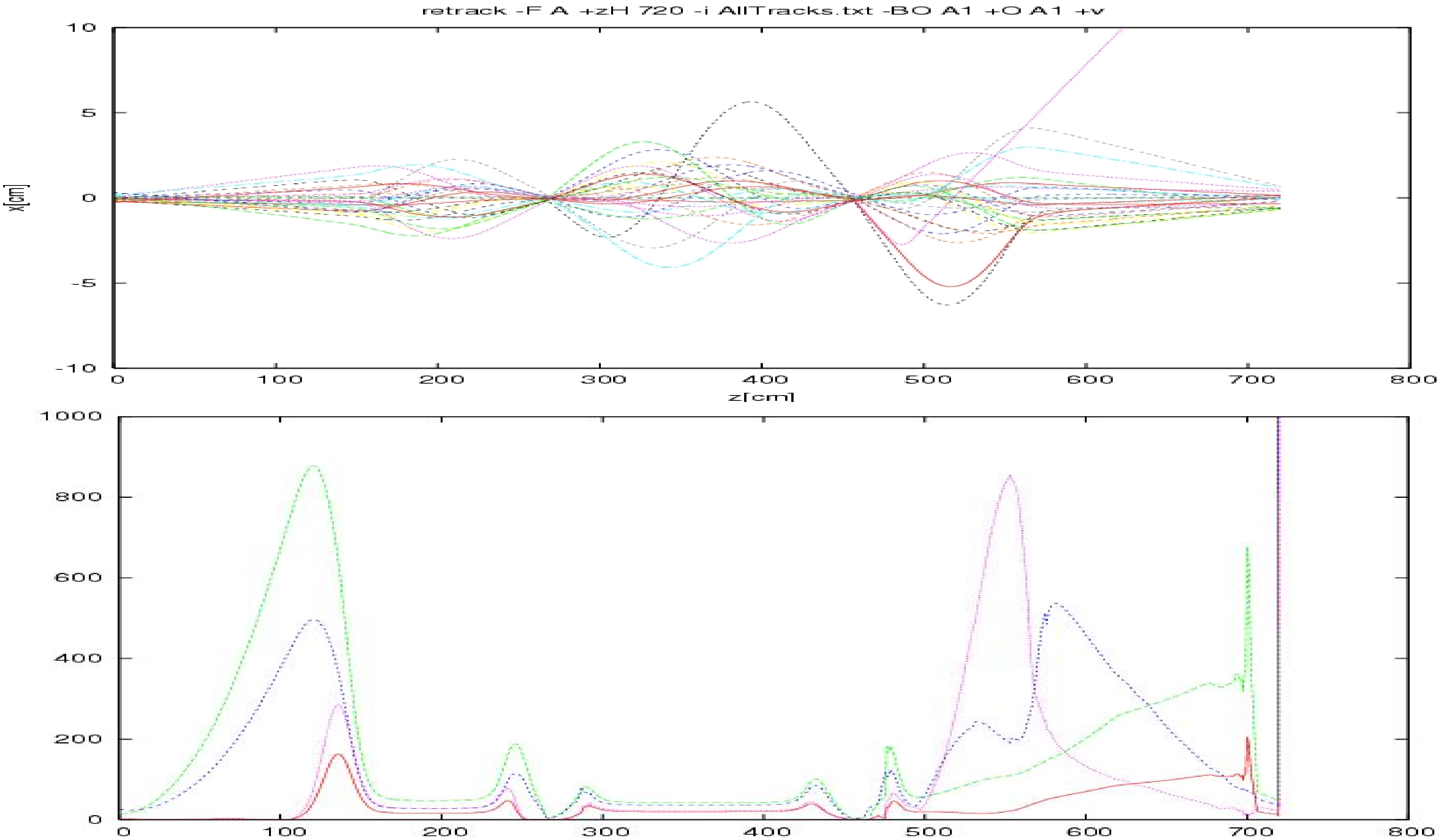
...

