# RMS ENVELOPE BACK-PROPAGATION IN THE XAL ONLINE MODEL*

C.K. Allen[#], ORNL, Oak Ridge, TN 37831, U.S.A.
H. Sako, JAEA, Tokai-mura, Naka-gun, Ibaraki 319-1195, Japan
M. Ikegami, KEK, Tsukuba-shi, Ibaraki 305-0801, Japan

*Abstract*

The ability to back-propagate RMS envelopes was added to the J-PARC XAL online model. Specifically, given an arbitrary downstream location, the online model can propagate the RMS envelopes backward to an arbitrary upstream location. This feature provides support for algorithms estimating upstream conditions from downstream data. The upgrade required significant refactoring, which we outline. We also show simulations using the new feature.

## INTRODUCTION

XAL is a Java-based (object-oriented) development environment for high-level accelerator applications initially developed at the Spallation Neutron Source. There are numerous references on XAL in the literature [1], as well as online documentation [2][3]. The basic function is to provide a dynamic, device-oriented interface to the accelerator hardware, however, XAL also provides a numerous tools and support packages.

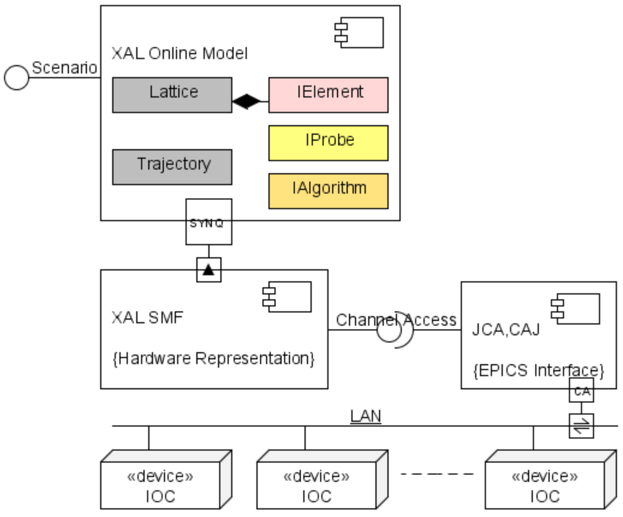

Figure 1: XAL with online model.

One such tool is an online simulation engine called the *XAL online model.* Figure 1 is a UML schematic of XAL emphasizing the online model. The model has been previously described both theoretically [4] and architecturally [5]. Many upgrades and customizations have been made to the online model deployed at the J-PARC facility [6]. Here we focus on the RMS envelope back-propagation upgrade implemented there. Since the upgrade required signification modification to the model architecture, we outline the design of the online model.

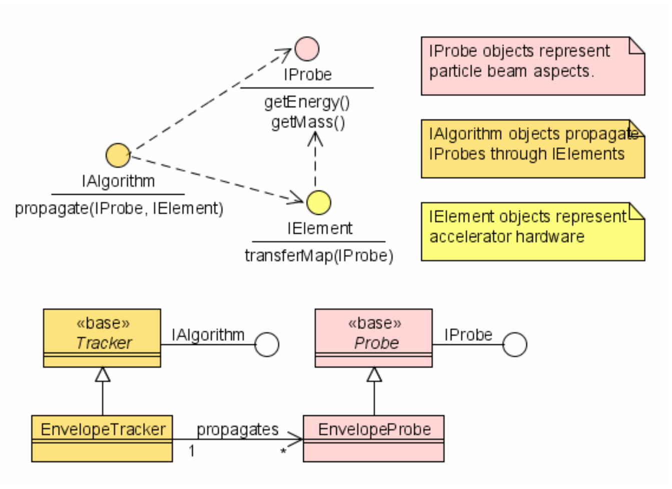
IProbe
getEnergy()
getMass()
IAlgorithm
propagate(IProbe, IElement)
IElement
transferMap(IProbe)
IProbe objects represent particle beam aspects.
IAlgorithm objects propagate IProbes through IElements
IElement objects represent accelerator hardware
«base» Tracker
IAlgorithm
«base» Probe
IProbe
EnvelopeTracker
propagates
1
*
EnvelopeProbe

Figure 2: Online model architecture.

