John Talman

# Unified Accelerator Libraries (UAL)

## ETEAPOT Status With EDM Experiment Emphasis

# E(lectric)TEAPOT Partially Cloned From(magnetic)TEAPOT

(Thin Element Accelerator
Program For Optics And
Tracking)

Little electric/magnetic distinction for Quads and Sexts (e.g. fringe field).

"Exactly" solve Bends (treating them as thick elements) in ETEAPOT(as contrasted with modeling them as kicks in UAL/TEAPOT). Correct for deviant radial field dependence using kicks.

Status:

Physical Ring – Stable
Lattice – Stable
Survey – Finished
Orientation – Stabilizing
Equations – Stabilizing
Code – Stabilizing

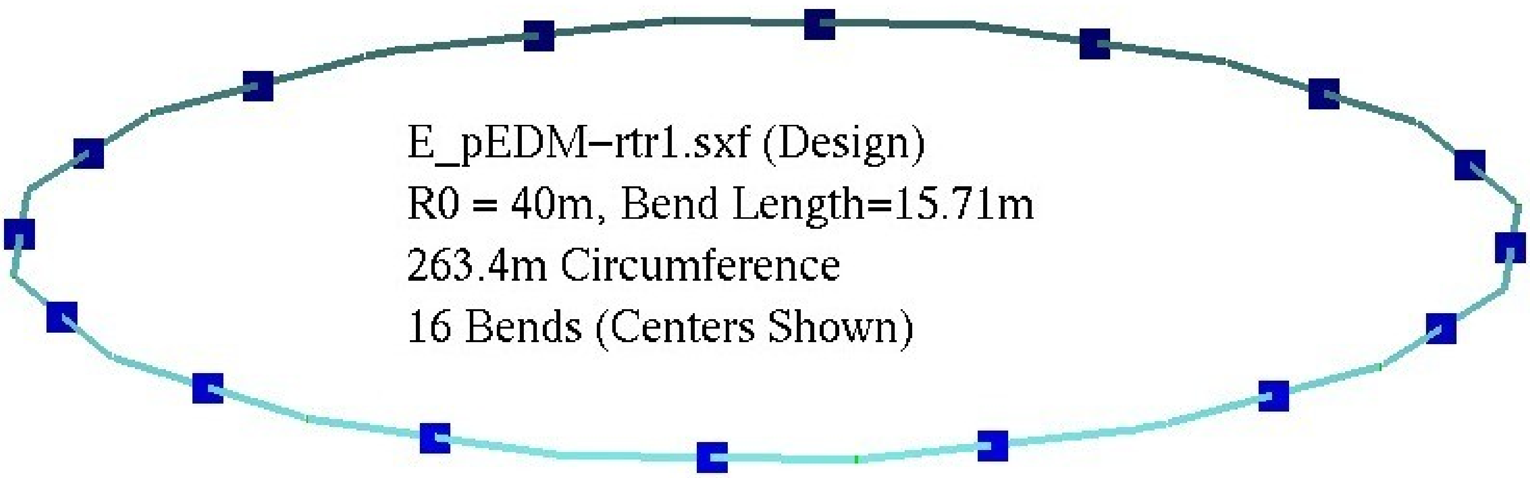
Probably biased to the optimistic side!

