

Online Model Architecture

AP Video, 4/8/03

- Xal view of the accelerator: Accelerator nodes
- Model view of the accelerator: lattice
- Connection between the two views, and working with the model.

- View the accelerator as a set of “Accelerator Nodes”
 - These nodes correspond to devices in the beamline
 - Node types include quad, corrector, BPM, current monitor,
 - Physical devices with multiple components are separated into multiple “nodes” at the same location
 - E.g. MEBT quad + correctors + BPM = 4 nodes
- These Accelerator nodes are grouped together in sequences
 - MEBT, DTL1, ...CCL, SCL1, ...HEBT
 - Sequences can be “pasted together”

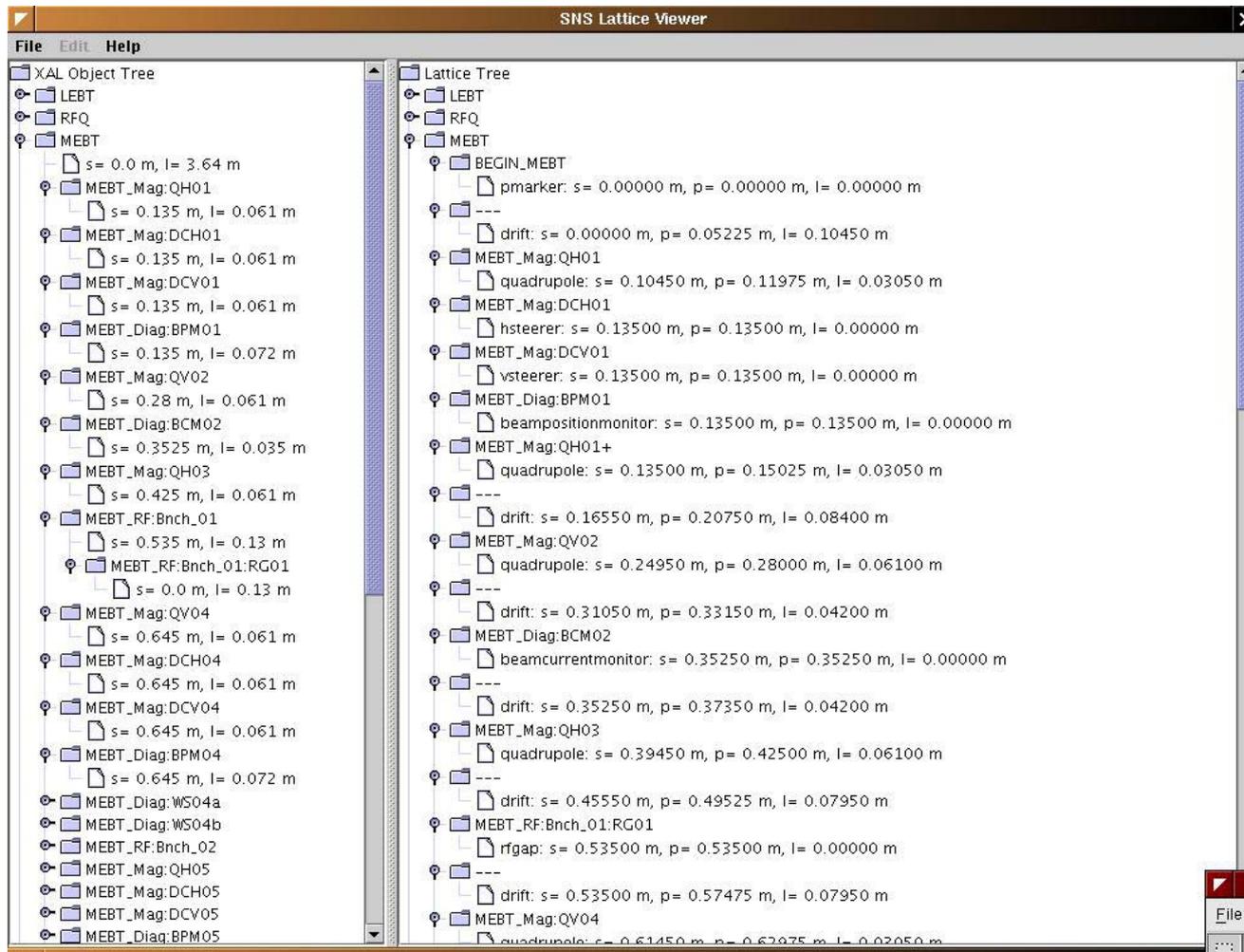
- The information about these nodes is stored in the Global Database
 - We create an xml file with this information in it
 - Only static information is included here
 - Note: no drifts are included in the database – only real devices
 - Populated through HEBT

- “Model” consists of 2 main parts (based on Malitsky design)
 - Probe: the type of beam model, needs initial conditions.
 - ParticleProbe (Working: for centroids, single-particle, etc.)
 - EnvelopeProbe (Working: for Trace3D-type simulations)
 - *EnsembleProbe (In Progress: for multi-particle simulations)*
 - *ParticleRespProbe (first-order perturbations from a single-particle trajectory)*
 - *EnvelopeRespProbe (first-order perturbations from an envelope trajectory)*
 - Elements: the usual set of lattice elements used in lattice transport codes
 - Have drifts, thin lens, quads, dipoles, + rf gaps implemented
- The results (i.e. twiss parameters + particle coordinates) are stored in the probe as a function of longitudinal position along the elements

- Lattice generator is the intermediate step
 - Input a sequence of accelerator nodes, output a list of elements
 - Split the quads, correctors are not split, diagnostics are “markers”
 - Each lattice element includes a reference to its parent Accelerator Node
 - Have a map indicating which element represents middle of the node
- From the lattice generator file:
 - Make a set of model elements
 - Make external lattice files
 - Requires synchronization with the machine to get live values

Lattice Generation

- Accelerator Node to Lattice Element mapping



- Need to hide much of the details with convenience methods
- Details are being worked out but the steps include:
 - Start with Accelerator Sequence
 - Generate lattice using the lattice generator
 - `Lattice lat = LatticeFactory.getLattice(seq);`
 - Includes a map from Accelerator Node to model element(s)
 - Set up probe
 - `IProbe probe = ProbeParser.parseUrl(strFileName);`
 - Launch the probe through the elements
 - `lat.propagate(probe);`
 - Retrieve info from the probe using the “lattice map”
 - `probe.getTwiss(node1);`
 - Higher level model manipulations will involve propagating probes within a solver framework
 - E.g. tune