The State of the ETEAPOT Code (*E*lectric *T*hin *E*lement *A*ccelerator *P*rogram for *O*ptics and *T*racking), and its Context/Environment

J. Talman

January 25, 2013

Abstract

An overview of ETEAPOT is presented. A brief introduction continues with explicit instructions and examples of repeatable output. This allows simple direct verification of benchmarked work (both transverse and longitudinal). While the betatron/transverse propagation is well in hand ([1], [2]), the longitudinal/synchrotron is less established. Allowing new physicists to exploit this functionality fairly easily; I regard this as the main result of this project (to date). Physics implications can be expected to follow. For example, the need for reduced dispersion has already been suggested. Significant constraints remain. For example "modest curvature" lattice elements (-1 $\leq$ m $\leq$ 1) have turned out to be nearly equivalent for our purposes. However, some recent work ([3]) indicates that what's needed is m values more than 100 times this big! This make the "Bend, Kick, Bend" algorithm approach all the more essential. This report aims to flesh all this out.

1 Introduction

As seen at URL http://code.google.com/p/ual/source/list?num=25&start=25 the "Initial directory structure" for UAL was created May 6, 2008. N. Malitsky was the primary repository contributer until Aug 30, 2009 (UAL::243) or so. This is a very powerful/robust software architecture accomodating arbitrary accelerator lattice descriptions (assuming they permit a closed orbit). I have been the primary repository contributer since (though in the restricted domain of all electric accelerators). "*E*lectric *D*ipole *M*oment latest" (UAL::272 - UAL::984 or so) is at URL http://code.google.com/p/ual/source/list . Roughly speaking (and this is all a little rough), Nikolay created/imported the original TEAPOT (mostly magnetic) functionality as UAL and I have been grafting on the electric lattice capability. The UAL architecture underpins all my work. Figure 1 is an iconic representation of its foundational nature.

Expanding slightly, the acronym Accelerator Propagator Description Format shown there "enables" any and all ($\forall$) lattice element algorithms! I'm exaggerating, a little!

Expanding slightly again, the acronym Standard eXchange Format enables any and all ($\forall$) lattice elements! I'm exaggerating, a little, again!

Probably the most important modification to the above synopsis is the central role of R Talman. He was central to the progenitor TEAPOT code, the philosophy of which circumscribes the UAL repository. Of course, Richard has been central to the entire UAL (essentially all 983 revisions) development. Of course my involvement is due to him.

Probably the second most important modification to the above synopsis is the role of A Luccio and SPINK. The proof of concept of "adding" spin to the UAL architecture interested some important people.

Fanglei Lins name should be mentioned as well.

Nikolay is likely winding down his UAL involvement, and I might be, as well.

Again, my role has been to augment the existing magnetic functionality with electric (bend) functionality. Possible scenarios/trajectories for this effort are

(T1) I drop out, and the only result is the improved ability of Richard to help create a satisfactory electric storage ring for the various EDM experiments (eEDM, pEDM, ...)

(T2) I drop out, and the code continues to help Richard, and possibly other physicists

(T3) I continue developing the code, but the effort remains mostly confined to me and Richard

(T4) I continue developing the code, become part of the relevant community, and the code develops a foothold

With an eye on handing off this work, this report targets scenario (T2) in/on the off chance that some fairly independent physicist(s) attempt to use it.

Possibly it helps me achieve (T3) and/or (T4).

2 Graphical, somewhat whimsical, restatement

Figure 2 presents an iconic portrayal of (T1): UAL, at least the electric part, closes. Presently, Richard moves on.

Figure 3 presents an iconic portrayal of (T2): UAL/ETEAPOT is but a small part of a general tool set. Physics is open ended (ie the primary domain).

Thus, with an eye on adding UAL/ETEAPOT to some physicists tool boxes, I turn to more concrete specification.

3 Computer Syntax for Repeatable Results

The exact steps:

```
svn co http://ual.googlecode.com/svn/trunk/ ual1
cd ual1
tcsh
setenv UAL `pwd`
source setup-linux-ual
```

```
make clean
make >&! make.log

cd examples/ETEAPOT
cat README
```

and the inexact step:

```
"follow the relevant instruction file"
```

are designed to lead to repeatable results.

For now, I will follow one of the three instruction files referred to in README,

runExampleFor_ETEAPOT_MltTurn

as it targets previously benchmarked longitudinal/synchrotron motion. It looks like:

```
cp Shell.cc $UAL/ext/UI/src/UAL/UI/Shell.cc
pushd $UAL/ext/UI
make clean
make
ll lib/linux            // e.g. 1154818 libUalUI.so
popd

cp $UAL/codes/ETEAPOT_MltTurn/src/ETEAPOT_MltTurn/Integrator/RFCavityTracker-mod.cc
   $UAL/codes/ETEAPOT_MltTurn/src/ETEAPOT_MltTurn/Integrator/RFCavityTracker.cc

pushd $UAL/codes/ETEAPOT_MltTurn
make clean
make
ll lib/linux            // e.g.  574307 libETEAPOT_MltTurn.so
popd

cp data/eteapot_MltTurn.apdf data/eteapot.apdf

make outputReady
make clean
make

// We recommend the lenovo ThinkPad line with ubuntu, or Scientific Linux
// On an ubuntu single processor vostro 200, 12/31/2012, this next command
// takes 10 minutes or so, and generates a 3G or so file
./ETEAPOT_MltTurn ./data/E_BM_M1.0.RF.sxf -1 40 1 500 >! OUT
// On an ubuntu single processor vostro 200, 12/31/2012, this next command
// takes 10 minutes or so, and generates a 3G or so file

// "look" at NikolayOut
perl SplitNikolayOut.pl NikolayOut >! IG

gnuplot
   gnuplot> l 'forGnuplot_Figure2'
```

4 Results/"Ongoing Longitudinal"

When these instructions are adhered to religiously (easier said than done), gnuplot gives Figure 4. The reader can take my word that this is identical to Figure 2 in [13], or he can verify it for himself. This latter is my preferred option as it facilitates ETEAPOT becoming part of the tool kit.