# Status: Physical Ring

Status: Physical Ring

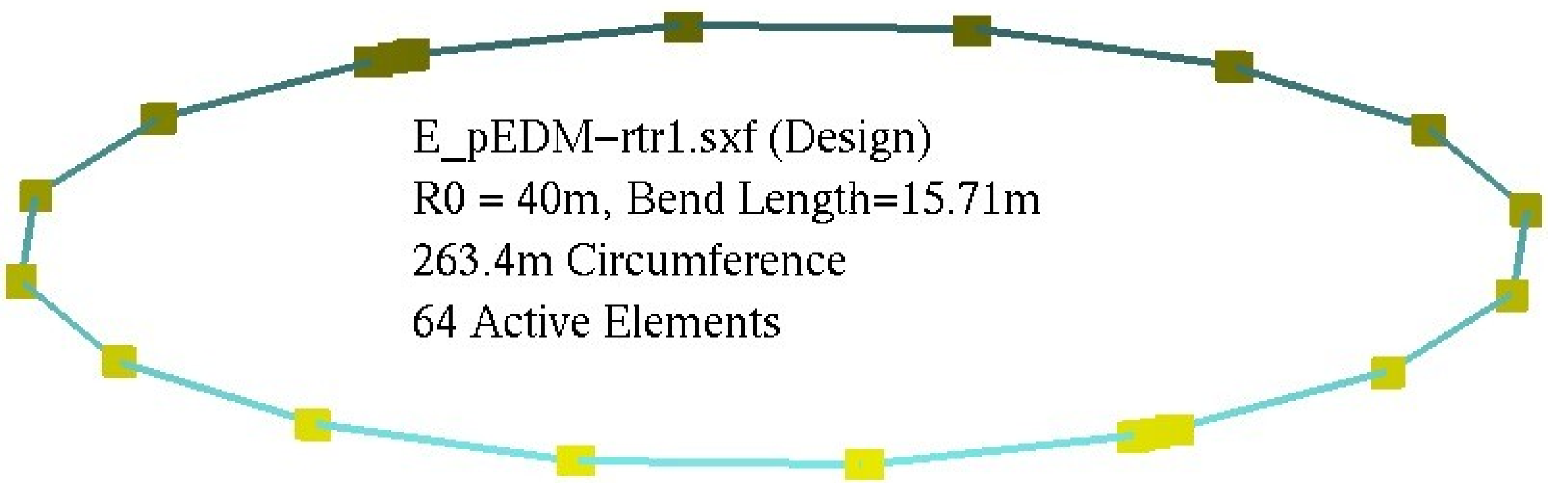
Stable

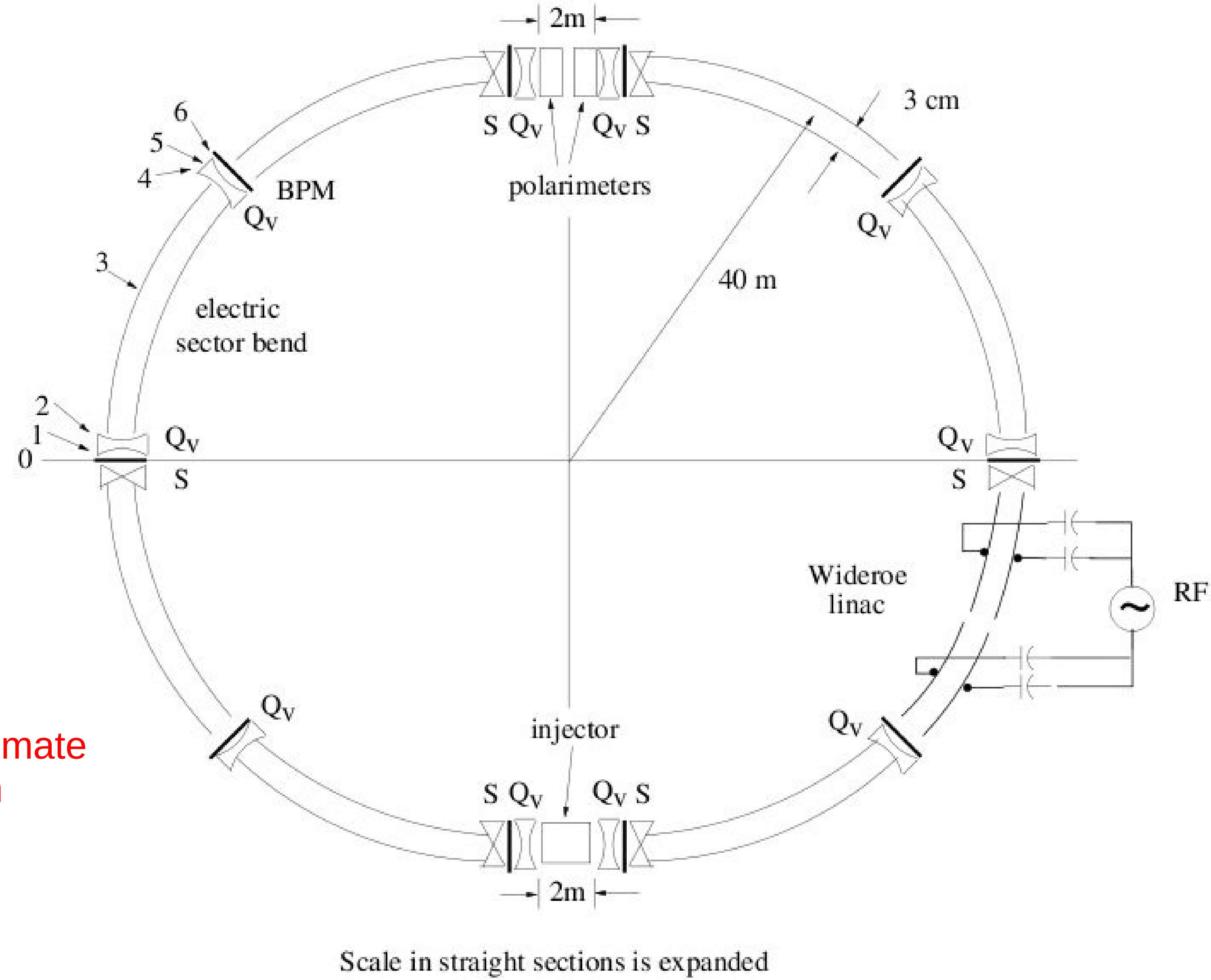
# Status: Lattice




Drift, Kick, Drift --->>> Exact, Split Boundary Kick, Exact

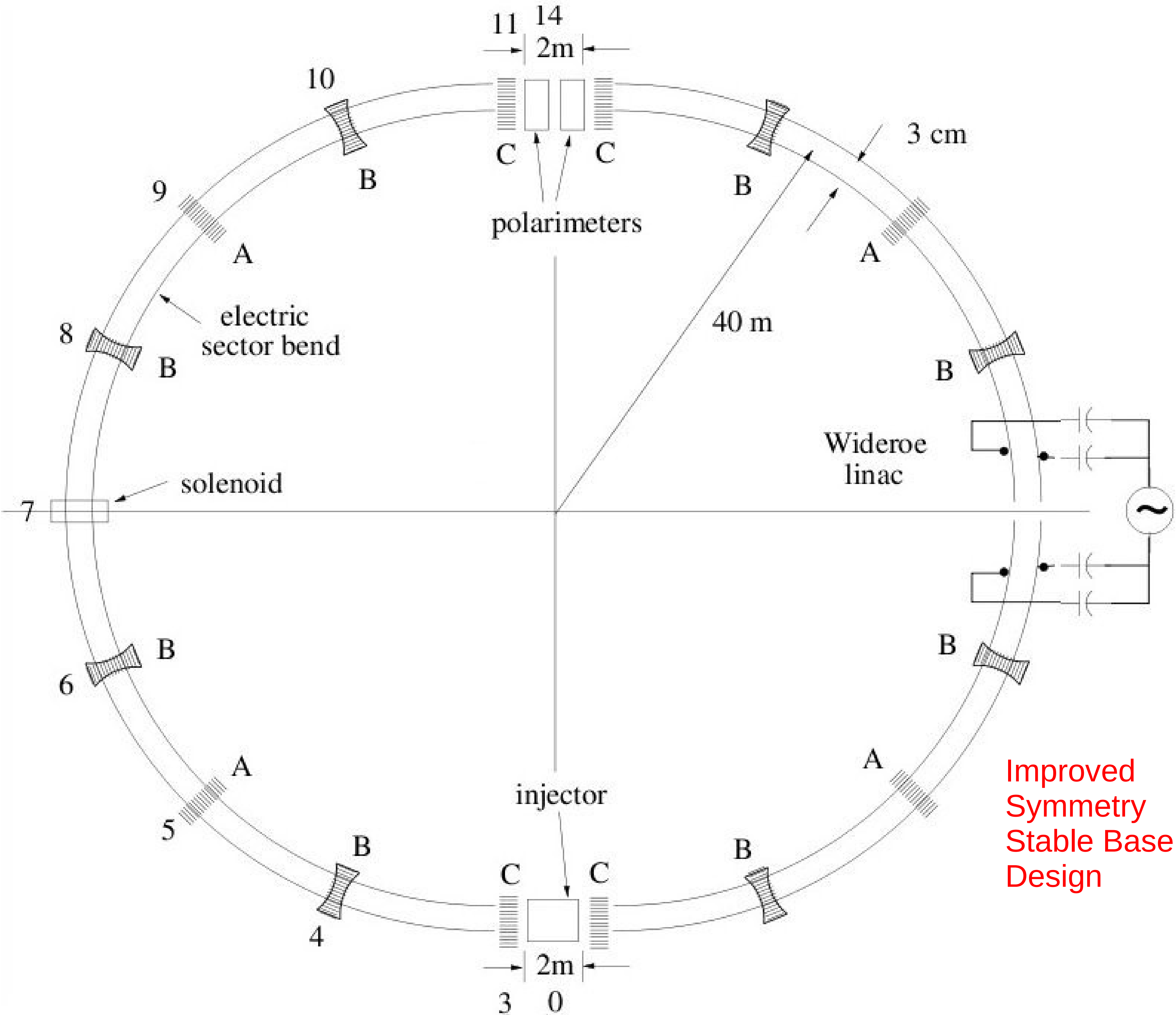
# Status: Lattice






Penultimate
Design

Figure 1: Full ring of the wCFSD, (weak) Combined-Focusing, Separate-Defocusing, lattice. The horizontal focusing is provided by the (combined function) saddle-shaped ($m = -1.2$) electrodes. (Except for the extremely short straightaways of the racetrack,the ring is made from nothing but eight of these cells. Not all elements are shown.




