Porting UAL (Unified Accelerator Libraries) Lattice Element Propagators - Constrained First Look

J. Talman

January 22, 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. Electric lattice, elements, and capability, is 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. This report is part of this ongoing effort.

1 Introduction

The UAL code is available on the web. The Appendix briefly expands on this. The specific "example" looked at here uses

$UAL/examples/ETEAPOT/evolver.cc ("evolver")

and

$UAL/codes/ETEAPOT/...

These are each discussed in the eponymous next two sections. The main effort is in the subsequent sections.

2 $UAL/examples/ETEAPOT/evolver.cc

Evolver uses the existing apdf file ($UAL/examples/ETEAPOT/data/evolver.apdf)

```
<apdf>
  <propagator id="Evolver" accelerator="ring">
     <create>
       <link algorithm="TEAPOT::DriftDaIntegrator" types="Default"/>
       <link algorithm="TEAPOT::DriftDaIntegrator" types="Drift"/>
       <link algorithm="TEAPOT::DipoleDaIntegrator" types="Sbend"/>
       <link algorithm="TEAPOT::MltDaIntegrator" types="Quadrupole"/>
       <link algorithm="TEAPOT::MltDaIntegrator" types="Sextupole"/>
     </create>
  </propagator>
</apdf>
```

These ”algorithms” reside in

```
$UAL/codes/TEAPOT/src/TEAPOT/Integrator/DriftDaIntegrator.cc
$UAL/codes/TEAPOT/src/TEAPOT/Integrator/DipoleDaIntegrator.cc
$UAL/codes/TEAPOT/src/TEAPOT/Integrator/MltDaIntegrator.cc
```

and the corresponding header (.hh) files. The main content of this report documents how I create file

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleDaIntegrator.cc
```

so that evolver can use it via the apdf file

```
<apdf>
  <propagator id="Evolver" accelerator="ring">
     <create>
       <link algorithm="TEAPOT::DriftDaIntegrator" types="Default"/>
       <link algorithm="TEAPOT::DriftDaIntegrator" types="Drift"/>
       <link algorithm="ETEAPOT::DipoleDaIntegrator" types="Sbend"/>
       <link algorithm="TEAPOT::MltDaIntegrator" types="Quadrupole"/>
       <link algorithm="TEAPOT::MltDaIntegrator" types="Sextupole"/>
     </create>
  </propagator>
</apdf>
```

Example syntax/usage for evolver is

```
./evolver ./data/pre-E_pEDm.sxf
```

and it currently reports that it is using the TEAPOT DipoleDaIntegrator e.g.

```
Element-by-element propagation.
0 TEAPOT::DriftDaIntegrator
1 TEAPOT::DriftDaIntegrator
2 TEAPOT::MltDaIntegrator
3 TEAPOT::MltDaIntegrator
4 TEAPOT::DriftDaIntegrator
```

```
5 TEAPOT::MltDaIntegrator
6 TEAPOT::MltDaIntegrator
7 TEAPOT::DipoleDaIntegrator
8 TEAPOT::MltDaIntegrator
9 TEAPOT::MltDaIntegrator
10 TEAPOT::MltDaIntegrator
11 TEAPOT::MltDaIntegrator
12 TEAPOT::DipoleDaIntegrator
13 TEAPOT::MltDaIntegrator
14 ...
```

3 $UAL/codes/ETEAPOT/...

The first step is to "clone" the existing algorithm

```
cp $UAL/codes/TEAPOT/src/TEAPOT/Integrator/DipoleDaIntegrator.hh
   $UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleDaIntegrator.hh

cp $UAL/codes/TEAPOT/src/TEAPOT/Integrator/DipoleDaIntegrator.cc
   $UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleDaIntegrator.cc
```

and the second step is to edit these files.
Ignoring the file lineage stanza, the first change in

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleDaIntegrator.hh is the "Include Guard"

     #ifndef UAL_TEAPOT_DIPOLE_DA_INTEGRATOR_HH
          ----->>>>>
     #ifndef UAL_ETEAPOT_DIPOLE_DA_INTEGRATOR_HH
```

The second change is the namespace

```
namespace TEAPOT {
     ----->>>>>
namespace ETEAPOT {
```

