Model Update

Nikolay Malitsky, APEX , November 2, 2007

Outline:

❑ New Development (from Apex 2006)

- Online nonlinear chromaticity correction

❑ 2006-2007 Upgrade and Commissioning

- Online nonlinear chromaticity correction
- Beta* squeeze
- Tune fitting at the near-integer working point
- RhicLiveOrbitCorrector: the first sequel of the OrbitCalc library
- Resolving the dispersion issue in the d-Au online model

❑ Next Step

Nonlinear Chromaticity Correction

HARMON Approach

Chapter 12. HARMON Module

The HARMON calculations are based on a pr
MAD by D. Schofield [8]. Various changes and

HARMON algorithm is based on linear optics => it turns out as (1) a natural extension and (2) a precise test of the OptiCalc linear model

12.3 Higher-order Chromaticities

The second- and third-order chromaticities consider only the effects of quadrupoles and higher-order multipoles. We introduce the functions

$$
\begin{aligned}
f_{x1}(s) &= \beta_x(s)\Big(K_1(s) - K_2(s)Dx(s)\Big),\\
f_{y1}(s) &= \beta_y(s)\Big(K_1(s) - K_2(s)Dx(s)\Big),\\
f_{x2}(s) &= -f_{x1} - \beta_x(s)\Big(K_2(s)D^2x(s) + \frac{1}{2}K_3(s)(Dx(s))^2\Big),\\
f_{y2}(s) &= -f_{y1} - \beta_y(s)\Big(K_2(s)D^2x(s) + \frac{1}{2}K_3(s)(Dx(s))^2\Big),\\
f_{x3}(s) &= -f_{x2} - \beta_x(s)\Big(K_2(s)D^3x(s) + K_3(s)Dx(s)D^2x(s) + \frac{1}{6}K_4(s)(Dx(s))^3\Big),\\
f_{y3}(s) &= -f_{y2} - \beta_y(s)\Big(K_2(s)D^3x(s) + K_3(s)Dx(s)D^2x(s) + \frac{1}{6}K_4(s)(Dx(s))^3\Big).
\end{aligned}
\tag{12.4}
$$

$$
\begin{aligned}
\frac{d^3Q_x}{dp_t^3} = &-\frac{1}{4\pi}\int_0^C f_{x3}(s)ds\\
&-\frac{1}{4\pi\sin(2\pi Q_x)}\int_0^C\int_0^{s_1}\Big(f_{x1}(s_1)f_{x2}(s_2) + f_{x2}(s_1)f_{x1}(s_2)\Big)\times\\
&\quad\times\sin\Big(\mu_x(s_2)-\mu_x(s_1)\Big)\sin\Big(2\pi Q_x + \mu_x(s_1)-\mu_x(s_2)\Big)ds_2ds_1\\
&+\frac{1}{4\pi\sin(2\pi Q_x)}\int_0^C\int_0^{s_1}\int_0^{s_2} f_{x1}(s_1)f_{x1}(s_2)f_{x1}(s_3)\times\\
&\quad\times\sin\Big(\mu_x(s_3)-\mu_x(s_2)\Big)\sin\Big(\mu_x(s_2)-\mu_x(s_1)\Big)\sin\Big(2\pi Q_x + \mu_x(s_1)-\mu_x(s_3)\Big)ds_3ds_2ds_1
\end{aligned}
$$

Nonlinear Chromaticity Correction

Online Model vs HARMON

1st Order	Natural Chromaticity		+ SF's and SD's		+ b2 of dipoles	
	h	v	h	v	h	v
Harmon	-82.1	-83.1	9.3	6.9	N/A	N/A
OptiCalc	-88.9	-90.0	2			
OptiCalcV2	-82.1	-83.1	8			

Error fixed with an appropriate representation of IR quadrupoles: Simpson's variant vs kick

2nd Order	Natural Chromaticity		+ SF's and SD's	
	h	v	h	v
Harmon	3.9e+3	4.5e+3	1.7e+3	1.4e+2
OptiCalcV2	4.0e+3	4.6e+3	2.0e+3	5.5e+2