The original description in [13] is

```
Longitudinal dynamics in lattice BM M1.0.RF (having m=-1) is shown in Figure 2. (For the outermost
amplitude the initial conditions have been adjusted to eliminate the initial betatron amplitudes.) The
three benchmark lattices are compared in the similar plots of Figure 3. For this comparison only the
three innermost amplitudes are tracked. In these plots three particles start at the origin traveling in the
forward direction, with fractional energy offsets UAL = 0.000001, 0.000002, 0.000003. The three plots
are essentially identical. We interpret this to mean that the quadrupoles adjacent to the bend elements,
having been adjusted to restore the vertical tune to its standard value Qy = 0.2, effectively cancel the
focusing effect coming from non-zero field index m. In other words, as far as longitudinal dynamics is
concerned, the three benchmark lattices are equivalent.
```

A similar procedure (only difference being l 'forGnuplot_Figure11' in gnuplot) duplicates Figure 11.

The original description in [13] is

```
A possibly surprising aspect of the oscillation is illustrated by the bottom figure of Figure 11. The
tracked particle executes longitudinal oscillations of amplitude 0.5 m. This graph can therefore be
usefully compared with the outermost of the orbits shown in Figure 2. Both have a similarly large
ct-ranges. The tracked particle starts side-by-side with the reference particle, but with too little speed
(UAL = 105), as can be seen from Figure 2. Acting as if above resonance its shorter path length
dominates is too-low speed, and it takes the tracked particle only 25 turns to get to the front of the bunch,
as can be seen from Figure 11. But then it takes about 50 turns to get back level with the reference
particle, and 50 turns more to get to the tail of the bunch. The time distortion in Figure 11 therefore
corresponds to the up-down asymmetry in Figure 2
```

The preceding is only the (probably most important) tip of the iceberg.

Richard and I have generated a whole series of benchmark results ([9], [10], [11], [12], [13]). These can, in principle, all be recreated by the methods outlined above.

The instruction file

runExampleFor_ETEAPOT_MltTurn

is only one of three mentioned in file $UAL/examples/ETEAPOT/README

Others are

determineTwissCleanRunPlotM1.0_sl4

runExampleFor_orbitsWithSpin

Results from file

determineTwissCleanRunPlotM1.0_sl4

are presented in the "Verified Transverse" section below.

These other "near scripts" are touched on as well.

5 Gotchas

This is all a little error prone. I'm working on improving it! The consensus seems to be that UAL is difficult, if not "dang near impossible" [2], to use.

This whole approach is predicated on the work of Munoz and Pavic [4]. The time of flight calculation, in particular, should be looked at more closely. One could take a different point of view, that I have been evaluating their work. Richard and I feel that it is at least mostly correct. There's a Munoz Evaluation section that goes into more detail on this.

My sense is that Richard and I will normalize to the Lenovo ThinkPad line.

Furthermore, we'll probably impose an operating system, Scientific Linux, as well.

There seems to be some (non show stopping) inconsistencies among the various potential UAL platforms. These we've learned to live with. There seems to be some (show stopping) inconsistencies among more widely divergent platforms. These have to be filtered out.

Error prone files, and some of their contents include:

$UAL/examples/ETEAPOT/designBeamValues.hh
particle type ⇒ mass (m0), charge (q0), ...
design frequency, f0
$UAL/examples/ETEAPOT/extractParameters.h
split values, e.g. splitForBends
$UAL/examples/ETEAPOT/probeDataForTwiss
monodromy values

Directory

$UAL/examples/ETEAPOT/data

has sample apdf files e.g.

$UAL/examples/ETEAPOT/data/eteapotMARKER.apdf
$UAL/examples/ETEAPOT/data/eteapot_MltTurn.apdf

which must be copied to

$UAL/examples/ETEAPOT/data/eteapot.apdf

as the current client side code is written. The "near scripts" mentioned in README specify this, but it's still easy to forget!

It's often unnecessary, but I like to automatically run

make outputReady
make clean
make

before most runs as I sometimes open files for appending so that ones results may be from more than one simulation.

6 Discussion

Again, I am here targeting my "scenario (T2)" which involves other people potentially using this code and related material.

I have gathered all the documents cited below electronically. How long this will be available is anybodies guess!

Of course the google codes repository is likely to be around for a while! I have checked in these supporting documents. The url for this paper is `http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/statusAndRegression/stateOfETEAPOT.pdf`.

Again Figure 4 is simply recreatable, and duplicates previously benchmarked work. [13] shows that this "m=-1" ("saddle") lattice is nearly indistiguishable from "parallel" (m=0) and "toroidal"

(m=1) lattices. This is within the constraint of small ($Q_y \approx 0.2$) values (so that the vertically focussing quads in the various lattices are quite different) for the vertical tune. Obviously very important if true! A corollary might be that m values more than 100 times these values might be called for ([14]). This in turn, might obsolete my ETEAPOT code as it uses a linearized equation for handling deviations from m=1. Relevant theory is in section 5.2 of [8].

These results, while repeatable/verifiable (or hopefully at least something to have a dialog over) are possibly incorrect. The time of flight per probe, per bend, is critically important. Our current results are a little controversial ([1]). Again, the Munoz Evaluation, goes into a little more detail.

7 TBD

I'm hoping/planning to make this a stand alone, living, document. I feel that it is close now, though oriented for this presentation (EDM collaboration meeting Thursday, Jan 10, 2013 and Friday Jan 11, 2013). As mentioned above, the current iteration of this overview is checked into the Google code repository (`http://code.google.com/hosting/`). Figure 6 portrays my philosophy for my presentation at the EDM collaboration: "a lot of material", several places to start". I guess this is the norm!

I'd like to better document the betatron/transverse capability and cross checking.

Figure 6 quotes transverse as "verified", and longitudinal as "ongoing". While I view this as correct, I should expand!