Improved
Symmetry
Stable Baseline
Design

Status: Lattice

Stable

Status: Survey

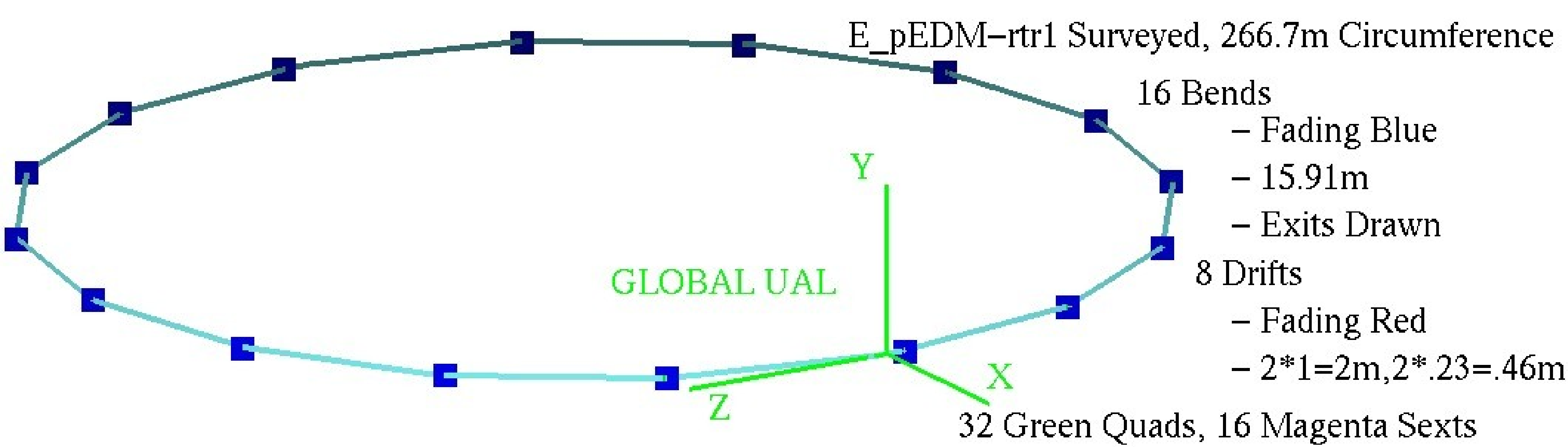
Very Good Corroboration

Fully Arc Length Sequenced

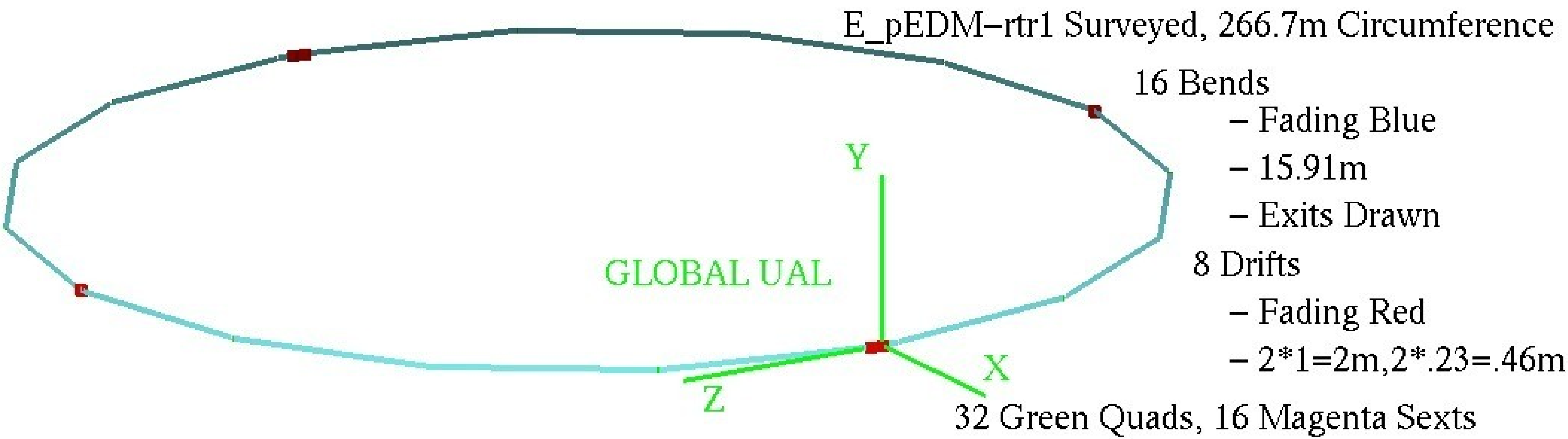
Absolute Coordinates

(Not Strictly Necessary)

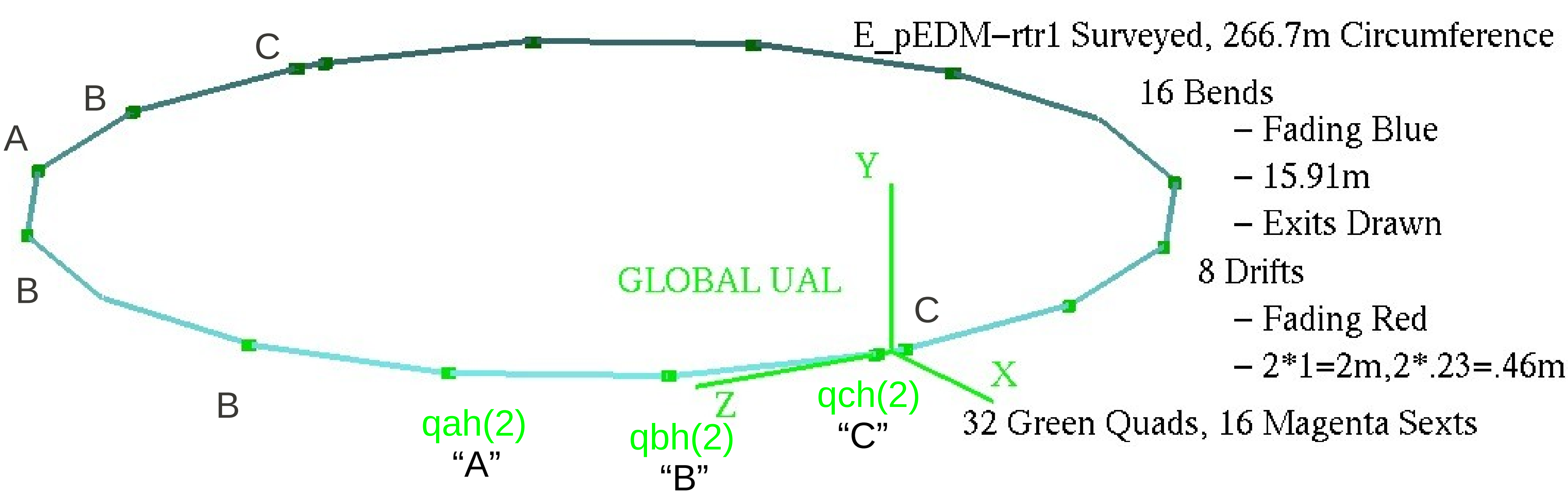
# Status: Survey



# Status: Survey



# Status: Survey




Fundamental Symmetries:

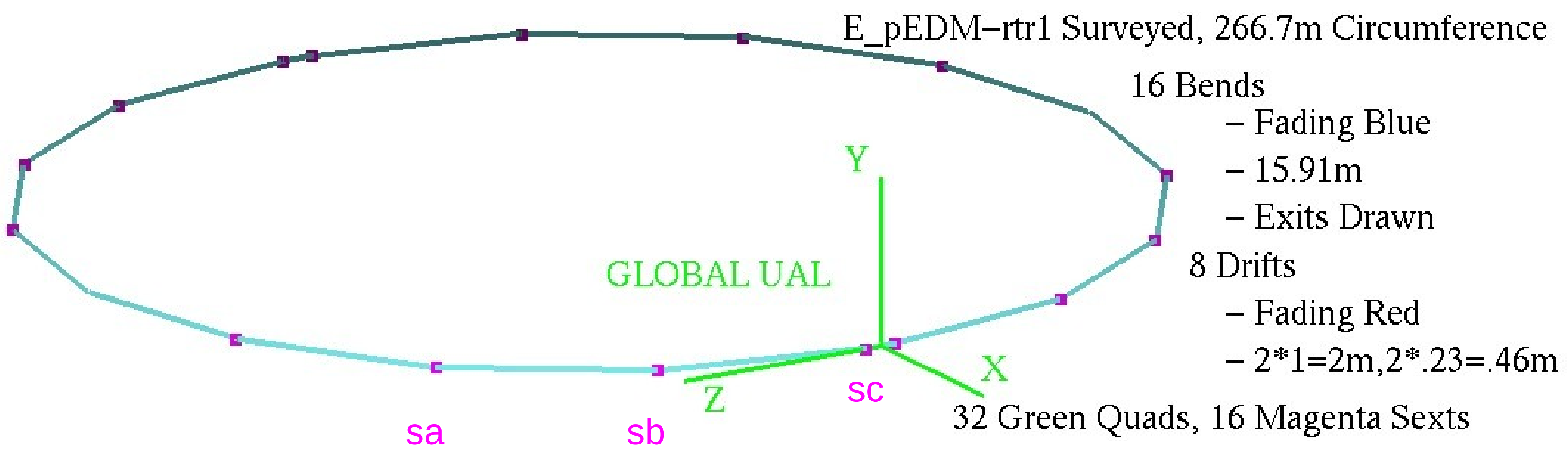
Superperiodicity – 2

Mirror:

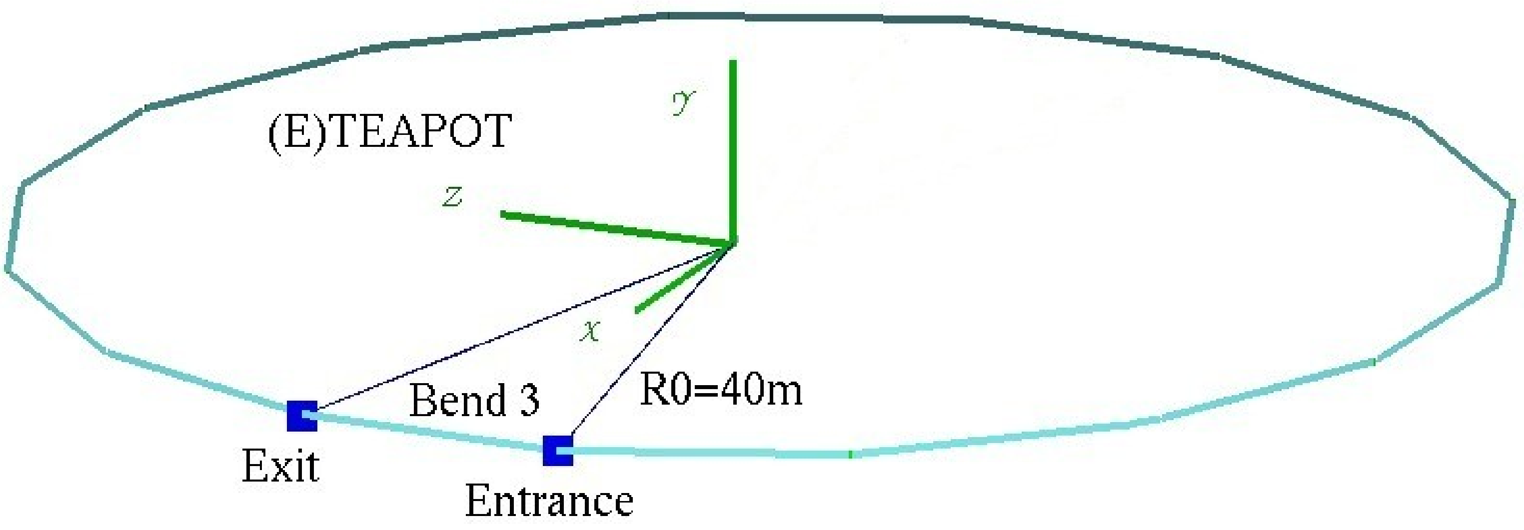
East – West (except for RF ↔ Solenoid)

North – South (except for Injection ↔ Polarimeter)

# Status: Survey



# Status: Survey




**Not Strictly Necessary**

Status: Survey

Finished

## Status: Orientation

Only Need <u>Local</u> Coordinates Relative To The Appropriate Reference/Design Orbit!

New Initial Value Problem (F=dp/dt) At Each Interval

Status: Orientation

Reference Orbit:

Drift – Straight Line
Bend – Circular Arc
Quad – Straight Line
Sext – Straight Line

Status: Orientation

Reference Orbit:

p[0], p[2], p[4] are coordinates referred to the reference orbit of the particular element.

p[1], p[3], p[5] are the canonical conjugates.

# Status: Orientation

## Reference Orbit:

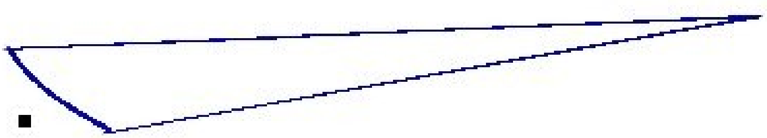
```
p[0] = x
p[2] = y
p[4] = time delta

p[1] = px/p0
p[3] = py/p0
p[5] = energy delta
```

(Deviations From Reference Orbit)