Calculation Time:

Linear optics: 6 ms
Nonlinear effects: +2 ms

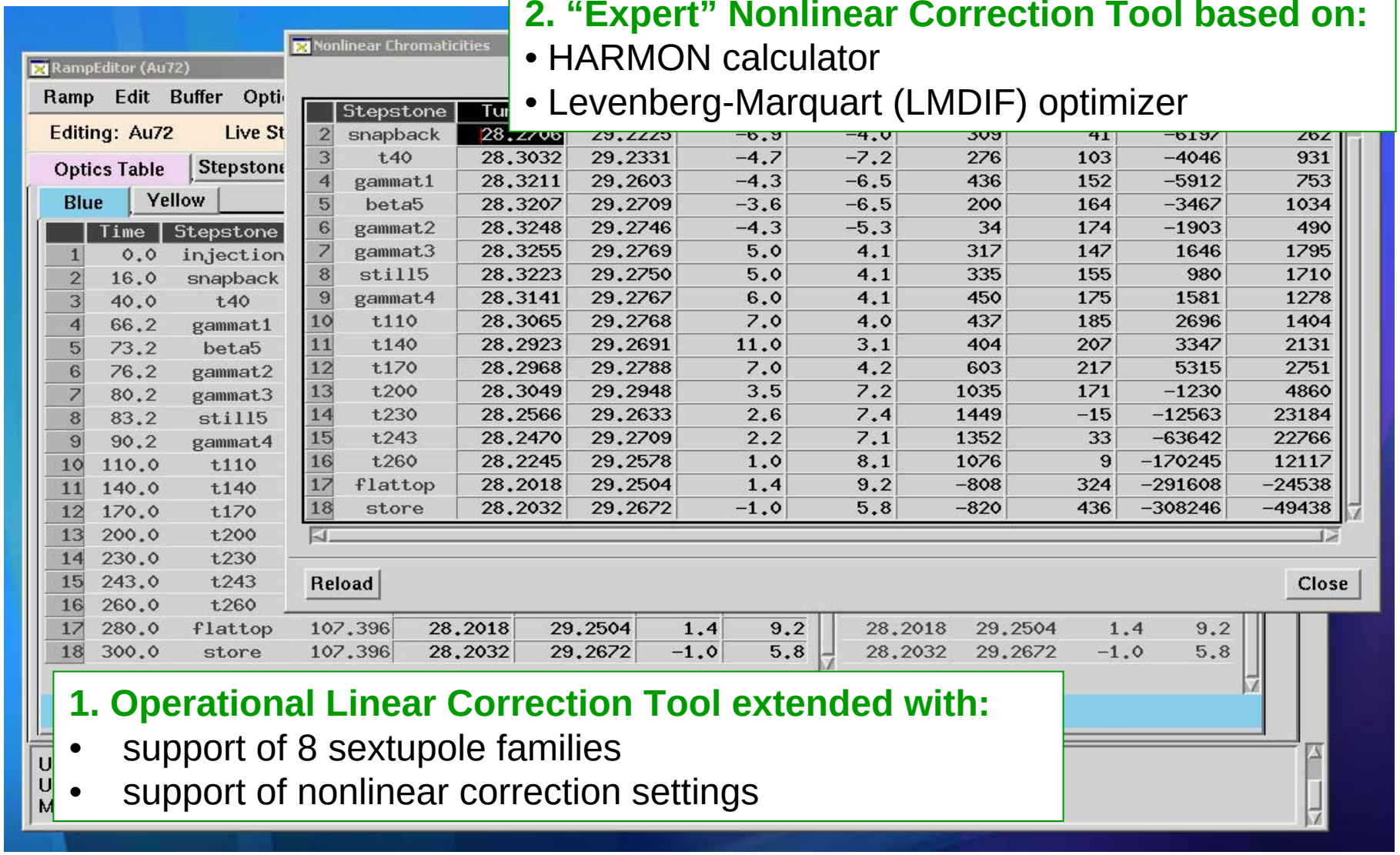
3rd Order	Natural Chromaticity		+ SF's and SD's	
	h	v	h	v
Harmon	2.9e+6	3.1e+6	-3.3e+4	-8.2e+4
OptiCalcV2	2.9e+6	3.2e+6	-3.2e+4	-9.1e+4

