Porting Front-End ($UAL/examples) Code Into the Back-End (($UAL/codes): Constrained First Look at the Existing ETEAPOT Facility.

J. Talman

February 24, 2012

Abstract

The ”UAL/TEAPOT code” is very powerful accelerator simulation code. It is available on the web and is mostly geared for magnetic lattice elements. Possibly its greatest strength is its APDF (Accelerator Propagator Description Format) mechanism that allows flexible linking of algorithms with lattice elements. Several powerful examples exist, and it would be useful to be able to create new ones. Also important is code organization, reorganization, and refactoring. Electric lattices, elements, and capability, are particularly germane. This is looked at here in the context of the existing ”electric lattice code” ($UAL/examples/ETEAPOT/..., $UAL/codes/ETEAPOT/...). TEAPOT stands for Thin Element Accelerator Program for Optics and Tracking. ETEAPOT is Electric TEAPOT. While these ETEAPOT directories are intended to eventually be fully electric, they have substantial magnetic legacy. They also have possibly mismatched code locations. Specifically, some code that probably belongs in the back-end is currently in the front-end (it started out exploratory, and tentative). Presumably there will eventually be several ”electric threads” representing various approaches to electric lattices. For the time being they are expected to be in $UAL/codes/ETEAPOT/... This raises naming and disambiguation issues in the apdf file (among other things). This report is part of this ongoing effort. A first ”benchmarking” comparing the existing code and its output with its port into the back-end is presented in the Results section. This is all hoped to eventually facilitate full-fledged physics models comparisons, and benchmarking.

1 Introduction

UAL/TEAPOT has a front-end, back-end, architecture/orientation. Thus the code in $UAL/examples/... typically allows the user to concentrate on the results of particular parameters of a physical simulation (”front-end”). The code in $UAL/codes/... typically purports to correctly, or effectively, model these inputs (”back-end”). In practice, this demarcation is often not completely upheld. Also the code is not always fully debugged, physicswise. This C++ code and directory structure is very complex, and a good first step is getting the new code to compile, run, and give sensible results.

The main usage of ETEAPOT is expected to eventually follow this organization closely. Thus

```
(front-end) \$UAL/examples/ETEAPOT/tracker.cc
```

uses the classes in

```
(back-end) $UAL/codes/ETEAPOT/...
```

for its simulation results. This is the conventional organization. These classes are a good first step, but are substantially magnetic clones, however.

$UAL/examples/Spherical started out as a tentative ”standalone” electric application based substantially on Munoz [1]. Thus many of the simulation classes it uses are in the same front-end directory ($UAL/examples/Spherical). Again, this doesn’t follow the conventional UAL/TEAPOT architecture.

Migrating these specific classes into their more conventional back-end location and the associated new general directory structure are the main subject of this report.

2 APDF

The syntax for running different front-end programs tends to be similar:

```
./tracker ./data/XXX.sxf p1 p2 ... (>&! myOut)
```

where XXX.sxf represents the accelerator lattice and p1, p2, ... are additional parameters. The Appendix has the exact run syntax used in this report.

This may change, but the (implicit) apdf file used is encoded somewhere off the command line. The Results section has the exact apdf file, and procedure used in this report.

Confusingly, an apdf file used by $UAL/examples/ETEAPOT

```
<apdf>
  <propagator id="ETEAPOT" accelerator="ring">
     <create>
       <link algorithm="ETEAPOT::DriftTracker" types="Default"/>
       <link algorithm="ETEAPOT::DipoleTracker" types="Sbend"/>
       <link algorithm="ETEAPOT::MltTracker" types="Quadrupole|Sextupole"/>
       <link algorithm="ETEAPOT::RFCavityTracker" types="RfCavity"/>
     </create>
  </propagator>
</apdf>
</apdf>
```

uses a different dipole tracking algorithm (ETEAPOT::DipoleTracker) than that used by $UAL/examples/Spherical even though the apdf files are the same:

```
http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/data/eteapotTRACK.apdf
http://code.google.com/p/ual/source/browse/trunk/examples/Spherical/data/eteapot.apdf
```