# Reference Orbit - Bend

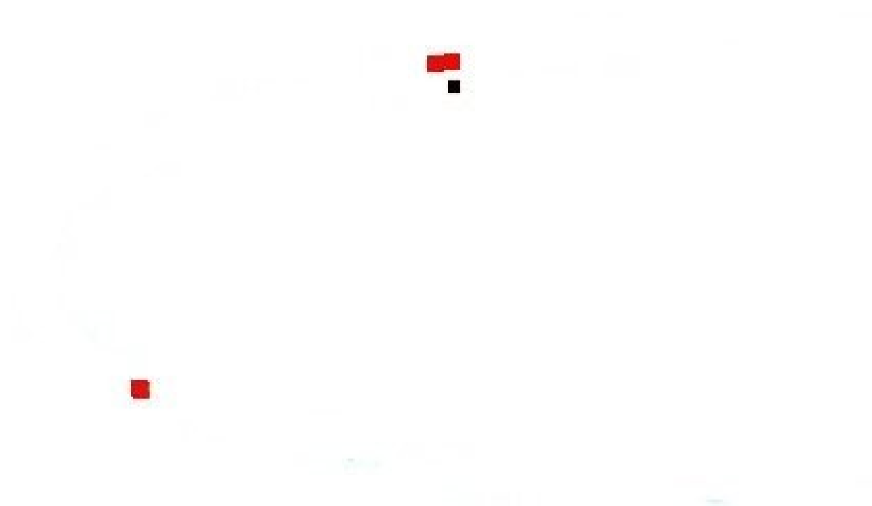
$$\mathbf{E_m} = -E_0 \frac{R_0^{1+m}}{r^{1+m}} \hat{\mathbf{r}}$$



$$V(r) = -\frac{E_0 R_0}{m} \left( \frac{R_0^m}{r^m} - 1 \right)$$

- particle/probe has R0 available
- r = r(x,y)
- r close to R0+p[0]
- V(r) = ln(r)
- or 1/r
- or (1/r)^m
- …
- treat “exactly” (thick element, precision of machine, analytical solutions), correct with kick

# Reference Orbit - Drift,Quad,Sext



- supposed to indicate straight sections!
- same as magnetic
- probe x = p[0]
- px = p[1] * p0
- y = p[2]
- …
- same as TEAPOT

# Status: Orientation

Element – Algorithm – Probe

UAL algorithms are developed with respect to reference system of the particular element. In short, we (beam physics) need to know only relative positions represented by PAC::Position (probe).

GOOD SOFTWARE ARCHITECTURE!

Status: Orientation

Stabilizing

# Status: Equations

The cylindrical coordinate $\rho$ is approximately equal to the spherical coordinate r (nearly co-planar orbits, very close to the horizontal plane)

Conservation of energy, differentiation with respect to $\theta$, ...

# Status: Equations

$$\xi = 1 - \frac{R_0}{r} = \frac{x/R_0}{1 + x/R_0}, \quad \text{and} \quad \frac{d\xi}{d\theta} = \frac{R_0}{r^2}\frac{dr}{d\theta}.$$

$$\left(\mathcal{E} + eE_0R_0 \ln(1-\xi)\right)^2 = \frac{L_y^2c^2}{R_0^2}\left(\frac{d\xi}{d\theta}\right)^2 + \frac{L_y^2c^2}{R_0^2}(1-\xi)^2 + p_y^2c^2 + m^2c^4$$

$$\begin{aligned}
\frac{d^2\xi}{d\theta^2} &= 1 - \xi - \mathcal{E}\,\frac{eE_0R_0^3}{L_y^2c^2}\frac{1}{1-\xi} - \frac{e^2E_0^2R_0^4}{L_y^2c^2}\frac{\ln(1-\xi)}{1-\xi} \\
&= 1 - \xi - \frac{\mathcal{E}/e}{L_yc/(eR_0)}\frac{E_0R_0}{L_yc/(eR_0)}\frac{1}{1-\xi} - \left(\frac{E_0R_0}{L_yc/(eR_0)}\right)^2\frac{\ln(1-\xi)}{1-\xi} \\
&= 1 - \xi - \frac{\mathcal{E}}{\mathcal{E}_0}\frac{L_0^2}{L_y^2}\frac{1}{1-\xi} - \frac{L_0^2}{L_y^2}\beta_0^2\frac{\ln(1-\xi)}{1-\xi}.
\end{aligned}$$

Why ξ? It permits analytical solution in special cases, and ξ = x/R0 in linear approximation.
Thus, ...

# Status: Equations

$$(1-\xi)^{-1} = 1 + \xi + \xi^2 + \ldots, \quad \frac{\ln(1-\xi)}{1-\xi} = -\xi - \frac{3}{2}\xi^2 + \ldots.$$

The orbit equation becomes

$$\frac{d^2\xi}{d\theta^2} = 1 - \frac{L_0^2}{L_y^2}\frac{\mathcal{E}}{\mathcal{E}_0} - \left(1 + \frac{L_0^2}{L_y^2}\left(\frac{\mathcal{E}}{\mathcal{E}_0} - \beta_0^2\right)\right)\xi - \frac{L_0^2}{L_y^2}\left(\frac{\mathcal{E}}{\mathcal{E}_0} - \frac{3}{2}\beta_0^2\right)\xi^2 + \ldots$$

$$\approx -Q^2(\xi - \xi_0),$$

Nice/Convenient Form!

$$Q^2 = 1 + \frac{L_0^2}{L_y^2}\left(\frac{\mathcal{E}}{\mathcal{E}_0} - \beta_0^2\right), \quad \text{and} \quad \xi_0 = \frac{1 - \frac{L_0^2}{L_y^2}\frac{\mathcal{E}}{\mathcal{E}_0}}{Q^2}.$$

# Status: Equations

$$
\begin{aligned}
C_\xi(\theta) &= \cos(Q\theta)(1-\xi_0)+\xi_0 \\
C'_\xi(\theta) &= -\sin(Q\theta)\,Q(1-\xi_0) \\
S_\xi(\theta) &= \frac{\sin(Q\theta)}{Q} - \cos(Q\theta)\,\xi_0+\xi_0 \\
S'_\xi(\theta) &= \cos(Q\theta)+\sin(Q\theta)\,Q\xi_0
\end{aligned}
$$

cosine-like orbit/trajectory

sine-like orbit/trajectory

# Status: Equations

$$\xi_{\rm in} = \frac{x_{\rm in}}{R_0 + x_{\rm in}}, \quad \xi'_{\rm in} = \frac{R_0 x'_{\rm in}}{(R_0 + x_{\rm in})^2}.$$

$$\frac{x/R_0}{1 + x/R_0} = C_\xi(\theta)\,\xi_{\rm in} + S_\xi(\theta)\,\xi'_{\rm in}.$$