8 "Verified Transverse"

"Near script" determineTwissCleanRunPlotM1.0_sl4 specifies a command line

./determineTwiss ./data/E_BM_M1.0_sl4.sxf -1 40 >! OUT

This implicitly uses a "split". File extractParameters.h has a parameter splitForBends. This manifests itself on lines 238 -253 of determineTwiss.cc

```
238   if( typeOutput=="Sbend       "){
239 // nonDrifts++;
240    totSplits=2*pow(2,splitForBends);
241 // std::cerr << "totSplits " << totSplits << "\n";
242    sBndDlta=(sX-sPrevious)/totSplits;
243 // sBndDlta=(sX-sPrevious)/(1+splitForBends);
244    for(int j=0;j<totSplits;j++){
245     std::cerr << "name " << nameOutput << " type " << typeOutput << " " << xX << " " << yX << "
246 //  std::cerr << "name " << nameOutput << " type " << typeOutput << " " << xX << " " << yX << "
247     algorithm<double,PAC::Position>::bend_m_elementName[bend]=nameOutput;
248     algorithm<double,PAC::Position>::bend_m_sX[bend++]=sPrevious+sBndDlta;
249     nonDrifts++;
250     sPrevious+=sBndDlta;
251 //  algorithm<double,PAC::Position>::bend_m_sX[bend++]=sX;
252    }
253   }
```

Following determineTwissCleanRunPlotM1.0_sl4 religiously gives the following table:

Table 1: Split Tabulation

splitForBends	Q_x	β_x	Q_y	β_y
0	0.460553	35.9195	0.200880	263.356
1	0.460479	35.9227	0.200401	263.975
2	0.460460	35.9235	0.200281	264.131
3	0.460455	35.9238	0.200251	264.169
4	XXXXXXXX	XXXXXXX	XXXXXXXX	XXXXXXX
4	0.460455	35.9245	0.200244	264.179

The line with Xs is the result of the "out of box" code (8192) with splitForBends = 4. To get the next row, one must modify algorithm.hh to e.g.

```
29     static std::string bend_m_elementName[32768];   // [16384];   // [8192];
30     static double bend_m_sX[32768];   // [16384];   // [8192];
```

and algorithm.icc to e.g.

```
 7   std::string algorithm<Coordinate, Coordinates>::bend_m_elementName[32768];   // [16384];   //
 8 template<class Coordinate, class Coordinates>
 9 double algorithm<Coordinate, Coordinates>::bend_m_sX[32768];   // [16384];   // [8192];
```

These results are "in [12]" (Table 4, page 10), though the exact numbers deviate by e.g. 100 * (35.9195 - 35.8566)/35.9195 $\approx$ 0.2%. This table was generated with bend radius 40, while that one uses bend radius 30. Also, there $Q_x \approx 1.46$ has "the 1 restored", while here $Q_x \approx 0.46$ which is the raw output.

These details have to be controlled by the reader/user of this code.

9 Munoz Evaluation (Time Theory)

TEAPOTs "Drift, Kick, Drift" approach for maintaining symplecticity and other reasons is well documented (see e.g. [6]). For ETEAPOT, this approach has been upgraded to "Bend, Kick, Bend". This upgrade is made necessary by the variation of particle speed in electric fields (unlike magnetic). The lattice element whose continuous evolution is to be modelled as a delta function impulse is assumed known exactly. Here we are trying a refinement where we know the spherical bend element and its kick exactly, and then kick correct that for elements whose propagation is not known exactly (e.g. Cylindrical bend elements). Taking $\mathbf{F} \equiv -k\hat{\mathbf{r}}/r^2$ as our "to be kick corrected force", this gives

$$\mathbf{F} = -k\frac{\hat{\mathbf{r}}}{r^2} = \frac{k}{r^2}\frac{d\hat{\boldsymbol{\theta}}}{d\theta} \tag{1}$$

(using $d\hat{\boldsymbol{\theta}}/d\theta = -\hat{\mathbf{r}}$). This implies planar motion and conserved angular momentum L (per probe, per bend). Choosing polar coordinates (θ, ϕ) in this plane, L = $\gamma m_p r^2 \dot{\theta}$. Note that [8] calls γ, γ^I, to

emphasize that it is inside a bend element. Since I am always inside a bend element here, I use the simpler notation. Defining $\mathbf{u} \equiv \gamma \mathbf{v}$ so that $\mathbf{p} = m_p \mathbf{u}$

$$\frac{d\mathbf{p}}{dt} = \frac{d\mathbf{p}}{d\theta}\frac{d\theta}{dt} = m_p \frac{d\mathbf{u}}{d\theta}\frac{d\theta}{dt} = \frac{\mathrm{L}}{\gamma r^2}\frac{d\mathbf{u}}{d\theta} \tag{2}$$

The classical law of motion $\mathbf{F} = d\mathbf{p}/dt$ holds good including in the semi relativistic region we are working with ([7]). Thus

$$\frac{k}{r^2}\frac{d\hat{\boldsymbol{\theta}}}{d\theta} = \frac{\mathrm{L}}{\gamma r^2}\frac{d\mathbf{u}}{d\theta} \tag{3}$$

so that

$$\frac{d\mathbf{u}}{d\theta} = \frac{k\gamma}{\mathrm{L}}\frac{d\hat{\boldsymbol{\theta}}}{d\theta}. \tag{4}$$

Both the r dependence, and the t dependence have been hidden. Of course the exact time dependence is of critical importance in the phasing of particles relative to the RF Cavity source. This is the main concern, here.

Defining $\mathbf{h} = \mathbf{u} - \frac{k\gamma}{\mathrm{L}}\hat{\boldsymbol{\theta}}$ gives

$$\frac{d\mathbf{h}}{d\theta} = -\frac{k}{\mathrm{L}}\frac{d\gamma}{d\theta}\hat{\boldsymbol{\theta}}. \tag{5}$$

This form motivates the choice of $\mathbf{h}$ as it shows that it is conserved nonrelativistically ($\gamma = 1$, a constant). It turns out that it is conserved relativistically in circular orbits ([4], their Eq. 8). It actually turns out that $\mathbf{h} = 0$ on cicular orbits ([4], their Eq. 13, and see page 1018). This makes it especially good for the near circular design orbits of the EDM experiment. In practice, however, while this sounds good, $\mathbf{h} = 0$ often seems to lead to numerical imprecison due to differences of large numbers giving near zero results.