### *XAL Online Model*

The XAL online model is a real-time beam simulation engine presenting a simple programming interface for high-level physics and control applications. In addition, the same interface is used for different simulation categories (e.g. single-particle, multi-particle, envelope, response matrix, etc.). Figure 1 depicts this situation, the entire simulation engine is encapsulated as a `Scenario` object. From an engineering aspect, we have a software component that is easily maintained and upgraded to support the varying requirements of accelerator commissioning and operation.

The online model is based upon a special software architecture called the Element-Algorithm-Probe design pattern introduced by Malitsky and Talman [6]. The basic architecture is shown in the UML schematic of Figure 2 (encapsulated in Figure 1). The design strategy decouples the machine model from beam representation, and from dynamics calculations; thus, a variety of simulation capabilities can be realized with a single simulation engine. Shown in Figure 2 are two abstract classes, `Tracker` and `Probe`, that expose the `IAlgorithm` and `IProbe` interfaces, respectively. These two base classes handle most boilerplate functionality required of the interfaces. Also shown are the two derived classes `EnvelopeTracker` and `EnvelopeProbe` that support RMS envelope simulation. We highlight these classes since they required significant refactoring.

## BACK PROPAGATION

The back-propagation feature was implemented as a separate mechanism within the XAL online model, following a separate execution thread than that of forward propagation. This strategy was taken to avoid side-effects,

*Work supported by KEK under a foreign visiting scientist grant.
[#]allenck@ornl.gov

increase code readability, and to minimize any obfuscation to the software developer. Even so, significant refactoring was required to maintain compatibility with the existing simulation features, and to avoid code duplication.

Because of space charge and RF gap effects, implementing back propagation requires considerable more effort than simple matrix inversion. To support the upgrade in a robust fashion, significant refactoring of the underlying architecture was required. A major modification is the `Tracker` class hierarchy. Also significant is the addition of the method `backPropagate()` in the `Scenario` model interface. That modification forces a cascade of similar modifications in order to conform to the architecture. In addition, the technique for computing space charge effects required modification.

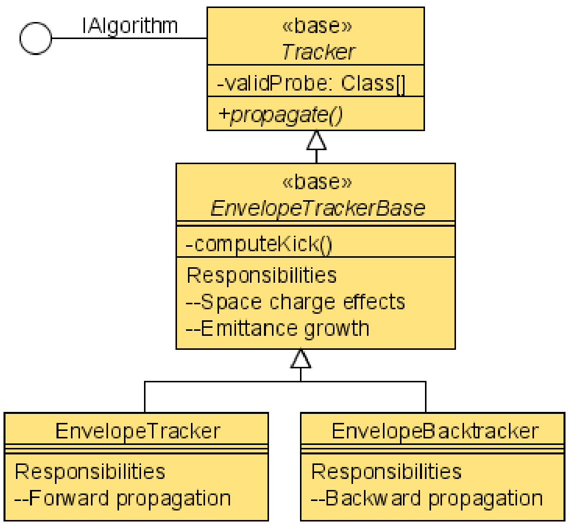

Figure 3: Algorithm hierarchy refactored.

## *Tracker Hierarchy Refactoring*

The algorithm hierarchy is founded on the `Tracker` base class, the class hierarchy before back propagation shown in Figure 2. A schematic of the refactored hierarchy is shown in the UML diagram of Figure 3. To avoid duplicate code, much of the functionality previously in `EnvelopeTracker` was pulled up into the new parent class `EnvelopeTrackerBase`. This functionality includes the space charge effects and emittance growth. Additionally, the method `retractProbe()` was added to the `Tracker` base class to back-propagate all the common beam probe attributes (i.e., those found in the `Probe` base class). The method is the functional complement of method `advanceProbe()`, which forward propagates the common attributes.

At the bottom of the algorithm hierarchy are the two concrete classes `EnvelopeTracker` and `EnvelopeBacktracker`. These algorithms both operate on `EnvelopeProbe` objects (the RMS envelope representation); one forward propagating and one backward propagating, respectively. All the technical calculations are handled in the heavy-weight parent class `EnvelopeTrackerBase`. The two children are light-weight classes that perform the logistics of forward or backward propagation. With this design we avoid code duplication, consolidating the expensive, high-risk code in a central location.