Nonlinear Chromaticity Correction

Interface to the linear and nonlinear corrections

2. "Expert" Nonlinear Correction Tool based on:

- HARMON calculator
- Levenberg-Marquart (LMDIF) optimizer



	Time	Stepstone								
1	0.0	injection								
2	16.0	snapback								
3	40.0	t40								
4	66.2	gammat1								
5	73.2	beta5								
6	76.2	gammat2								
7	80.2	gammat3								
8	83.2	still5								
9	90.2	gammat4								
10	110.0	t110								
11	140.0	t140								
12	170.0	t170								
13	200.0	t200								
14	230.0	t230								
15	243.0	t243								
16	260.0	t260								
17	280.0	flattop	107.396	28.2018	29.2504	1.4	9.2	28.2018	29.2504	1.4	9.2
18	300.0	store	107.396	28.2032	29.2672	-1.0	5.8	28.2032	29.2672	-1.0	5.8

	Stepstone	Tu							
2	snapback	28.2706	29.2225	-6.9	-4.0	309	41	-6197	262
3	t40	28.3032	29.2331	-4.7	-7.2	276	103	-4046	931
4	gammat1	28.3211	29.2603	-4.3	-6.5	436	152	-5912	753
5	beta5	28.3207	29.2709	-3.6	-6.5	200	164	-3467	1034
6	gammat2	28.3248	29.2746	-4.3	-5.3	34	174	-1903	490
7	gammat3	28.3255	29.2769	5.0	4.1	317	147	1646	1795
8	still5	28.3223	29.2750	5.0	4.1	335	155	980	1710
9	gammat4	28.3141	29.2767	6.0	4.1	450	175	1581	1278
10	t110	28.3065	29.2768	7.0	4.0	437	185	2696	1404
11	t140	28.2923	29.2691	11.0	3.1	404	207	3347	2131
12	t170	28.2968	29.2788	7.0	4.2	603	217	5315	2751
13	t200	28.3049	29.2948	3.5	7.2	1035	171	-1230	4860
14	t230	28.2566	29.2633	2.6	7.4	1449	-15	-12563	23184
15	t243	28.2470	29.2709	2.2	7.1	1352	33	-63642	22766
16	t260	28.2245	29.2578	1.0	8.1	1076	9	-170245	12117
17	flattop	28.2018	29.2504	1.4	9.2	-808	324	-291608	-24538
18	store	28.2032	29.2672	-1.0	5.8	-820	436	-308246	-49438