Then

$$h_\theta = u_\theta - \frac{k\gamma}{\mathrm{L}} = \gamma v_\theta - \frac{k\gamma}{\mathrm{L}} = \gamma r\dot{\theta} - \frac{k\gamma}{\mathrm{L}} = \frac{\mathrm{L}}{m_p r} - \frac{k\gamma}{\mathrm{L}} \tag{6}$$

so that

$$\gamma = \frac{\mathrm{L}^2}{k m_p r} - \frac{\mathrm{L}h_\theta}{k} = \frac{\mathrm{L}}{m_p r^2 \dot{\theta}} = \frac{dt}{d\theta}\frac{\mathrm{L}}{m_p r^2} \tag{7}$$

and

$$\frac{dt}{d\theta} = \frac{\mathrm{L}r}{k} - \frac{m_p r^2}{k}h_\theta = \frac{r}{k}(\mathrm{L} - m_p r h_\theta). \tag{8}$$

This equation is interesting in its own right, and seems to be characteristic of this (Munoz) approach. Calculated quantities often seem to be the difference of 2 terms, each of similar size, though this is not true here. Furthermore, the basic UAL strategy maintains deviations from the design particle for all 6 phase space coordinates ($UAL/codes/PAC/src/PAC/Beam/Position.hh).

Here $UAL is an environmental variable (mentioned in the Computer Syntax for Repeatable Results section). Numerical imprecision becomes a major worry. Richard and I have been wrestling with these for at least a couple of years. We feel that we've handled/overviewed them pretty well.

Our expression for the kepler electric field is

$$\vec{E} = -\hat{\mathbf{r}} E_0 (\frac{r_0}{r})^{1+m} = -\hat{\mathbf{r}} E_0 (\frac{r_0}{r})^2 \tag{9}$$

so that the force is

$$\vec{F} = -\hat{\mathbf{r}} q_0 E_0 r_0^2 (\frac{1}{r})^2. \tag{10}$$

Comparing this to (1) gives

$$k = q_0 E_0 r_0^2 = p_0 v_0 r_0 \tag{11}$$

from the centripetal force $\frac{p_0 v_0}{r_0}$. Now

$$L_0 = r_0 p_0 \tag{12}$$

so that

$$\frac{L_0 r_0}{k} = \frac{r_0}{v_0}. \tag{13}$$

On the (locally) circular design orbit h_θ=0 so

$$\left(\frac{dt}{d\theta}\right)_0 = \frac{L_0 r_0}{k} = \frac{r_0}{v_0} \tag{14}$$

a nice form!

From tracking, we think we know all these quantities (r, L, h_θ). L is constant (per probe, per bend). We think we have exact analytical formulas for r and h_θ right through each bend.

Various equivalent massaged expressions exist ([8] equations 148, 149, and 150).

We have tried a fair cross section of them, and always get quite similar answers. Still the possibility that we are making a mistake somewhere remains. A probably over pedantic list of error sources is:

- Is the Munoz formalism correct for this application?
- Did we calculate this formula correctly?
- Did we enter it into Maple correctly?
- Did Maple give the right answer(s)?
- Did I transcribe the answer(s) correctly?
- Should we have a "kick" correction in time, analogous to the kick correction in momentum?
- Is/are there simple "back of the envelope" approximation(s) that increase confidence?

Investigating some of these may be a future direction for me.

10 Munoz Evaluation (Time Coding)

The above theory is substantially implemented in files
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/getTimeAlternate.inline
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/hamilton.inline
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/reference.inline
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/rotate.insert
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/timeViaExpansion
and probably most critically in
$UAL/codes/ETEAPOT_MltTurn/src/ETEAPOT_MltTurn/Integrator/algorithm.icc.
A code fragment from here is

```
193 double tof0       = Rsxf*th/v0;                                          //  mks
194 double fac1       = EscM* lambda* lambda/ L/ kappa/ kappa/ c/c;           //
195 double A          = fac1;
196 double A_0        = EscM0*lambda0*lambda0/L0/kappa0/kappa0/c/c;          //  ?
197 double fac2       = m_p*k*C_tilda*lambda*lambda/kappa/kappa/L/L/c/c;     //
198 double B          = fac2/C_tilda;
199 // double time1      = getTime1(fac1,th);                      // get_timeFromFirstTermViaMaple
200 // double time2      = getTime2(fac2,th);                      // get_timeFromSecondTermViaMapl
201 // double time1      = t1h(th);
202 // double time2      = t2h(th);
203    double time1      = kappa_time2int(th)/kappa;
204    double time2      = kappa_epsilonSqInt(th)/kappa;
205 double tofA       = time1+time2;
206
207 double dFac1      = epsilon*(-2.*A+B/epsilonTld);
208 double dFac2      = -A*epsilon*epsilon;
209 //     p[4]      -= c*(tofA-tof0);                                       //  ? correct sign cor
210 //     p[4]       =     dFac1*time1+dFac2*time2+(A-A_0)*th;
211        p[4]       -= c*(dFac1*time1+dFac2*time2+(A-A_0)*th);
```

The p[4] update on line 211 is the time component of the 6D phase space.

Note that some of these files are in "ETEAPOT", and some are in "ETEAPOT_MltTurn". This multiple source is one more potential error mechanism.

11 Spin

Page 65 of [8] has a complete theoretical account for implementing the BMT equation. It should be fairly straightforward to implement this practically.

Directory $UAL/examples/THINSPIN has a "version one" of a magnetic spin propagator.

Of course, directory $UAL/examples/SPINK has a mature magnetic spin propagator.

"Near script" runExampleFor_orbitsWithSpin, mentioned in README, has a prescription for exercising the current spin capability. At the moment, this only consists of seeding the spins initial values, and verifying that they can be read into Nikolays foundation code.

File $UAL/examples/ETEAPOT/userBunch mentions

PAC::Bunch bunch(21);
PAC::Spin * spinX = new PAC::Spin[21];

and then provides coordinate values(6) for these 21 particles (probes). Similarly, the spin initial values must be set. The current capability is very rudimentary.

Running

./orbitsWithSpin ./data/E_BM_M1.0_sl4.sxf -1 40 0

seeds the spins as all perfectly polarized in Z [$\forall \text{probes}_i \vec{S}_i = (0, 0, 1)$].

Running

./orbitsWithSpin ./data/E_BM_M1.0_sl4.sxf -1 40 1

seeds the spins randomly. [17] (the GNU Scientific Library) has some of the details. The exact "URL string" there has an extra character \ to get latex to compile.

The default random number generator is gsl_rng_mt19937.