data formats read by `retrack`

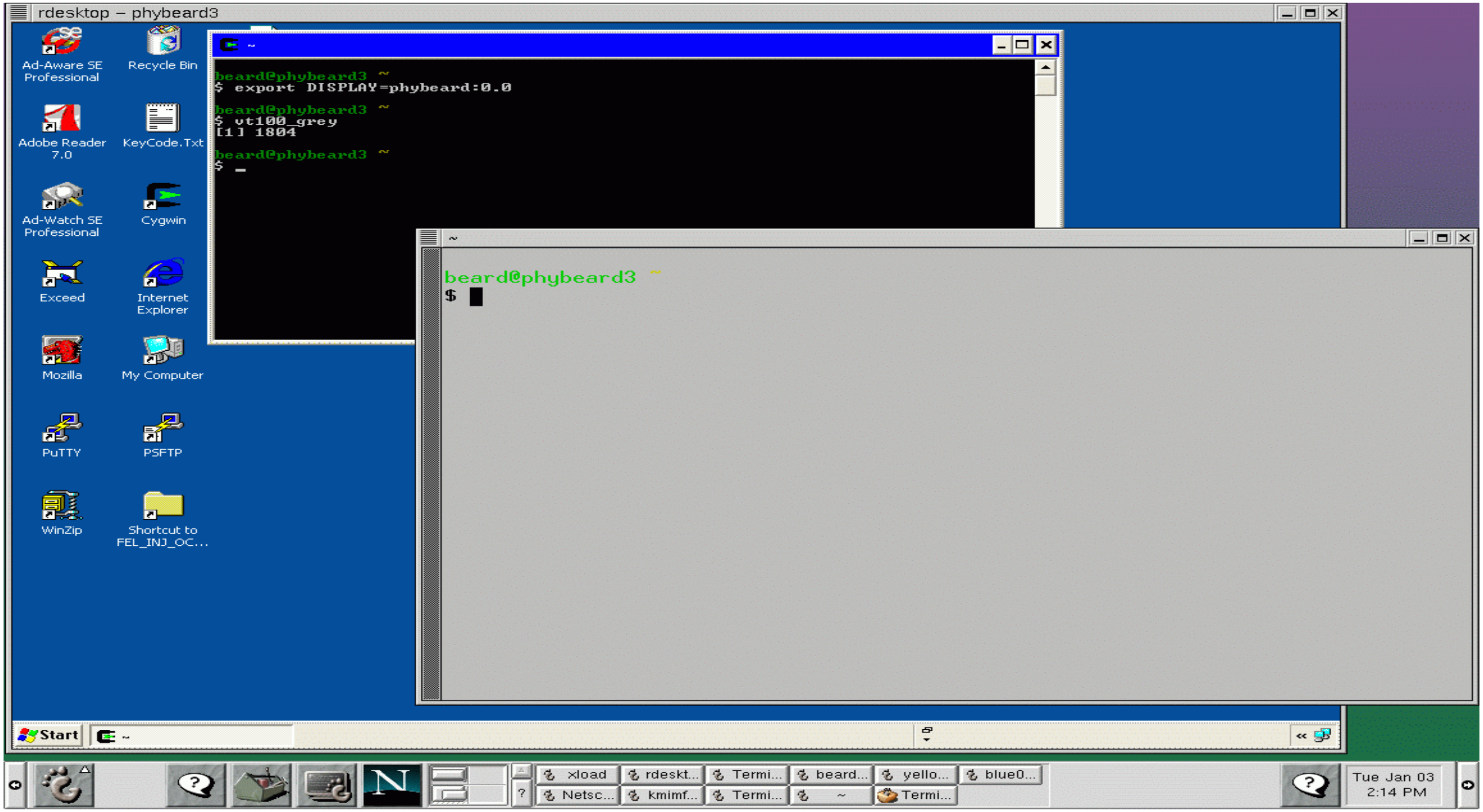
- **Rtext** - `retrack` text format
- **Rbinary** - `retrack` binary format
- **G** - `g4beamline` virtual detector format
- **V0** - `g4beamline 0.9x` verbose format
- **V1** - `g4beamline 1.02` verbose format
- **A** - `g4beamline 1.0x` alltrace format
- **I** - `ICOOL for009` format
- **O** - `OptiM` track format
- **P** - TJNAF `Parmela` KBB format
- **T2** - LANL `Parmela` TAPE2 format
- **T3** - LANL `Parmela` TAPE3 format

'The old order changeth, yielding place to new,
and God fulfils himself in many ways,
lest one good custom should corrupt the world'.
-- Tennyson

4D Twiss functions from a μ cooling cell



new (LANL) **parmela** (used for the new AES injector) *only* runs under Windows, so just run **kmimf** in **cygwin** under Windows too




```
$> kmimf -pf 3jan06.pf +p -if 3jan06.if +i -o 3jan06_1344.log -v | \
    sdup -o 3jan06_1344.syslog
```

other applications around Jlab...

Serkan Golge et al - positrons in the CEBAF injector
(**`g4beamline + parmela-cebaf + kmimf`**)

David Newsham et al - modeling of a dispersive muon cooling line
(**`OptiM + tune + g4beamline`**)

Y.Derbenev's muon parameter resonance ionization cooling line envisioned using A.Bogacz' model
(**`OptiM + g4beamline + kmimf`**)

FEL matching
(**`parmela-fel + dimad + elegant? + bmad?`**)

Sometimes more than half the job is finding the right tool and learning how to use it.

Man must shape his tools lest they shape him.
-- *Arthur R. Miller*

"Those who will be able to conquer software
will be able to conquer the world."
-- *Tadahiro Sekimoto, president, NEC Corp.*

Where to start looking...

`http://casa.jlab.org` --> internal --> code library
`http://www.muons.inc` --> programs
`http://www.jlab.org/~beard`