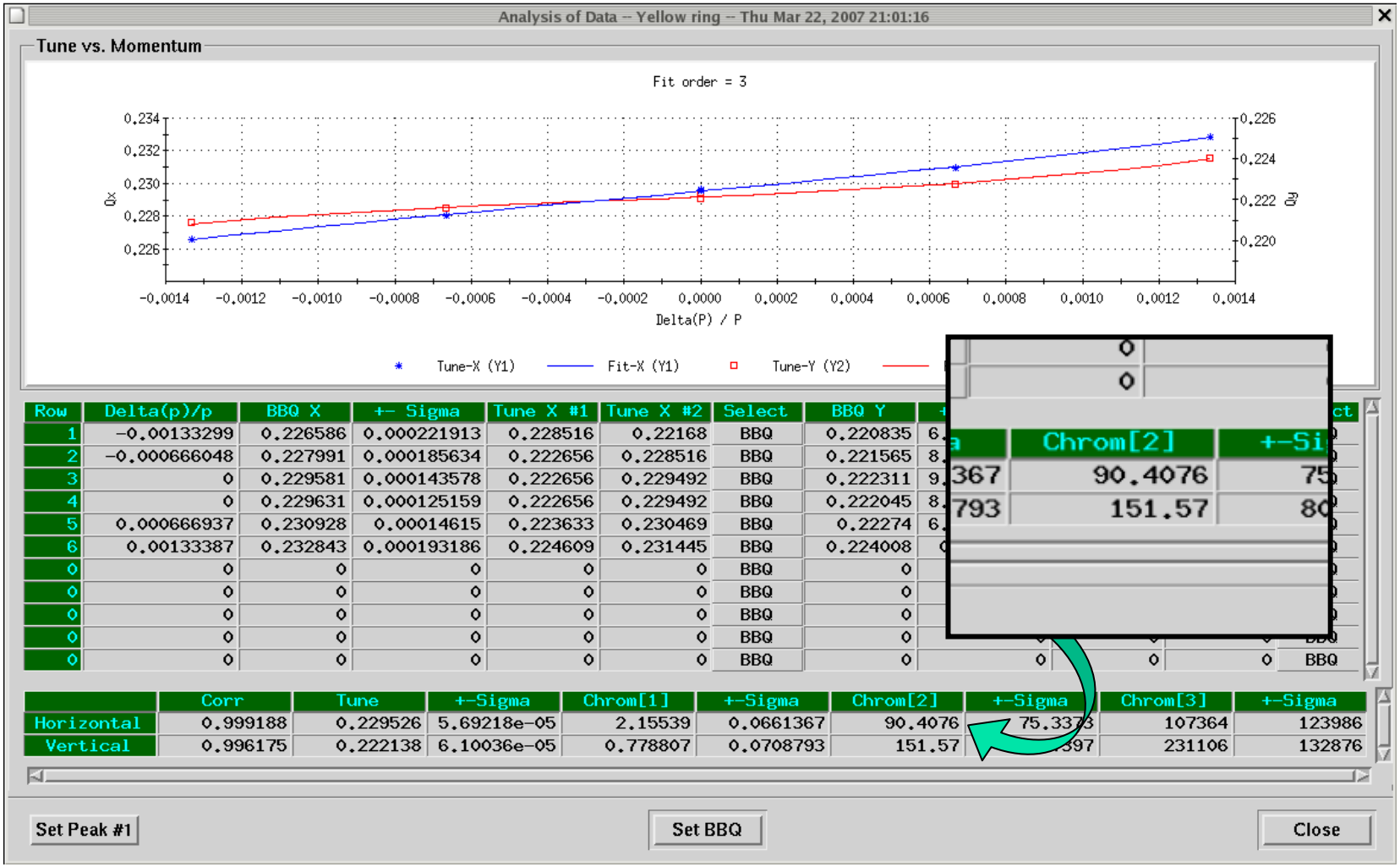
1. Operational Linear Correction Tool extended with:

- support of 8 sextupole families
- support of nonlinear correction settings

Nonlinear Chromaticity Correction

measurement after correction, March 22, 2007

The first attempt, a single shot : 1500 → 151



Row	Delta(p)/p	BBQ X	+- Sigma	Tune X #1	Tune X #2	Select	BBQ Y
1	-0.00133299	0.226586	0.000221913	0.228516	0.22168	BBQ	0.220835
2	-0.000666048	0.227991	0.000185634	0.222656	0.228516	BBQ	0.221565
3	0	0.229581	0.000143578	0.222656	0.229492	BBQ	0.222311
4	0	0.229631	0.000125159	0.222656	0.229492	BBQ	0.222045
5	0.000666937	0.230928	0.00014615	0.223633	0.230469	BBQ	0.22274
6	0.00133387	0.232843	0.000193186	0.224609	0.231445	BBQ	0.224008
0	0	0	0	0	0	BBQ	0
0	0	0	0	0	0	BBQ	0
0	0	0	0	0	0	BBQ	0
0	0	0	0	0	0	BBQ	0
0	0	0	0	0	0	BBQ	0