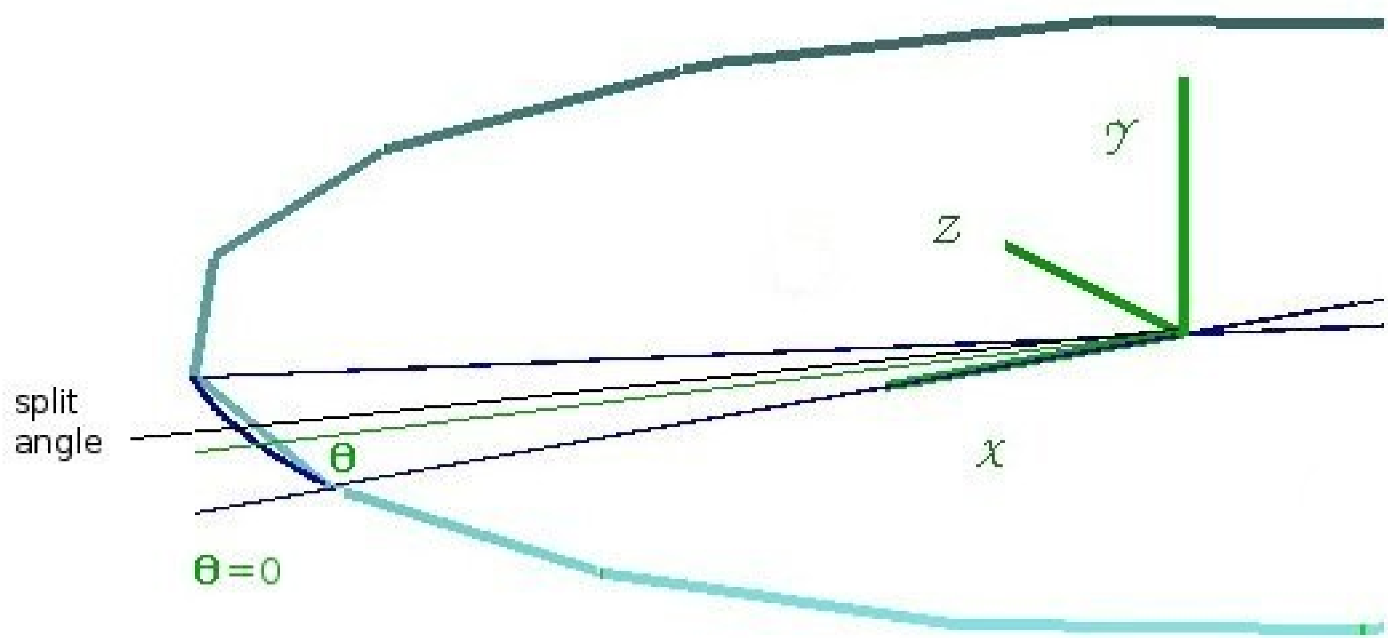
$$\frac{x}{R_0} = \frac{C_\xi(\theta)\,\xi_{\rm in} + S_\xi(\theta)\,\xi'_{\rm in}}{1 - C_\xi(\theta)\,\xi_{\rm in} - S_\xi(\theta)\,\xi'_{\rm in}}$$

$$\frac{x'}{R_0} = \frac{C'_\xi(\theta)\xi_{\rm in} + S'_\xi(\theta)\xi'_{\rm in}}{\left(1 - C_\xi(\theta)\,\xi_{\rm in} - S_\xi(\theta)\,\xi'_{\rm in}\right)^2}.$$

"Exact"
Output in terms of Input

Stable!

# Status: Equations



# Status: Equations

Stabilizing

# Status: Code

## Bunch loop

```
PAC::Position& p =
bunch[ip].getPosition();
```

(currently 1 probe in bunch)

## Splitting

simplest split (into 2 equal halves)

# Status: Code

## const PAC::BeamAttributes cba

```
double e0=cba.getEnergy();
double m0=cba.getMass();
double q0=cba.getCharge();
double t0=cba.getElapsedTime();
double f0=cba.getRevfreq();
double M0=cba.getMacrosize();
double G0=cba.getG();
double L0=cba.getL();          // hybrid? p0 beam, R0 element?
double E0=cba.getE();          // more like an element property
double R0=cba.getR();          // more like an element property
                               // needs to be reworked

      “p0”=sqrt(e0*e0-m0*m0); //  design momentum
          =L0/R0;
      “g0”=e0/m0;             //  design gamma
```

g also stands for gap

# Status: Code

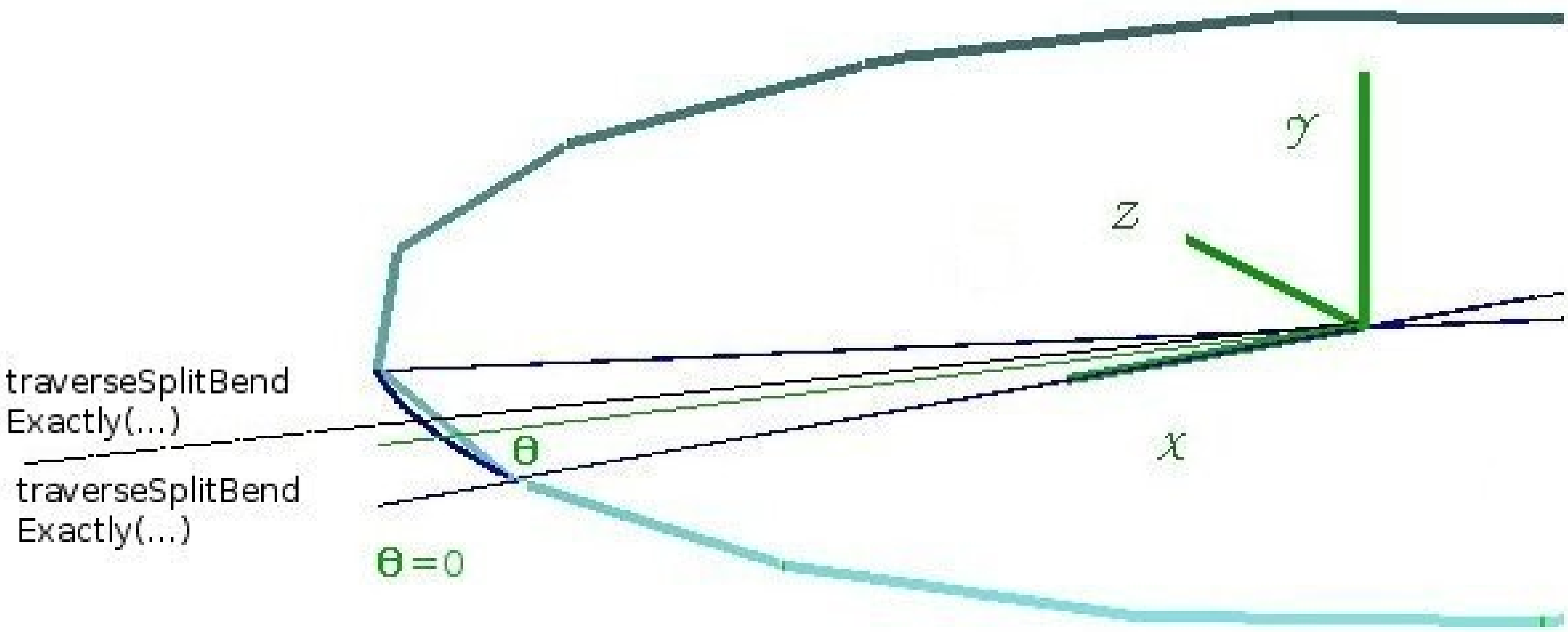
```
// Drift, Kick, Drift --->>> Exact, Split Boundary Kick, Exact


// passBendSlice              (data.m_slices[0], p, tmp, v0byc);
   traverseSplitBendExactly(data.m_slices[0], p, tmp, v0byc,
                             cba,data.m_angle/2);

// applyThinBendKick(data, edata, 1, p, v0byc);
   handleSplitBendBoundary(p,cba);

// passBendSlice(data.m_slices[1], p, tmp, v0byc);
   traverseSplitBendExactly(data.m_slices[1], p, tmp, v0byc,
                             cba,data.m_angle/2);
```