The third change is to explicitly tell the compiler where to find the base class

```
class DipoleDaIntegrator : public BasicDaIntegrator {
     ----->>>>>
class DipoleDaIntegrator : public TEAPOT::BasicDaIntegrator {
```

as I have not ported class BasicDaIntegrator into ETEAPOT (in this context). Thus the default symbol "location" has gone from TEAPOT to ETEAPOT. Continuing

```
DipoleData m_data;
     ----->>>>>
TEAPOT::DipoleData m_data;
```

```
MagnetData m_mdata;
     ----->>>>>
TEAPOT::MagnetData m_mdata;

static DipoleAlgorithm<ZLIB::Tps, ZLIB::VTps> s_algorithm;
     ----->>>>>
static TEAPOT::DipoleAlgorithm<ZLIB::Tps, ZLIB::VTps> s_algorithm;
```

Ignoring the file lineage stanza, the first change in

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleDaIntegrator.cc is to give it the right header

     #include "TEAPOT/Integrator/DipoleDaIntegrator.hh"
          ----->>>>>
     #include "ETEAPOT/Integrator/DipoleDaIntegrator.hh"
```

Continuing

```
TEAPOT::DipoleAlgorithm<ZLIB::Tps, ZLIB::VTps> TEAPOT::DipoleDaIntegrator::s_algorithm;
     ----->>>>>
TEAPOT::DipoleAlgorithm<ZLIB::Tps, ZLIB::VTps> ETEAPOT::DipoleDaIntegrator::s_algorithm;

const char* TEAPOT::DipoleDaIntegrator::getType() {
     return "TEAPOT::DipoleAlgorithm";
          ----->>>>>
const char* ETEAPOT::DipoleDaIntegrator::getType() {
     return "ETEAPOT::DipoleAlgorithm";

TEAPOT::DipoleDaIntegrator::DipoleDaIntegrator()
     ----->>>>>
ETEAPOT::DipoleDaIntegrator::DipoleDaIntegrator()

TEAPOT::DipoleDaIntegrator::DipoleDaIntegrator(const TEAPOT::DipoleDaIntegrator& dt)
     ----->>>>>
ETEAPOT::DipoleDaIntegrator::DipoleDaIntegrator(const ETEAPOT::DipoleDaIntegrator& dt)

TEAPOT::DipoleDaIntegrator::~DipoleDaIntegrator()
     ----->>>>>
ETEAPOT::DipoleDaIntegrator::~DipoleDaIntegrator()

UAL::PropagatorNode* TEAPOT::DipoleDaIntegrator::clone()
     ----->>>>>
UAL::PropagatorNode* ETEAPOT::DipoleDaIntegrator::clone()

return new TEAPOT::DipoleDaIntegrator(*this);
     ----->>>>>
return new ETEAPOT::DipoleDaIntegrator(*this);
```

```
void TEAPOT::DipoleDaIntegrator::setLatticeElements(const UAL::AcceleratorNode& sequence,
     ----->>>>>
void ETEAPOT::DipoleDaIntegrator::setLatticeElements(const UAL::AcceleratorNode& sequence,

void TEAPOT::DipoleDaIntegrator::setLatticeElement(const PacLattElement& e)
     ----->>>>>
void ETEAPOT::DipoleDaIntegrator::setLatticeElement(const PacLattElement& e)

void TEAPOT::DipoleDaIntegrator::propagate(UAL::Probe& probe)
     ----->>>>>
void ETEAPOT::DipoleDaIntegrator::propagate(UAL::Probe& probe)
```

This "new algorithm" will compile, and be available, for evolver with the changes enumerated in the next section.

4 Main Compilation Changes

File

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/Objects
```

must be told about this new class/algorithm.

```
12 $(UAL_ETEAPOT)/lib/$(UAL_ARCH)/obj/DipoleDaIntegrator.o \
```

File

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/Makefile
```

must be able to access the inherited TEAPOT functionality.

```
4 INC += -I$(UAL_TEAPOT)/include/
5 ...
6 LIBS += -L$(UAL_TEAPOT)/lib/$(UAL_ARCH) -lTeapot
```

Ditto for file

```
$UAL/codes/ETEAPOT/src/Makefile

     11 LIBS += -L$(UAL_TEAPOT)/lib/$(UAL_ARCH) -lTeapot
```

Finally, the next two sections show the exact code for the "DaIntegratorFactory" class which is cloned from

```
$UAL/codes/TEAPOT/src/TEAPOT/Integrator/DaIntegratorFactory.hh
$UAL/codes/TEAPOT/src/TEAPOT/Integrator/DaIntegratorFactory.cc
```

5 $UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DaIntegratorFactory.hh