	Chrom[2]	+-Si
367	90.4076	75
793	151.57	80

	Corr	Tune	+-Sigma	Chrom[1]	+-Sigma	Chrom[2]	+-Sigma	Chrom[3]	+-Sigma
Horizontal	0.999188	0.229526	5.69218e-05	2.15539	0.0661367	90.4076	75.3373	107364	123986
Vertical	0.996175	0.222138	6.10036e-05	0.778807	0.0708793	151.57	397	231106	132876

Beta* squeeze

Yellow ring (design + trim)

The second attempt, a single shot.

RhicLiveOrbitCorrector: the first sequel of the OrbitCalc library

OrbitCalc Library

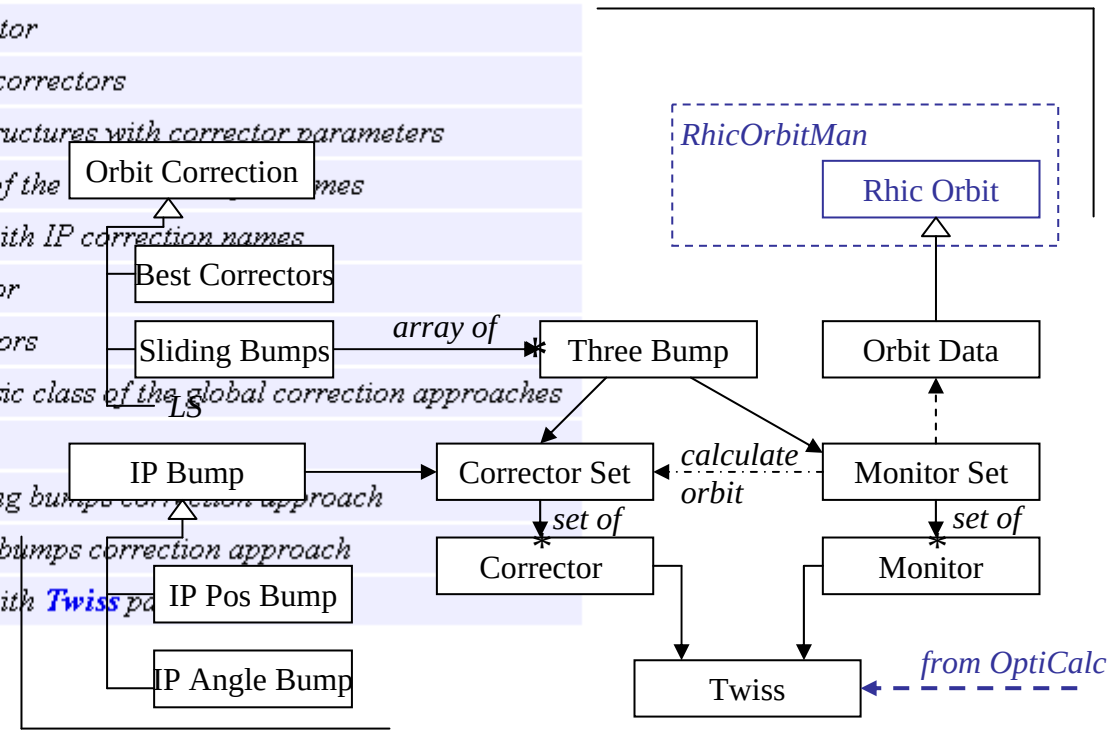
OrbitCalc is a collection of closed orbit algorithms encapsulated from the RhicOrbitDisplay application for reusing and extending by different applications (Sequencer, etc.)