# Status: Code




Plus Corrections at Entrance, Split, and Exit

# Status: Code

```
   enterBendCorrection(p,cba);

// passBendSlice              (data.m_slices[0], p, tmp, v0byc);
   traverseSplitBendExactly(data.m_slices[0], p, tmp, v0byc,
                             cba,data.m_angle/2);

// applyThinBendKick(data, edata, 1, p, v0byc);
   handleSplitBendBoundary(p,cba);

// passBendSlice(data.m_slices[1], p, tmp, v0byc);
   traverseSplitBendExactly(data.m_slices[1], p, tmp, v0byc,
                             cba,data.m_angle/2);

   leaveBendCorrection(p,cba);
```

# Status: Code

```
enterBendCorrection(Coordinates& p,
                    const PAC::BeamAttributes cba){

#include "getDesignBeam.h"

//                    eMD=e0+p0*p[5]                // Mechanical energy - "Drift" (non Bend)
//                    eTD=eMD                       // Total      energy -  Drift
//                    eTB=eTD                       // Total energy conserved
//                       =eMB+V(r)                  // Total      energy -  Bend
//                    eMD=eMB+V(r)                  // substitute
//
//       (eMD-e0)/p0=(eMB-e0)/p0+V(r)/p0            // rearrange
//        p[5]Drift =  p[5]Bend +V(r)/p0            // p[5]Bend is output

double q       = UAL::elemCharge;                   // units
double GeVperJ = 1/q/1e9;                           // units
double  r      = R0+p[0];
double p0      = sqrt(e0*e0-m0*m0);                 // derived beam momentum

       p[5]    = p[5]-GeVperJ*getPotentialEnergy(q,E0,R0,r)/p0;
}
```

# Status: Code

```
double getPotentialEnergy(double q0,double E0,double R0,double r){
     return q0*E0*R0*log(r/R0);
 }

 double Cxi(double Q,double theta,double xi0){
     double value=cos(Q*theta)*(1-xi0)+xi0;
     return value;
 }

 double CxiP(double Q,double theta,double xi0){
     double value=-sin(Q*theta)*Q*(1-xi0);
     return value;
 }

 double Sxi(double Q,double theta,double xi0){
     double value=sin(Q*theta)/Q-cos(Q*theta)*xi0+xi0;
     return value;
 }

 double SxiP(double Q,double theta,double xi0){
     double value=cos(Q*theta)+sin(Q*theta)*Q*xi0;
     return value;
 }
```

Convenient Equations!

# Status: Code

- good start on stubs with doubles
- need better time of flight (p[4])
- need better p[2], p[3] (y and py)
- TODO:
  - testing
  - benchmarking
  - spin
  - (vector of) truncated power series (polymorphism)

Status: Code

Stabilizing

# Status: Summary

- in control of geometry,
  element sequencing,
  code sequencing,
  probe parameters

- good UAL software architecture allows the algorithm to be developed independently

- probably about ½ way to code content

# Thanks To

Nikolay Malitsky
Richard Talman
Yannis Semertzidis
Bill Morse
Selcuk Haciomeroglu
Alfredo Luccio
...

## Status:

Physical Ring – Stable
Lattice – Stable
Survey – Finished
Orientation – Stabilizing
Equations – Stabilizing
Code – Stabilizing

Probably biased to the optimistic side!

# Free Associating

# Flight Time

- Flight time is given by an integral which is not in the spirit of the TEAPOT approach.

- Flight time should be viewed as a split operation; The higher the split # (finer the splitting), the more accurate the flight time.

$$mc^2\gamma^I(\theta) = \mathcal{E} - eE_0\big(C_\xi(\theta)\,x_0 + S_\xi(\theta)\,x_0'\big).$$

From this formula one can obtain $\beta^I$ using

$$\beta^{I^2}(\theta) = 1 - \frac{1}{\gamma^{I^2}(\theta)}.$$

For motion in the horizontal plane, the angular velocity can be obtained star
from the $y$-component of Eq. (9);

$$\frac{d\theta}{dt} = \frac{L}{mr^2\gamma^I}.$$

The right hand side of this equation can be expressed in terms of $\theta$, invari
and initial conditions using Eq. (48). This relation is useful primarily to ob
the flight time through bend elements. After rearrangement, the flight t
$t_o - t_i$ from input to output of a bend element is given by

$$t_o - t_i = \frac{m}{L}\int_{\theta_i}^{\theta_o} r^2(\theta')\gamma^I(\theta')\,d\theta' \approx \frac{mR_0^2}{L}\int_{\theta_i}^{\theta_o} \big(1 + 2x(\theta')/R_0\big)\gamma^I(\theta')\,d\theta'.$$

# Acceptance/
# Admittance/
# Emittance

- E0 central/design/ideal/reference Electric Field
  1.7E7 V/m

- g design gap
  3cm?

# Acceptance/ Admittance/ Emittance

Are "admittance" and "acceptance" more or less synonymous?

Yes. Furthermore "emittance" (which applies to a beam) and "admittance" (which applies to a lattice) are commensurate (same units) quantities. The lattice admittance has to exceed the beam emittance in order for the whole beam to be captured. "Dynamic aperture" is similar to "admittance" though it has different units since it is just the maximum stable particle transverse displacement from the design orbit.

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleTracker.hh
   68   static newDipoleAlgorithm<double,PAC::Position> s_algorithm;

$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleTracker.cc
   16   ETEAPOT::newDipoleAlgorithm<double,PAC::Position> ETEAPOT::DipoleTracker::
                                                                   s_algorithm;



--->>>


   68   static newDipoleAlgorithm<ZLIB::Tps,ZLIB::VTps> s_algorithm;

   16   ETEAPOT::newDipoleAlgorithm<ZLIB::Tps,ZLIB::VTps> ETEAPOT::DipoleTracker::
                                                                   s_algorithm;
```

```
$UAL/codes/ETEAPOT/src/ETEAPOT/Integrator/DipoleTracker.cc
   93     for(int ip = 0; ip < bunch.size(); ip++) {
    if(bunch[ip].isLost()) continue;
    PAC::Position& p = bunch[ip].getPosition();      <<<---
    tmp = p;
    s_algorithm.passEntry(m_edata, p);
    s_algorithm.makeVelocity(p, tmp, v0byc);
    s_algorithm.makeRV(p, tmp, e0, p0, m0);

    s_algorithm.passBend(m_data, m_edata, p, tmp, v0byc, cba);
//                    [      original interface      ] [central orbit ]
    s_algorithm.passExit(m_edata, p);
    // testAperture(p);
  }



$UAL/examples/ETEAPOT/tracker.cc
   50   UAL::Shell shell;

          // *******************************************************************
          std::cout << "\nDefine the space of Taylor maps." << std::endl;
          // *******************************************************************

          shell.setMapAttributes(UAL::Args() << UAL::Arg("order", 5));   <<<---
```

## Simulation Initialization

Beam -

```
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("energy",               e0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("mass",                 m0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("charge",               q0));
//shell.setBeamAttributes(UAL::Args() << UAL::Arg("elapsedTime",          t0));
//shell.setBeamAttributes(UAL::Args() << UAL::Arg("frequency",            f0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("macrosize",            M0));
//shell.setBeamAttributes(UAL::Args() << UAL::Arg("gyromagnetic",         G0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("designAngularMomentum",L0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("designElectricField",  E0));
  shell.setBeamAttributes(UAL::Args() << UAL::Arg("designRadius",         R0));
```

# Simulation Initialization

Probe -