Finally, we note that there exists an additional class, `EnvTrackerAdapt`, in the `Tracker` hierarchy not shown in Figure 3. This algorithm class uses an adaptive integration algorithm for forward propagation of RMS envelopes. This algorithm is not often used at the J-PARC site because the modelling of permanent magnet quadrupoles requires equal integration steps. However, the refactoring was also extended to this class.

## *Space Charge Mechanism*

The technique for computing transfer matrices with space charge was changed in order to accommodate the back-propagation mechanism. Previously the calculation was coupled with control flow; that was more efficient but functioned correctly only for forward propagation. Specifically, given a beamline element *n*, denote by $\mathbf{\Phi}_n(l)$ the transfer matrix which propagates beam particle coordinates the length $l$ down element *n*. At each integration step, with step length $h$, the covariance matrix $\boldsymbol{\sigma}$ (the state object of the RMS envelope class `EnvelopeProbe`) was first advanced using the transfer matrix $\mathbf{\Phi}_n(h/2)$ for the half step *h*/2. The space charge transfer matrix, $\mathbf{\Phi}_{sc}$ was then computed and used to provide the space charge "kick" to $\boldsymbol{\sigma}$. Finally, $\boldsymbol{\sigma}$ is advanced the final distance *h*/2 by matrix $\mathbf{\Phi}_n(h/2)$ to complete the integration over length *h*.

In the new implementation we explicitly compute the transfer matrix $\mathbf{\Phi}(h)$ for the full integration step. At each element *n* we first propagate $\boldsymbol{\sigma}$, forward or backward, by $\mathbf{\Phi}_n(h/2)$ for the specific purpose of computing the space charge matrix $\mathbf{\Phi}_{sc}$. Once $\mathbf{\Phi}_{sc}$ is computed, $\boldsymbol{\sigma}$ is rolled back to its original state. Then the full transfer matrix, including space charge, is formed as $\mathbf{\Phi}(h) = \mathbf{\Phi}_n(h/2)\mathbf{\Phi}_{sc}\mathbf{\Phi}_n(h/2)$, which is then used to propagate $\boldsymbol{\sigma}$ over length *h*. Although this implementation is slightly more expensive (it has one additional $6 \times 6$ matrix multiplication per integration step), it is more general and allows for the direct computation of the back propagation matrix, $\mathbf{\Phi}_n(-h)$, including space charge. This procedure also supports the adaptive integration algorithm used in the class `EnvTrackerAdapt`.

## SIMULATION RESULTS

Figure 4 shows RMS envelope back-propagation simulation results. Also plotted in the charts are forward simulation results from the online model and from Trace3D; this is a single curve with legend label XAL-T3D. (The forward propagated results of XAL and Trace3D are identical and seen as a single trace.) Also shown are particle-in-cell forward simulations results from the IMPACT code; this curve has label IMPACT.

The IMPACT curve is included for benchmark comparison. The beamline under simulation is the MEBT1-S03B line of the J-PARC linac. All simulations use the same beam parameters, except for initializing Twiss parameters. Forward propagation simulations are initialized with the design Twiss parameters at the beamline entrance. The back-propagation simulation is initialized with the design Twiss parameters at the exit of the beamline. The plots are an interesting method of displaying the discrepancy between "what you have" and "what you want." When the forward propagation Twiss parameters are used to initialize back-propagation, the results are identical and, thus, not shown.

## SUMMARY

RMS envelope back propagation was implemented and tested in the XAL online model at J-PARC. Concerning this upgrade we make a couple practical remarks. Back propagation was implemented to support the online model phase-slip calculation, however the feature is, as of yet, untested. Emittance growth can be simulated when back-propagating, however, only the Trace3D emittance growth model appears to be stable over longer distances; apparently a numerical issue considering that the emittance growth mechanism is nonlinear and back propagating involves matrix subtraction here.