This can be seen from the following program:

```
#include <stdio.h>
#include <gsl/gsl_rng.h>

gsl_rng * r;  /* global generator */

int
main (void)
{
  const gsl_rng_type * T;

  gsl_rng_env_setup();

  T = gsl_rng_default;
  r = gsl_rng_alloc (T);

  printf ("generator type: %s\n", gsl_rng_name (r));
  printf ("seed = %lu\n", gsl_rng_default_seed);
  printf ("first value = %lu\n", gsl_rng_get (r));

  gsl_rng_free (r);
  return 0;
}
```

compiled via

gcc random.c -o random -lgsl -lgslcblas

and run with

./random

giving

generator type: mt19937
seed = 0
first value = 4293858116

This can be configured with environmental variables. Thus, in the "tee-shell" (tcsh):

tcsh

```
    setenv GSL_RNG_TYPE taus
    setenv GSL_RNG_SEED 123
gives
    ./random
        GSL_RNG_TYPE=taus
        GSL_RNG_SEED=123
        generator type: taus
        seed = 123
        first value = 2720986350
```

12 References

References

[1] Semertzidis, Y. Personal Communication

[2] Talman, R. Personal Communication

[3] Talman, R. Reduced Dispersion Proton EDM Storage Ring Lattices December 10, 2012

[4] Munoz, G., Pavic, I. A Hamilton-like vector for the special-relativistic Coulomb problem European Journal of Physics 27(2006) 1007-1018

[5] Boyer, T. Unfamiliar trajectories for a relativistic particle in a Kepler or Coulomb potential American Journal of Physics 72(2004) 992-997

[6] TEAPOT: A Thin-Element Accelerator Program For Optics And Tracking Schachinger, L. and Talman, R. Particle Accelerators 1987, Vol. 22, pp. 35 - 56

[7] Talman, R. Geometric Mechanics Page 360 John Wiley & Sons, Inc. 2000

[8] Malitsky, N., Talman, J., and Talman, R. Appendix UALcode: Development of the UAL/ETEAPOT Code for the Proton EDM Experiment Personal Communication December 15, 2012

[9] Talman, R. All-Electric Proton EDM Lattices for Benchmarking Simulation Codes Mar. 12, 2012

[10] Talman, J. and Talman, R. UAL/ETEAPOT Results for Proton EDM Benchmark Lattices April 27, 2012

[11] Talman, J. and Talman, R. UAL/ETEAPOT Proton EDM Benchmark Comparisons II: Transfer Matrices and Twiss Functions August 30, 2012

[12] Talman, J. and Talman, R. UAL/ETEAPOT Results (Augmented) for Proton EDM Benchmark Lattices October 29, 2012

[13] Talman, J. and Talman, R. UAL/ETEAPOT Proton EDM Benchmark Comparisons III: Dispersion, Longitudinal Dynamics and Synchrotron Oscillations January 10, 2013

[14] Talman, R. Reduced Dispersion Proton EDM Storage Ring Lattices December 10, 2012

[15] Peraire, J. and Widnall, S. Lecture L15 - Central Force Motion: Keplers Laws Fall 2008

[16] Schechter, P. `http://ocw.mit.edu/courses/physics/8-902-astrophysics-ii-fall-2004/lecture-notes/lec6.pdf` Fall 2001

[17] `http://www.gnu.org/software/gsl/manual/html\_node/Random-number-environment-variables.html` 2011

[18] Talman, J. JohnTalmanStatus_2_3_2011.pdf February 3, 2011

[19] Talman, J. Jan11_2012_Report.pdf January 8, 2012

[20] Talman, J. matrixInterface.txt December 10, 2011

[21] Talman, J. portDaIntegrator.pdf January 22, 2012

[22] Talman, J. portSphericalIntoGeneralETEAPOT.pdf February 24, 2012

[23] Talman, J. Porting Front-End ($UAL/examples) Code Into the Back-End (($UAL/codes): Constrained First Look at the Existing ETEAPOT Facility. February 24, 2012