This is due to $UAL/examples/Spherical using the "local" ETEAPOT::DipoleTracker class, and a "customized" build process (Makefile)

```
http://code.google.com/p/ual/source/browse/trunk/examples/Spherical/Makefile
```

Disambiguating these two physical classes sharing the same name is one of the first tasks looked at here. I rename

```
ETEAPOT::DipoleTracker --->>> ETEAPOT::bendTracker
```

here, though this may change. I look at a tentative directory structure in the next section, and ultimately the class names should probably reflect this. Also, the "...Tracker" is probably superfluous/redundant.

The Discussion on How to Proceed section goes into this in more detail.

3 Directory Structure

Historically, the "main tracking" physics is in the directory .../Integrator e.g.

```
http://code.google.com/p/ual/source/browse/trunk/codes/TEAPOT/src/TEAPOT/Integrator
```

and the current "first port" for ETEAPOT follows this

```
http://code.google.com/p/ual/source/browse/trunk/codes/ETEAPOT/src/ETEAPOT/Integrator
```

With an eye on various electric threads, starting with the $UAL/examples/Spherical migration, I propose to create a directory named something like

```
$UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend
```

which hasn't been "checked in" to google codes yet. This is "m = 1" in our terminology (Talman [2]). This leaves room for a peer directory such as "oneOver_rBend" to contain code that emphasizes ("cylindrical") m = 0

An electric field with index m power law dependence on radius r for y=0 is

$$\mathbf{E}(r,0) = -E_0 \frac{r_0^{1+m}}{r^{1+m}} \hat{\mathbf{r}}, \tag{1}$$

and the electric potential $V(r)$, adjusted to vanish at $r = r_0$, is

$$V(r) = -\frac{E_0 r_0}{m} \left(\frac{r_0^m}{r^m} - 1 \right). \tag{2}$$

There is current m = 0 code in

```
$UAL/examples/Cylindrical
```

though this thread is not actively being developed.

With an eye on the specific approach to the (m = 1)/Spherical/Kepler/Coulomb physics of Munoz and Pavic [1], I propose to create a directory named something like

```
$UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend/MunozPavic
```

Finally, with an eye on a first ”benchmark” I propose to create a directory named something like

```
$UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend/MunozPavic/legacyBenchmark
```

to contain the necessary classes in the $UAL/examples/Spherical migration. This code will eventually be contrasted, and compared with the results/output of the code in e.g.

```
$UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend/MunozPavic/conservedVector
```

These names are all subject to critique and change, though I have a working prototype with them.

Again, the Discussion on How to Proceed section goes into this in more detail.

4 Build Structure

The build process in UAL tends to be hierarchical, with the Makefiles at the leaves (where the class code is) doing the compiling, and the other Makefiles sequencing the appropriate subdirectories (”recursively”). The actual compile line executed in the leaves typically resides in .../src/Makefile.config (line 10)

```
 1 include $(UAL)/env/$(UAL_ARCH)/Makefile.config
 2
 3 INC      = -I$(UAL_ETEAPOT)/src -I$(UAL_PAC)/src  -I$(UAL_CORE)/include -I$(UAL_ZL    IB)/src
 4 LIBS     = -L$(UAL_ETEAPOT)/lib/$(UAL_ARCH) -L$(UAL_PAC)/lib/$(UAL_ARCH) \
 5                 -L$(UAL_ZLIB)/lib/$(UAL_ARCH) \
 6                 -L$(UAL_CORE)/lib/$(UAL_ARCH) -lUal -ldl
 7 LIBS += -L$(UAL)/tools/lib/$(UAL_ARCH)          -lgsl -lgslcblas
 8
 9 $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj/%.o : %.cc
10         $(CC) $(CCFLAGS) $(INC) -c $< -o $@;
11
```