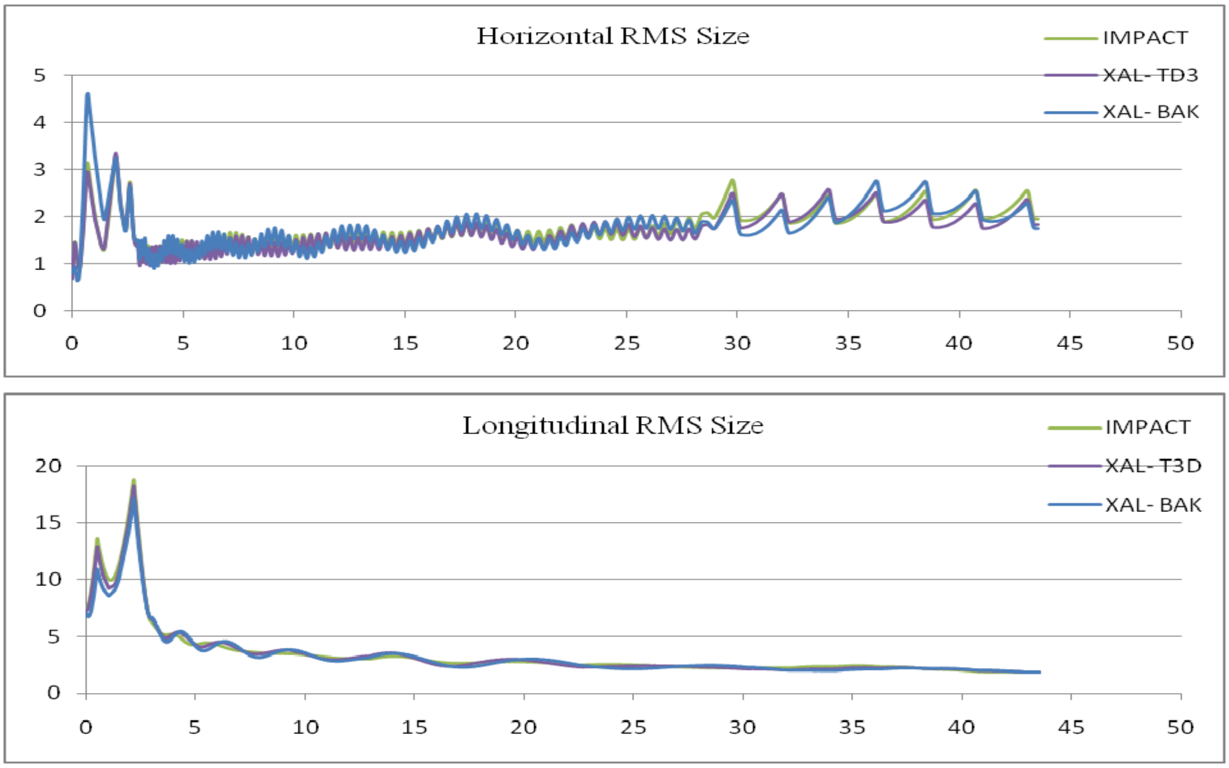

Figure 4: Simulations showing forward and backward propagated RMS envelopes along with an IMPACT comparison.

## REFERENCES

[1] J. D. Galambos, C.P. Chu, S.M. Cousineau, T.A. Pelaia, A.P. Shishlo, C.K. Allen, "XAL Application Programming Structure", PAC05 Conference Proceedings, Oak Ridge, Tennessee , May 16-20, 2005, pp. 79-83.

[2] http://www.ornl.gov/~t6p/Main/XAL.html.

[3] https://wiki.ornl.gov/sites/xaldocs/PHYS798X%20Course%20Material/Forms/AllItems.aspx.

[4] C.K. Allen, C.A. McChesney, N.D. Pattengale, C.P. Chu, J.D. Galambos, W.-D. Klotz, T.A. Pelaia, A. Shislo, "A Modular On-Line Simulator for Model Reference Control of Charged Particle Beams", PAC 2003 Conference Proceedings, Portland, OR, May 12-16, 2003.

[5] C.K. Allen, C.A. McChesney, C.P. Chu, J.D. Galambos, W.-D. Klotz, T.A. Pelaia, A. Shislo, "A Novel Online Simulator for Applications Requiring a Model Reference", ICALEPCS 2003 Conference Proceedings, Kyongju, Korea, October 13-17, 2003.

[6] C. K. Allen, H. Ikeda, M. Ikegami, T. Ohkawa, H. Sako, and G. B. Shen, "XAL Online Model Enhancements for J-PARC Commissioning and Operation", MOPAN029, PAC 2007, Albuquerque, USA, pp 218.

[7] N. Malitsky and R. Talman, "The Framework of the Unified Accelerator Libraries", ICAP 1998.