Description :

Here are the classes, structs, unions and interfaces with brief descriptions:

OrbitCalc::Corrector	*Orbit corrector*
OrbitCalc::CorrectorSet	*Set of orbit correctors*
OrbitCalc::CorrStruct	*Vector of structures with corrector parameters*
OrbitCalc::IPBump	*Basic class of the* [illegible] *mes*
OrbitCalc::IPCorrectionNames	*Container with IP correction names*
OrbitCalc::Monitor	*Orbit monitor*
OrbitCalc::MonitorSet	*Set of monitors*
OrbitCalc::OrbitCorrection	*Abstract basic class of the global correction approaches*
OrbitCalc::OrbitData	*Orbit*
OrbitCalc::SlidingBumps	*Global sliding bu*[illegible]*proach*
OrbitCalc::ThreeBump	*Local three-bumps correction approach*
OrbitCalc::Twiss	*Container with* ***Twiss*** *pa*

Structure



RhicLiveOrbitCorrector: the first sequel of the OrbitCalc library

RhicLiveOrbitCorrector (1)

```
1. std::cout << "Get monitor data " << std::endl;
OrbitCalc::OrbitData orbitData(inFileName.c_str());
OrbitCalc::MonitorSet* _Xmonitors = new OrbitCalc::MonitorSet(orbitData, true, OrbitCalc::X);
OrbitCalc::MonitorSet* _Ymonitors = new OrbitCalc::MonitorSet(orbitData, true, OrbitCalc::Y);
```

```
2. std::cout << " Get initial corrector strengths " << std::endl;
corrIn.Read(inFile, OrbitCalc::X);
_RM.GetCorrectorStrength(corrIn);
OrbitCalc::CorrectorSet* _Xcorrectors = new OrbitCalc::CorrectorSet(corrIn, false, OrbitCalc::X);
corrIn.Read(inFile, OrbitCalc::Y);
_RM.GetCorrectorStrength(corrIn);
OrbitCalc::CorrectorSet* _Ycorrectors = new OrbitCalc::CorrectorSet(corrIn, false, OrbitCalc::Y);
```

```
3. std::cout << “Load optics “ << std::endl;
_RM.GetTunes(nu_x, nu_y, machineType);
_RM.GetTwiss(twissTmp, _nTw, machineType);
_Xmonitors->LoadTwiss(_twiss, _nTw);
_Ymonitors->LoadTwiss(_twiss, _nTw);
_Xcorrectors->LoadTwiss(_twiss, _nTw);
_Ycorrectors->LoadTwiss(_twiss, _nTw);
```

RhicLiveOrbitCorrector: the first sequel of the OrbitCalc library

RhicLiveOrbitCorrector (2)