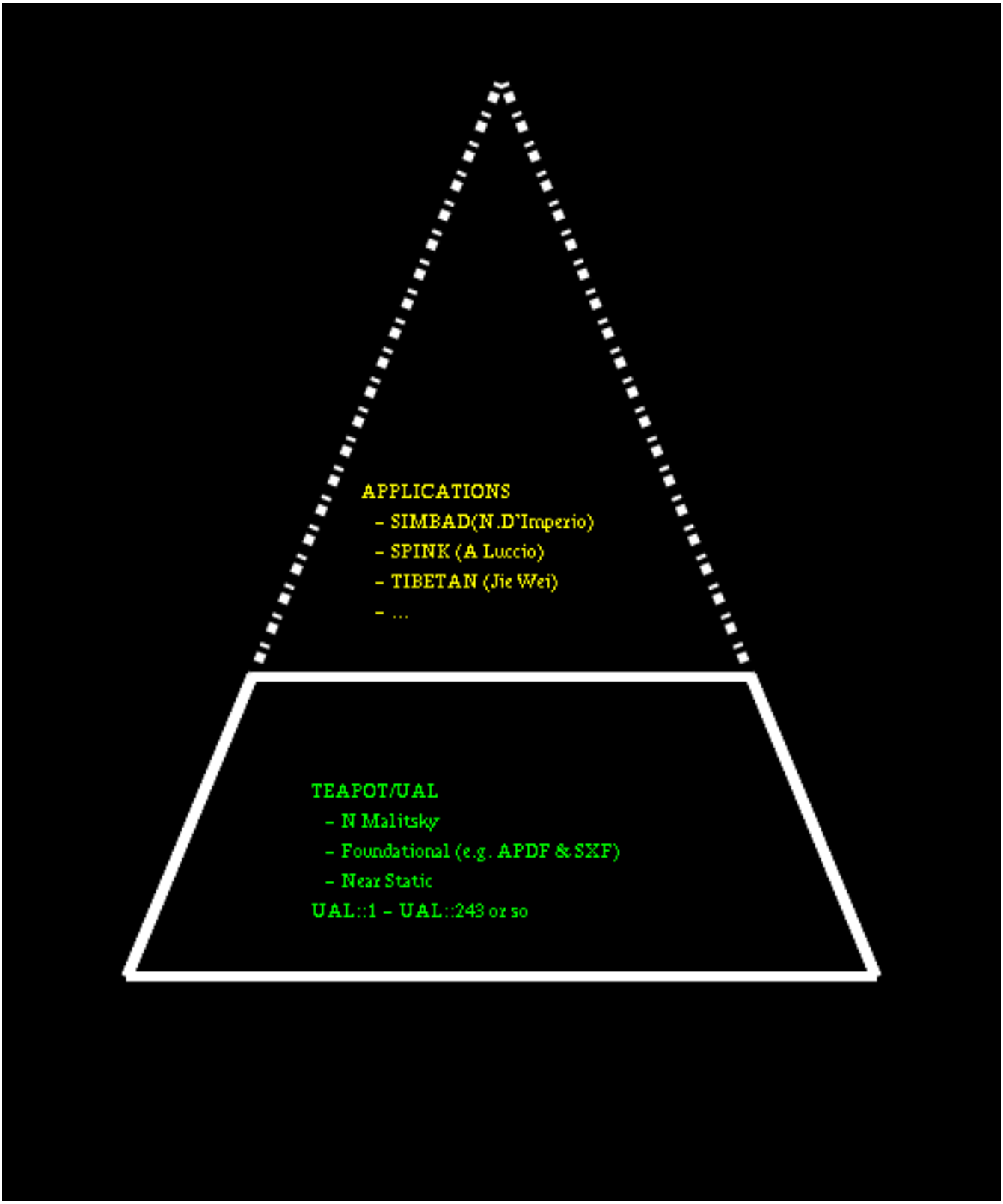

Figure 1: The "UAL Project View" (highly iconic representation of the UAL Architecture, and some successful prior projects)

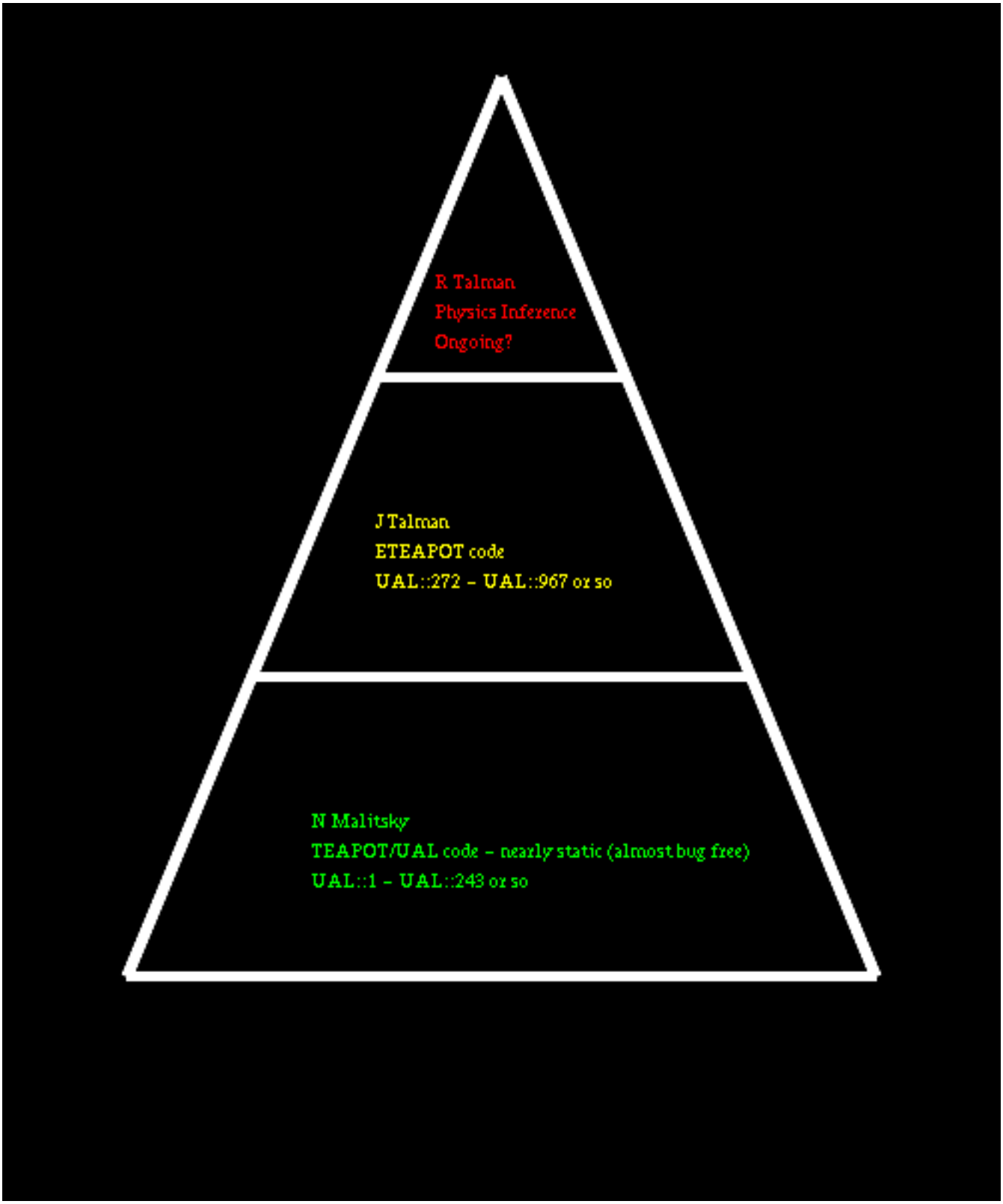

Figure 2: Johns ETEAPOT Project view of the UAL world. I've enabled/facilitated electric bend physics simulations for physicists, and I'm done! ;-)