```
for(int ip=0;ip<bunch.size();ip++){
   bunch[ip].getPosition().set
        (probe__dx0,probe_dpx0,probe__dy0,probe_dpy0,probe_cdt0,probeEscr0);
   bunch[ip].setSpin(spin);
}
```

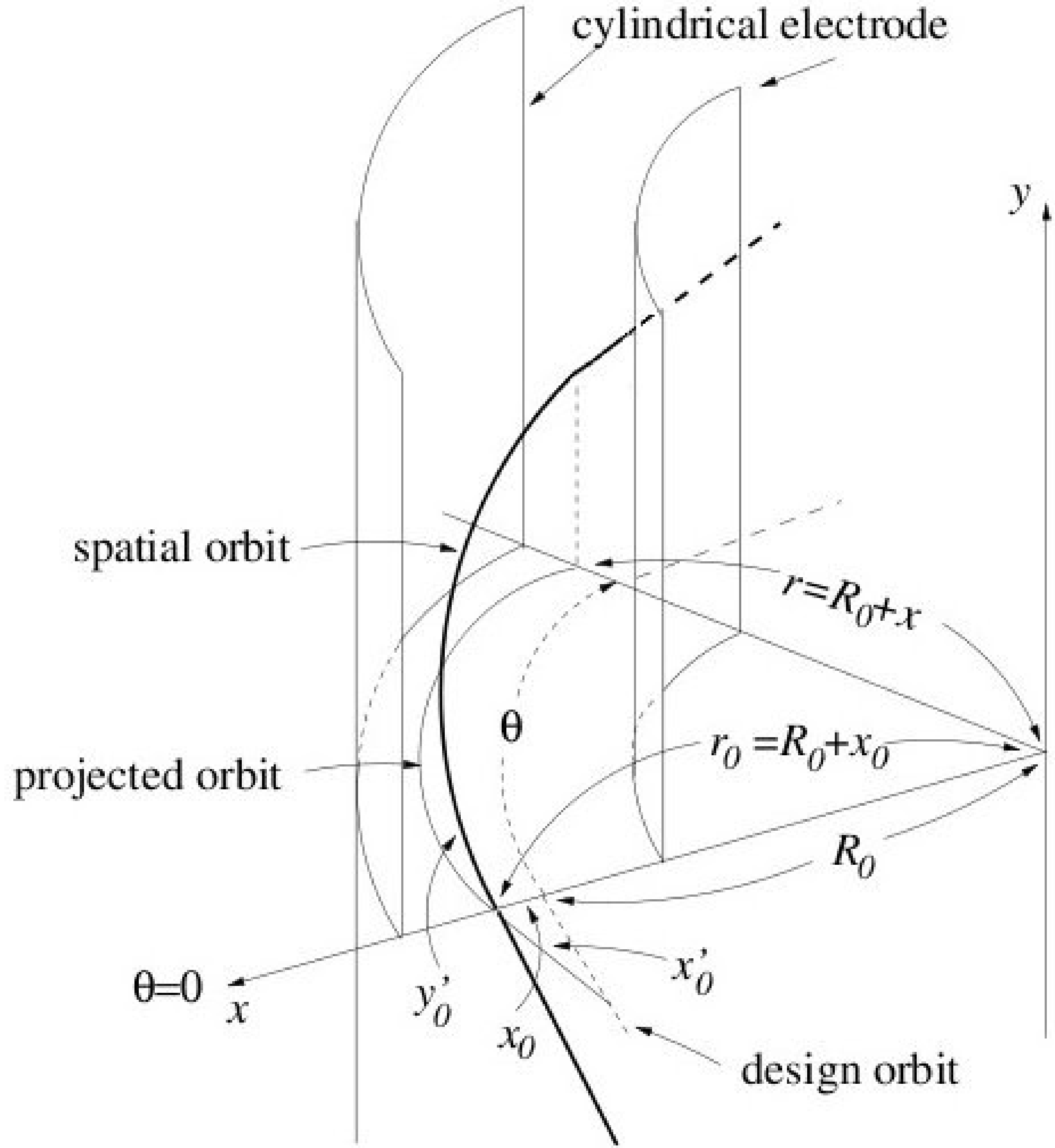



Figure 1: The bold curve shows a proton orbit passing through a curved-planar cylindrical electrostatic bending element. The electrode spacing is $g$ and the design orbit is centered between the electrodes.

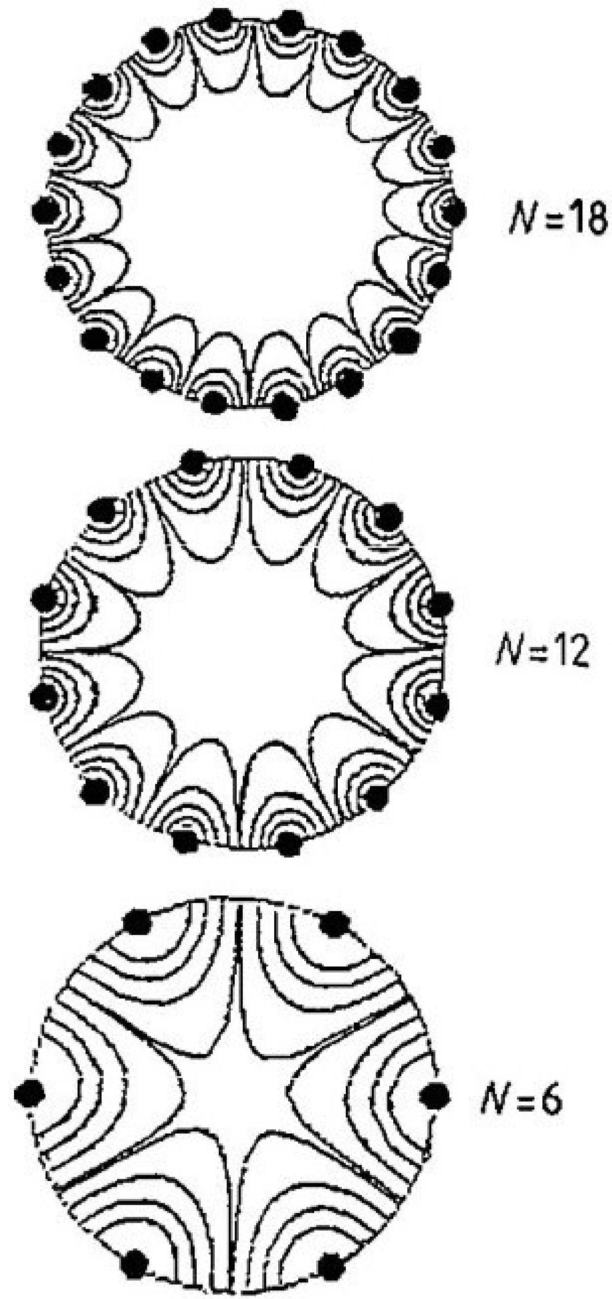



**Figure 3.** Configuration of multipole magnetic field in a 'squirrel cage' scheme.