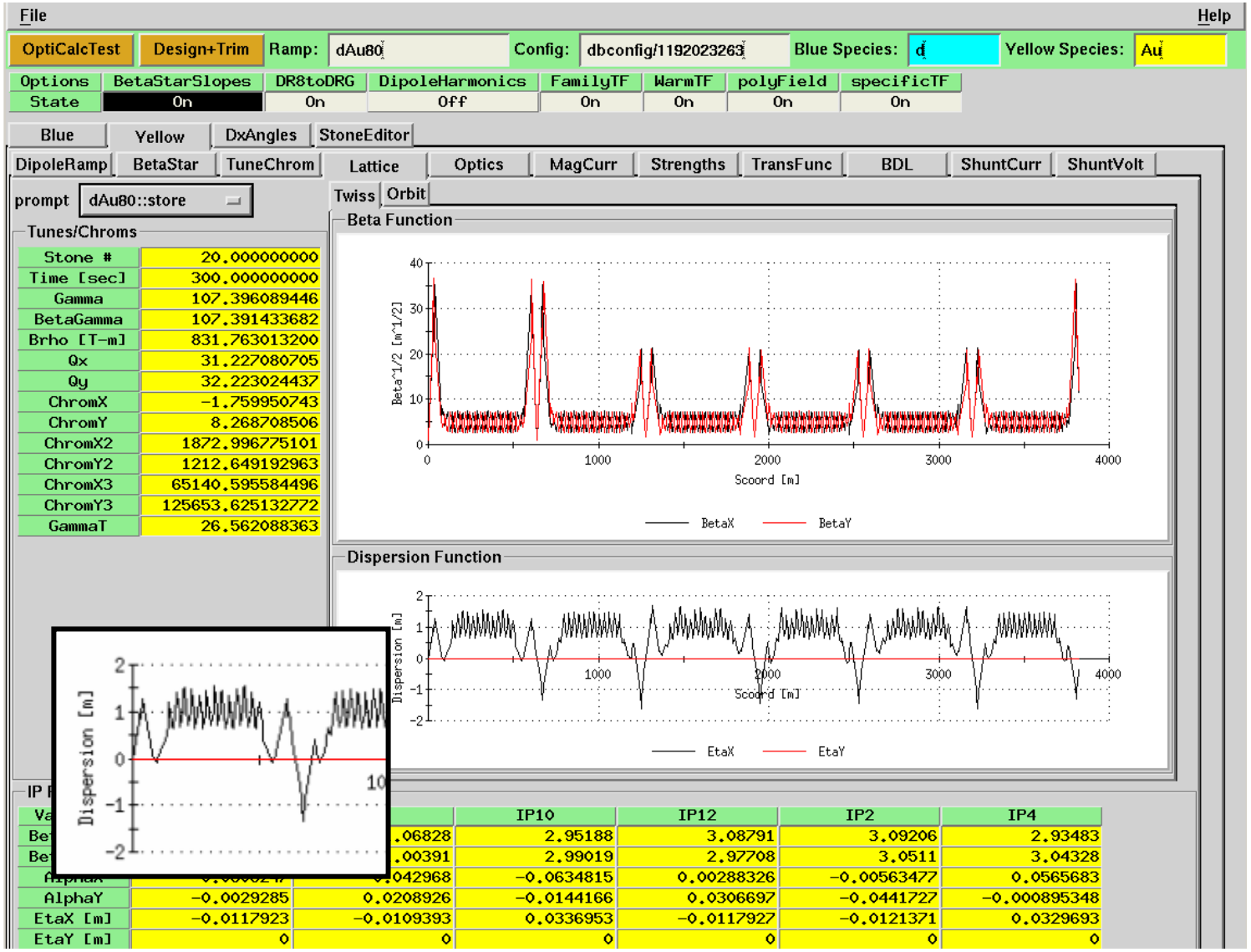
```
4. std::cout << "Get target orbit " << std::endl;
 _Xmonitors->LoadOrbit(filename,'g');
 _Ymonitors->LoadOrbit(filename,'g');
```

```
5. std::cout << "Apply Best Correctors " << std::endl;
 OrbitCalc::BestCorrectors bcorr(_bestCorrN);
 bcorr.LoadSettings(_Xcorrectors, _Xmonitors, OrbitCalc::TRIM_TYPE, _muTotX);
 bcorr.MakeCorrection(0., 4000., _withDpp);
 bcorr.LoadSettings(_Ycorrectors, _Ymonitors, OrbitCalc::TRIM_TYPE, _muTotY);
 bcorr.MakeCorrection(0., 4000., _withDpp);
```

```
6. std::cout << “Construct a new orbit” << std::endl;
RhicOrbit newOrbit = OrbitCalc::ConstructRhicOrbit(*_Xmonitors, *_Ymonitors);
```

```
7. std::cout << “send corrector data” << std::endl;
OrbitCalc::CorrStruct corrFin;
corrFin.GetData(*_Xcorrectors);
_RM.PutCorrectorStrength(corrFin);
 corrFin.GetData(*_Ycorrectors);
_RM.PutCorrectorStrength(corrFin);
```

Resolving the dispersion issue in the d-Au online model



Next Step

- **Further consolidation of RampDesigner, OptiCalc online model, and Levenberg-Marquart optimization module towards the following applications (in order of their complexity):**
  - Adding the **robust** tune fitting procedure for working at the near-interger working point
  - Automation of gamma-transition setting
  - Individual optimization of the important stones, such as injection, gamma transition, store.
  - Quarter/Semi/Automation of the ramp design procedure