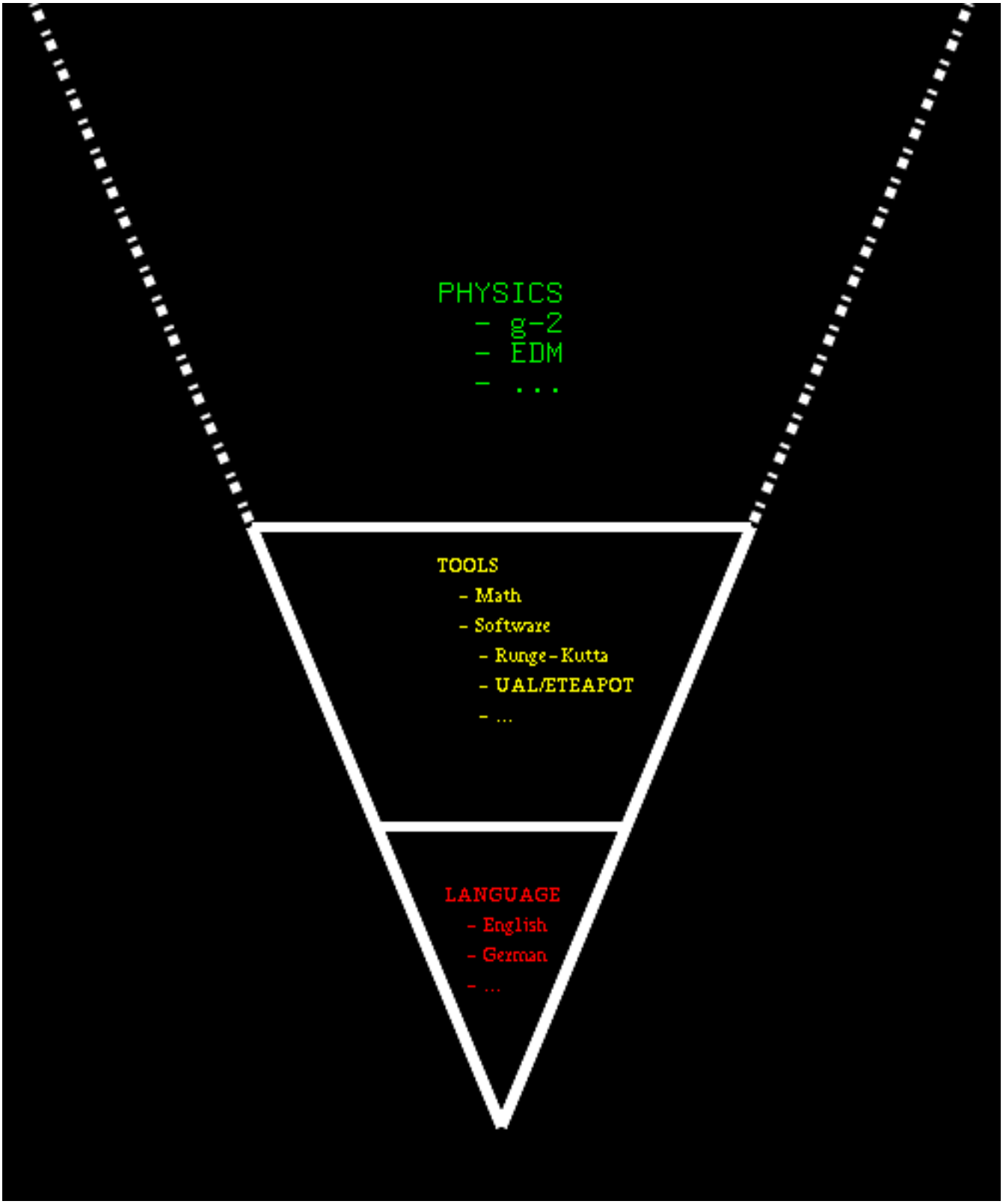

Figure 3: Johns Toolbox view of the physics world. I'm trying to add UAL/ETEAPOT to the tool box!

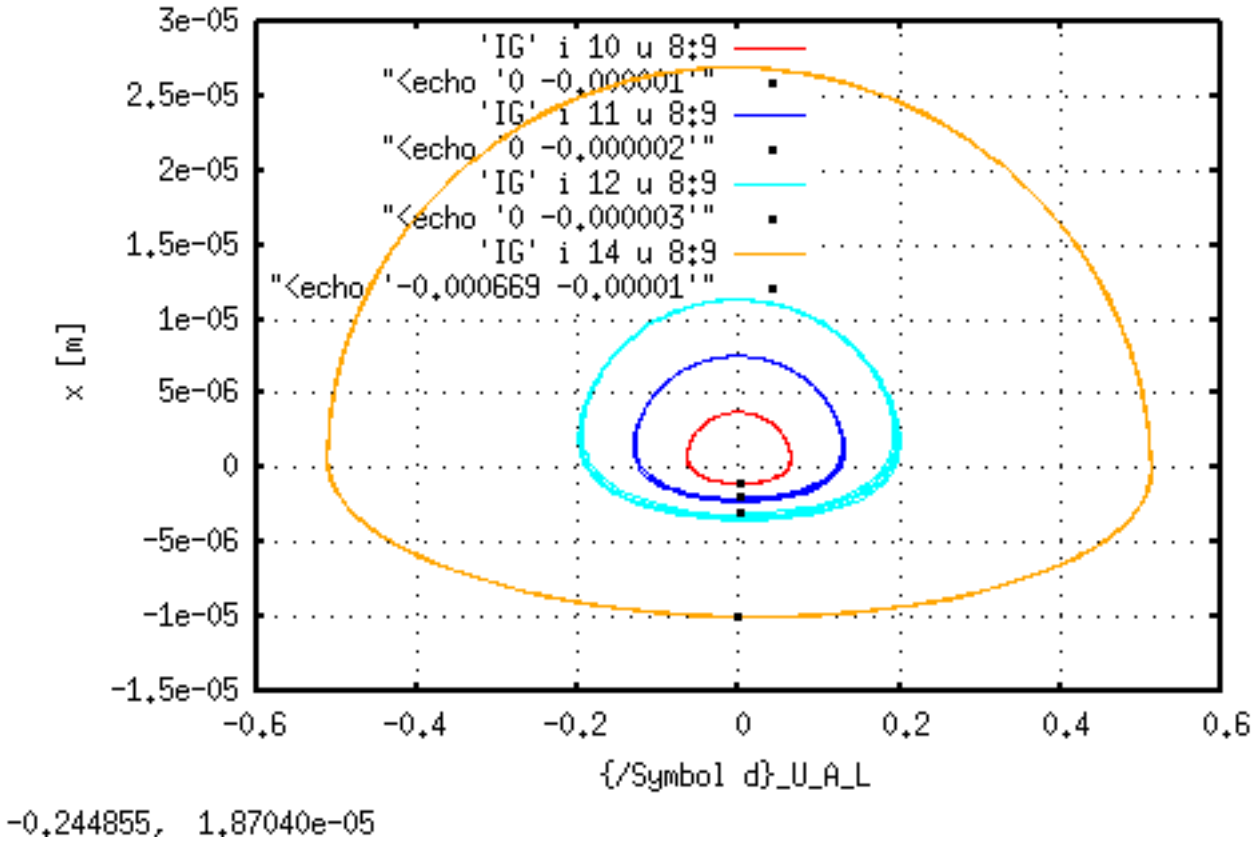

Figure 4: Duplication of the Figure 2 from [13]