which gets included (line 1) via .../src/Makefile prior to the recursion

```
 1 include $(UAL_ETEAPOT)/src/Makefile.config
 2
 3 DIRS =  ./ETEAPOT
 4
 5 LIBS += -L$(UAL_PAC)/lib/$(UAL_ARCH) -lPacSMF
 6 LIBS += -L$(UAL_PAC)/lib/$(UAL_ARCH) -lPacOptics
 7 LIBS += -L$(UAL_PAC)/lib/$(UAL_ARCH) -lPac
 8 LIBS += -L$(UAL_PAC)/lib/$(UAL_ARCH) -lPacSurvey
 9 LIBS += -L$(UAL_ZLIB)/lib/$(UAL_ARCH) -lZTps
10 LIBS += -L$(UAL)/tools/lib/$(UAL_ARCH)          -lgsl -lgslcblas
11
12 include ./ETEAPOT/Integrator/Objects
13 include ./ETEAPOT/inverseSquareBend/MunozPavic/legacyBenchmark/Objects
14
15 compile :
16         if [ ! -d $(UAL_ETEAPOT)/include ] ; \
17                 then (mkdir $(UAL_ETEAPOT)/include;); fi;
18         (if [ !  -d $(UAL_ETEAPOT)/lib ]; \
19                 then mkdir  $(UAL_ETEAPOT)/lib; fi;)
20         (if [ !  -d $(UAL_ETEAPOT)/lib/$(UAL_ARCH) ]; \
21                 then mkdir  $(UAL_ETEAPOT)/lib/$(UAL_ARCH); fi;)
22         (if [ !  -d $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj ]; \
23                 then mkdir  $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj; fi;)
24         @for dir in $(DIRS); do \
25                 (cd $$dir; if [ -f ./Makefile ]; then $(MAKE) compile; fi;); \
26         done
27         $(LD) $(LDFLAGS) $(INC) -o $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj/dummy   \
28                 ./dummy.cc $(OBJS) $(LIBS)
29         cp /dev/null $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj/dummy
30         $(DLD) $(DLDFLAGS) -o $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/libETeapot.so $(OBJS)     $(LIBS)
31 ...
```

This builds (line 30) the actual library ($UAL_ETEAPOT/lib/$UAL_ARCH/libETeapot.so) used by the front-end.

Thus directory

```
$UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend
```

gets Makefile

```
include $(UAL)/env/$(UAL_ARCH)/Makefile.config

DIRS = ./MunozPavic

compile:
        @for dir in $(DIRS); do \
                (cd $$dir; if [ -f ./Makefile ]; then $(MAKE) compile; fi;); \
        done

clean:
        @for dir in $(DIRS) ; do \
                (cd $$dir; if [ -f ./Makefile ]; then $(MAKE) clean; fi;); \
        done
        rm -rf ./lib/$(UAL_ARCH)
```

and directory MunozPavic gets the very similar

```
include $(UAL)/env/$(UAL_ARCH)/Makefile.config

DIRS = ./legacyBenchmark ./conservedVector

compile:
        @for dir in $(DIRS); do \
                (cd $$dir; if [ -f ./Makefile ]; then $(MAKE) compile; fi;); \
        done

clean:
        @for dir in $(DIRS) ; do \
                (cd $$dir; if [ -f ./Makefile ]; then $(MAKE) clean; fi;); \
        done
        rm -rf ./lib/$(UAL_ARCH)
```

which differs only in the subdirectories specified (here legacyBenchmark, and conservedVector). The leaf (src/ETEAPOT/inverseSquareBend/MunozPavic/legacyBenchmark/Makefile) is a little different

```
include $(UAL_ETEAPOT)/src/Makefile.config

INC += -I$(UAL_PAC)/include/
LIBS += -L$(UAL_PAC)/lib/$(UAL_ARCH) -lPac

include ./Objects

compile : $(OBJS)
        if [ ! -d $(UAL_ETEAPOT)/include/ETEAPOT/inverseSquareBend ] ; \
                then mkdir -p $(UAL_ETEAPOT)/include/ETEAPOT/inverseSquareBend ; fi;
        cp *.hh *.icc $(UAL_ETEAPOT)/include/ETEAPOT/inverseSquareBend

clean:
        rm -f $(OBJS)
        rm -rf $(UAL_ETEAPOT)/include/ETEAPOT/inverseSquareBend
```

though it mainly itemizes the appropriate files in

```
src/ETEAPOT/inverseSquareBend/MunozPavic/legacyBenchmark/Objects
```

which currently is only

```
OBJS += $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj/bendTracker.o
```

