ARCHITECTURAL IMPROVEMENTS AND NEW PROCESSING TOOLS FOR THE OPEN XAL ONLINE MODEL

C.K. Allen, T.A. Pelaia II, ORNL, Oak Ridge, Tennessee 37831 USA
J. Freed, University of South Carolina, South Carolina 29208 USA

Abstract

The online model is the component of Open XAL providing accelerator modeling, simulation, and dynamic synchronization to live hardware. Significant architectural changes and feature additions have been recently made in two separate areas: 1) the managing and processing of simulation data, and 2) the modeling of RF cavities. Simulation data and data processing have been completely decoupled. A single class manages all simulation data while standard tools were developed for processing the simulation results. RF accelerating cavities are now modeled as composite structures where parameter and dynamics computations are distributed. The beam and hardware models both maintain their relative phase information, which allows for dynamic phase slip and elapsed time computation.

BACKGROUND

Open XAL is an open source development environment used for creating accelerator physics applications, scripts, and services [1]. The project has seen collaboration among SNS, CSNS, ESS, GANIL, TRIUMF and FRIB. Open XAL was born out of XAL, an application framework originally developed for the SNS in the early 2000s [2]. Open XAL is configurable for multi-site operation and the project facilitates multi-institutional development. It was initially released at the end of 2010 and is currently working toward its 6th milestone. For a status report on Open XAL see Pelaia [3]. For information on using Open XAL see the USPAS 2014 course Control Room Accelerator Physics material [4]. For material on the architecture of Open XAL see [5].

One significant component of Open XAL is the online model. This paper is concerned with two recent upgrades to the online model, data processing and the RF cavity model. For a perspective of these upgrades consider the overall architecture of online model; it is built according to the Element/Algorithm/Probe software architecture of Malitsky and Talman [6], which offers a robust implementation strategy for accelerator system modeling.

SIMULATION DATA AND ANALYSIS

The representation, storage, and analysis of online model simulation data are significantly improved. A new architecture for data handling was implemented which separates data maintenance and data analysis. This refactoring reduced code support by ten classes.

Figure 1: Open XAL online simulator architecture.
Simulation Data

Simulation data now consists of a series of probe state objects containing beam state information at locations along the design path. Class Trajectory manages these state objects providing bookkeeping, sorting, and retrieval. At the heart of the simulation data architecture is the abstract base class ProbeState, shown in Figure 3. Properties specific to the simulation type are contained in derived classes, a particle simulation contains a phase vector (ParticleProbe), a transfer map computation contains a phase map (TransferMapState), and an envelope simulation contains a covariance matrix (EnvelopeProbeState). The Trajectory container has a template parameter \( \tau \) identifying its content type. The trajectory typing is also used by the analysis package to identify available processing tools for the data type. During simulation the Probe class records its history through the beamline as a trajectory object.

Data Analysis

Simulation data is processed using tools from the new package xal.tools.beam.calc. Currently there are four concrete classes for data processing, all derived from a common base CalculationEngine. Classes derived from CalculationEngine all accept Trajectory data, but only the proper type. The restriction is enforced through binding with template \( T \) in Trajectory<T>. Figure 2 shows that child classes CalculationsOnMachines and CalculationsOnRings can both process transfer map data; data that involves only machine operation without regard to beam dynamics. CalculationsOnParticles and CalculationsOnBeams support the data processing from beam simulations, specifically single particle and envelope simulation, respectively.

To support uniform treatment of processed simulation results the processing classes may expose one or both of the nested ISimulationResults interfaces. However, although the method signatures may be the same, the results must be interpreted within the context of the data. For example, “Twiss parameters” for a ring do not have the same interpretation as that for a beam envelope.

ACCELERATION MODEL

An RF cavity can be modeled as a cascade of RF gaps separated by drift spaces. This seems straightforward but in practice a computer model may require some advanced architecture. A good software design is one where the cavity is captured through component parts that function in concert through well-defined interfaces. This subject is explored more fully below. For now consider the necessary requirements for modeling the RF cavity motivated by beam dynamics.

Externally, an accelerating cavity has two variables, the phase \( \phi_{\text{RF}} \) and amplitude \( V \) of the RF power entering the cavity. The frequency of the RF is essentially fixed by the cavity design. The amplitude and phase of an internal gap must have some coupling mechanism to the cavity phase and amplitude; the architecture must have some method of distributing this coupling. To do so it is necessary to determine the relationship between cells, cavities, and drifts and their phasing and the distances between as well as the particle phasing; a significant amount of bookkeeping. A quick outline of RF cavity models and dynamics helps to drive the software design.

Figure 4 depicts a simple dynamics model for the particle phase and energy while it propagates through an RF cavity. The particle has phase \( \phi_0 \) (at that moment) and energy \( W_0 \) when it enters the cavity. As the particle propagates through the first drift its phase \( \phi(z) \) advances linearly according to \( \phi(z) = k_0 z + \phi_0 \) where wave number \( k_0 \) is determined by the energy \( W_0 \). At \( z = z_1 \) it encounters the first gap where the phase jump \( \Delta \phi_1 \) and energy gain \( \Delta W_1 \) are computed. Its phase and energy are then updated. If there are other dynamic variables they too must be advanced. The process continues throughout the cavity until the particle has exited.

Now consider the cavity model. Most RF cavities consist of a series of “cells” connected with some type of RF coupling. Injected RF power \( V \) is distributed among...
the internal cells, which produce the gap dynamics. The cavity can operate in one of many modes. Each mode $q$ has a different field distribution (and potentially different operating frequency). Simulation codes usually associate the cell phasing as a property of the particle phase $\phi$. However, we advocate assigning phase to the cell fields directly, and we have architecture to support it. Index each cell with an integer $m$ and consider the cell $m$ axial electric field magnitude $E_m$, we have

$$E_m = \alpha_m V \cos(q m \pi + \phi_{cav}), \quad m = 0, 1, 2, ...$$

(1)

where $\alpha_m$ is the coupling constant between the cavity input power $V$ and cell $m$ field strength. By assigning field strengths in this manner the beam particles experience the correct gap effects and maintain their correct phase $\phi(z)$ with respect to the cavity input phase. Figure 5 shows a software architecture supporting this approach. The cavity cells and the beam particles each maintain their own phases relative to the cavity phase.

Consider more closely the RF cavity software model of Figure 5. See that `IdealRfCavityCell` is a composed of one `IdealRfGap` and two `IdealRfCavityDrift` objects. The `IdealRfCavity` is, in turn, a composite of `IdealRfCavityCell` objects. These structures cleanly support the necessary bookkeeping of the above accelerating cavity model, plus it is easy to understand having been taken right from the problem domain. Each cell object knows its field, phasing, and amplitude, which are passed from the parent cavity. The design being from the problem domain affords the dynamics model a robust implementation. The drift-kick-drift dynamics are performed by cell components and supervised by the cavity.

**CONCLUSION**

Significant improvements in the Open XAL online model have been made in the modeling of RF accelerating cavities and the acceleration model in general. The new architecture supports on-the-fly phase slip and elapsed time calculations without requiring separate runs for synchronous phasing information. The calculation of various acceleration parameters is localized to specialized hardware classes while the beam model carries necessary information between them.

The simulation data and data processing component of Open XAL have been likewise improved. The data and operations are now two separate, compartmentalized components. The trajectory contains simulation data and does only maintenance operations, no processing. All processing is done in a separate package and only according to the type of data offered. There are separate tools for machine and ring calculations, and particle and envelope dynamics.

**REFERENCES**