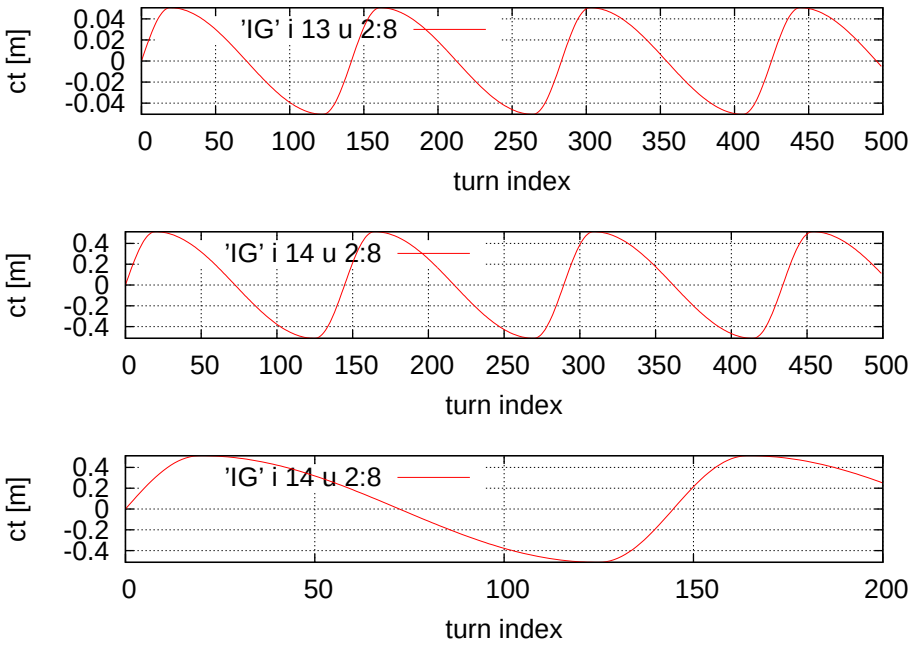

Figure 5: Duplication of the Figure 11 from [13]

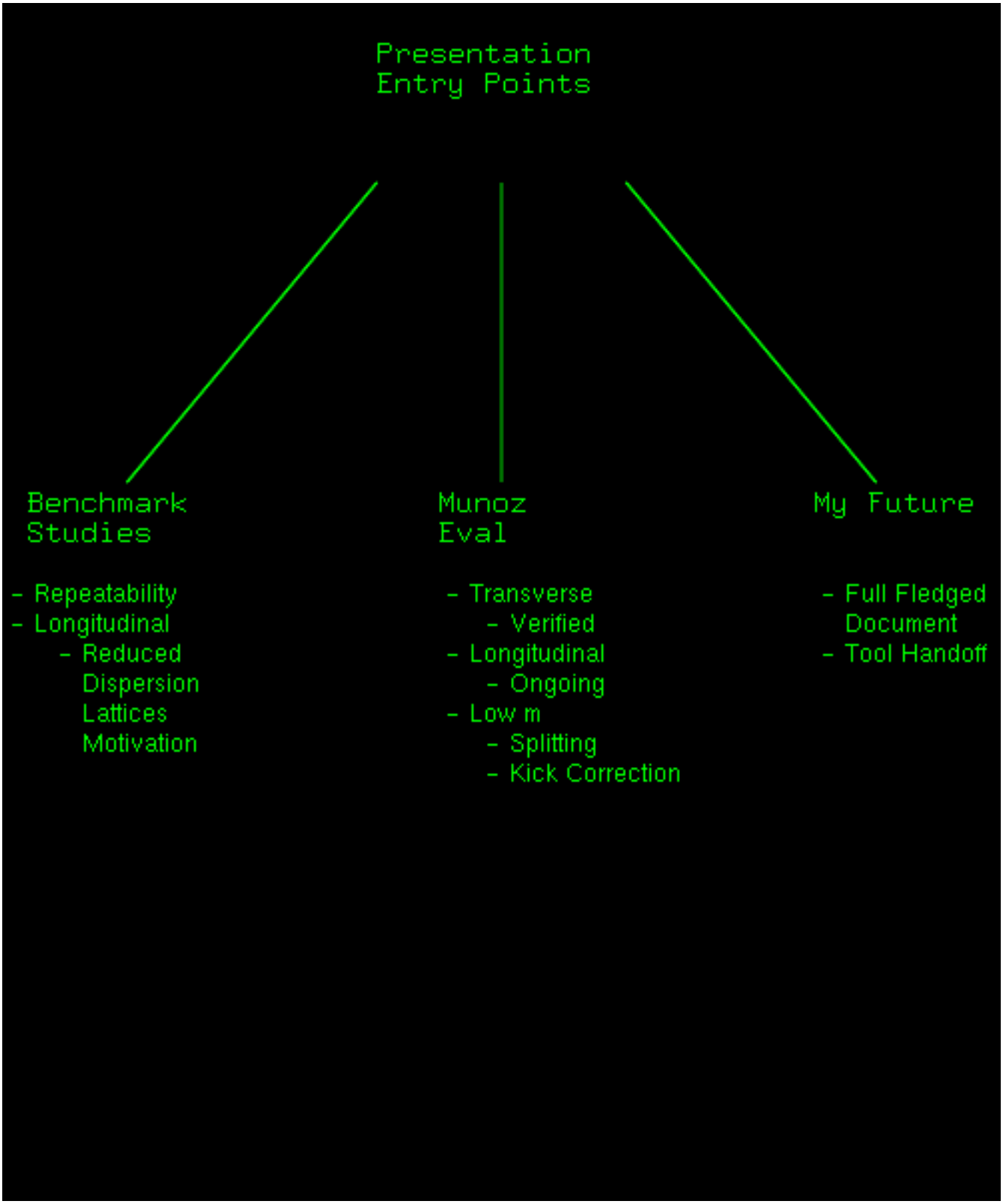

Figure 6: My philosophy for my presentation at the EDM collaboration.