Validation of Fast Simulators
Some Thoughts

Harrison B. Prosper

Florida State University

Workshop on Fast Simulators for the LHC

CERN

11 – 12 June, 2012

Automated Validation

Fast simulators are typically driven by parameters that can be tweaked in order to improve the fidelity of the simulation.

The tweaking is usually done by hand. However, there is no technical impediment to automating this process.

We suggest a couple of ways this could be done.

Automated Validation – 2

What is needed?

1. A measure of discrepancy between the distributions produced by the fast-sim and full-sim.
2. A robust method to minimize the discrepancy measure.
3. A method that can account for correlations.

In the following we describe two methods that have been briefly explored by Prosper and Sekmen.

Using 1-Dimensional Binning

The standard method for checking the similarity of two N-dimensional distributions in is to compare several 1-D histograms.

We can generalize this with a method called *Hedgehog*, which proceeds follows:

1. Generate a random set of rays passing through the origin of the N-dimensional space of observables.
2. Project the points onto each ray and histogram the values along each ray.
3. Define the discrepancy between the outputs of the simulators as the worst discrepancy among the histograms, of which there could be thousands.

Using 1-Dimensional Binning – 2

The intuition behind the Hedgehog method is that the more randomly chosen histograms that are tested, and that pass the validation criterion – for example, a standard test of discrepancy between 1-D histograms – the more probable the statement that the two N-dimensional distributions are the same.

In effect we perform “tomography” using the rays of histograms.

Using N-Dimensional Binning

We could try to bin the N-dimensional space of observables directly, in a way that avoids the "curse of dimensionality", and thereby reduce the data to a sequence of counts. There are at least two ways to do this:

1. Use N-dimensional Voronoi tessellation
2. Use recursive binary partitioning (k-d tree)

Root contains a class TKDTreeBinning that we have used in a program called *Turtle* to bin data in N-dimensions in such a way that the counts per bin are approximately the same.

Implementing the Validation

Given a discrepancy measure, validation entails running a program such as Delphes repeatedly to minimize the discrepancy measure.

Since it is likely that the discrepancy measure is an insufficiently smooth function of the program parameters (because of the finite number of events available for binning), a non-gradient method of minimization such as the simplex would be necessary.

With the computational resources available, implementing these suggestions is perfectly doable.