There are several "include files", however:

```
ll $UAL/codes/ETEAPOT/src/ETEAPOT/inverseSquareBend/MunozPavic/legacyBenchmark
   bendTracker.cc
   bendTracker.hh
   getDesignBeam.h
   getTimeAlternate.inline
   hamilton.inline
   Makefile
   newDipoleAlgorithm.hh
   newDipoleAlgorithm.icc
   Objects
   printDesignBeam.h
   printPropagateInfo.h
   reference.inline
   rotate.insert
   verboseBlock.h
```

5 Results

For the Spherical tracker, the apdf file is encoded via file

```
   extractParameters.h
      15  std::string apdfFile = "./data/eteapot.apdf";
```

Making the change

```
   extractParameters.h
      15  std::string apdfFile = "./data/eteapotLegacyBenchmark.apdf";
```

allows the new ETEAPOT::bendTracker algorithm to be used via apdf file

```
<apdf>
  <propagator id="ETEAPOT" accelerator="ring">
     <create>
       <link algorithm="ETEAPOT::DriftTracker" types="Default"/>
       <link algorithm="ETEAPOT::bendTracker" types="Sbend"/>
       <link algorithm="ETEAPOT::MltTracker" types="Quadrupole|Sextupole"/>
       <link algorithm="ETEAPOT::RFCavityTracker" types="RfCavity"/>
     </create>
  </propagator>
</apdf>
```

Specific probe parameters are on lines 4 - 10 in file

```
   simulatedProbeValues
      1 double k       = IA*p0*v0;                        //
      2
      3 //                   probe deviations
      4 double  dx     = 0.01;
```

```
double  dy     = 0.0;
double  dz     = 0.0;

double dpx     = 0.0;
double dpy     = 0.0;
double dpz     = 0.0;

double dt      = 0.0;
//                   probe deviations

double rin     = IA+dx;                            //
double rinEx   = sqrt((IA+dx)*(IA+dx)+dy*dy+dz*dz);//
  double gamma = (k/rinEx-k/IA)/m0/c/c;             //
         gamma = gamma+gamma0;
//double gamma   = gamma0;                         //
double Ein     = gamma*m0*c*c-k/rinEx;             // E for compatibility with Munoz

double Edes    = m0*c*c/gamma0;                    // Design Energy (Munoz potential)
double dE      = Ein-Edes;

double dpxbyp0 = dpx/p0;
double dpybyp0 = dpy/p0;
double dEbyp0  = dE /p0;

PAC::Bunch bunch(1);                               // bunch with 1 particle(s)
bunch.setBeamAttributes(ba);
bunch[0].getPosition().set(dx,dpxbyp0,dy,dpybyp0,dt,dEbyp0);

#include "printProbeValues"
```

and show that the example run here has an initial horizontal offset of 1cm (0.01) from the design orbit with all other deviations 0.

When the syntax in the Appendix is followed Fig. 1 is produced. A close look at file .01_0_0_0_0_0.orbit shows that it implies a horizontal tune, Q_x, of

$$Q_x = (8\,\text{oscillations})/(1598\,\text{split bends}) * (160\,\text{split bends})/\text{turn} = 0.801\text{oscillations}/\text{turn} \quad (3)$$

This corroborates the familiar result derived purely via the front-end. I regard it as implying a successful port validation into the back-end.