```
 1 #ifndef UAL_ETEAPOT_DA_INTEGRATOR_FACTORY_HH
 2 #define UAL_ETEAPOT_DA_INTEGRATOR_FACTORY_HH
 3
 4 #include "ETEAPOT/Integrator/DipoleDaIntegrator.hh"
 5
 6 namespace ETEAPOT {
 7
 8   /** Factory of the TEAPOT DA integrators */
 9
10   class DaIntegratorFactory {
11
12   public:
13
14     /** Returns the DA integrator specified by the element type */
15     // static BasicDaIntegrator* createDaIntegrator(const std::string& type);
16
17     /** Returns the dipole DA integrator */
18     static DipoleDaIntegrator* createDipoleDaIntegrator();
19
20   };
21
22   class DaIntegratorRegister
23   {
24     public:
25
26     DaIntegratorRegister();
27   };
28
29
30 }
31
32 #endif
```

6 $UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DaIntegratorFactory.cc

```
1 #include "UAL/APF/PropagatorFactory.hh"
2 #include "ETEAPOT/Integrator/DaIntegratorFactory.hh"
3
4 ETEAPOT::DipoleDaIntegrator* ETEAPOT::DaIntegratorFactory::createDipoleDaIntegrator()
5 {
6   return new ETEAPOT::DipoleDaIntegrator();
7 }
8
```

```
 9 ETEAPOT::DaIntegratorRegister::DaIntegratorRegister()
10 {
11   UAL::PropagatorNodePtr dipolePtr(new ETEAPOT::DipoleDaIntegrator());
12   UAL::PropagatorFactory::getInstance().add("ETEAPOT::DipoleDaIntegrator", dipolePtr);
13 }
14
15 static ETEAPOT::DaIntegratorRegister theSingleton;
```

7 Conclusions

Now

```
./evolver ./data/pre-E_pEDm.sxf
```

reports that it is using the ETEAPOT DipoleDaIntegrator e.g.

```
Element-by-element propagation.
0 TEAPOT::DriftDaIntegrator
1 TEAPOT::DriftDaIntegrator
2 TEAPOT::MltDaIntegrator
3 TEAPOT::MltDaIntegrator
4 TEAPOT::DriftDaIntegrator
5 TEAPOT::MltDaIntegrator
6 TEAPOT::MltDaIntegrator
7 ETEAPOT::DipoleDaIntegrator
8 TEAPOT::MltDaIntegrator
9 TEAPOT::MltDaIntegrator
10 TEAPOT::MltDaIntegrator
11 TEAPOT::MltDaIntegrator
12 ETEAPOT::DipoleDaIntegrator
13 TEAPOT::MltDaIntegrator
14 ...
```

The code in evolver for this is

```
122  UAL::APDF_Builder apBuilder;
123  apBuilder.setBeamAttributes(ba);
124
125  UAL::AcceleratorPropagator* ap = apBuilder.parse(evolver_apdfFile);
126
127  if(ap == 0) {
128   std::cout << "Accelerator Propagator has not been created " <<
129 std::endl;
130   return 1;
131  }
132
133  std::cout << "\n  , apdf-based evolver";  std::cout << "size : " <<
```

```
ap->getRootNode().size() << " propagators " << endl;

 UAL::PropagatorSequence& apSeq = ap->getRootNode();

 int counter = 0;
 std::list<UAL::PropagatorNodePtr>::iterator it;  for(it =
apSeq.begin(); it != apSeq.end(); it++){
  std::cout << counter++ << " " << (*it)->getType() << std::endl;  }

// ***************************************************************************
 std::cout << "\nPropagate map. " << std::endl;
// ***************************************************************************

 PacTMap map2(6);
 map2.setEnergy(ba.getEnergy());
 map2.mltOrder(2);

 ap->propagate(map2);

 map2.write("evolver.map");

// ***************************************************************************
 std::cout << "\nElement-by-element propagation. " << std::endl;
// ***************************************************************************

 counter = 0;
 char mFile[40];
 for(it = apSeq.begin(); it != apSeq.end(); it++){

 PacTMap map3(6);
 map3.setEnergy(ba.getEnergy());
 map3.mltOrder(1);

 (*it)->propagate(map3);
 std::cout << counter++ << " " << (*it)->getType() << std::endl;

 sprintf(mFile, "map%d.map", counter);
 map3.write(mFile);
 }



 return 1;
}
```

with the heading Element-by-element propagation on line 156.

8 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 > ! .1mm_0_.1mm_0
cat JTout | grep rpOut > ! .1mm_0_.1mm_0.orbit

gnuplot
set terminal x11 size 478,145
set format x "%5.0f"
set format y "%7.5f"
p JT.orbit u 4
p JT.orbit u 6

xwd>figureName.xwd
convert figureName.xwd figureName.eps
```