6 Discussion on How to Proceed

Though I use it in this report, I don't really like the name "bendTracker"!

I'm tempted to call it something more like "m_1_MP_L" for

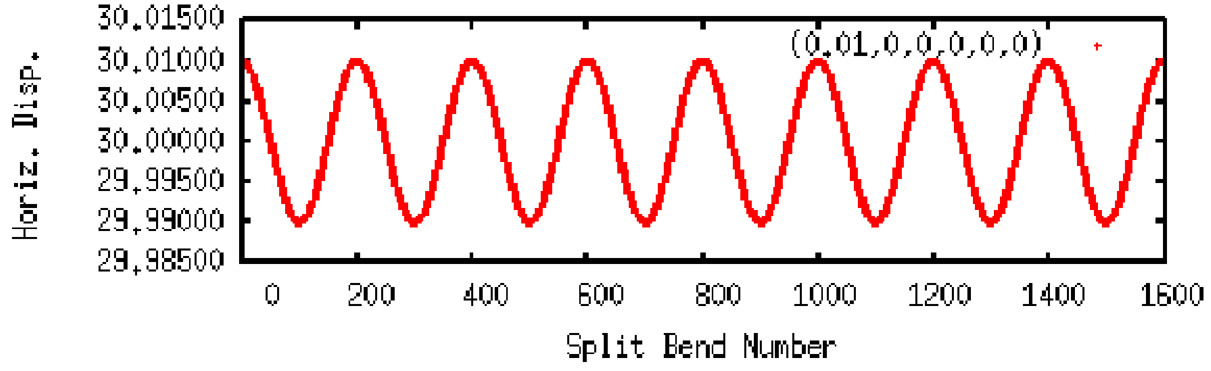

Figure 1: E_Kepler.sxf: Split=1. Horizontal displacement $Q_x \approx 0.801$

```
m = 1,
Munoz Pavic Hamilton Vector Theory,
Legacy (for port test)
```

Then the latest approach, "Conserved Vector" ([3], [4]), could be named "m_1_MP_CV" for

```
m = 1,
Munoz Pavic Hamilton Vector Theory,
Conserved Vector
```

This isn't just academic, as there has already been a name collision between (front-end) ETEAPOT::DipoleTracker and (back-end) ETEAPOT::DipoleTracker. As the models proliferate, and the benchmarking becomes serious, and disseminated, my sense is it will be worth it to be systematic wherever possible.

7 Appendix

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

cd examples/Spherical
make clean
make
./tracker ./data/E_Kepler.sxf 30 -1.3 >&! .01_0_0_0_0_0
cat .01_0_0_0_0_0 | grep rpOut > ! .01_0_0_0_0_0.orbit
```

```
gnuplot
set terminal x11 size 478,145
set xlabel "Split Bend Number"
set format x "%5.0f"
set ylabel "Horiz. Disp."
set format y "%7.5f"
set tmargin 1
p ".01_0_0_0_0_0.orbit" u 4 title "(0.01,0,0,0,0,0)"

xwd>.01_0_0_0_0_0.orbit.xwd
convert .01_0_0_0_0_0.orbit.xwd .01_0_0_0_0_0.orbit.eps
```

References

[1] G. Muñoz and I. Pavic, *A Hamilton-like vector for the special-relativistic Coulomb problem,* Eur. J. Phys. **27**, 1007-1018, 2006

[2] N. Malitsky, J. Talman, and R. Talman, *Appendix UALcode: Development of the UAL/ETEAPOT Code for the Protion EDM Experiment,* Feb 9, 2012 and Personal Communication

[3] R. Talman and J. Talman, *Juelich Lecture Notes: 4. UAL, ETEAPOT and Long Term Symplectic Tracking for Electric Lattices,* Personal Communication

[4] N. Malitsky and R. Talman, *Unified Accelerator Libraries,* Proceedings of Conference on Accelerator Software, Williamsburg, VA, 1996, *The Framework of Unified Accelerator Libraries,* ICAP98, Monterey, 1998