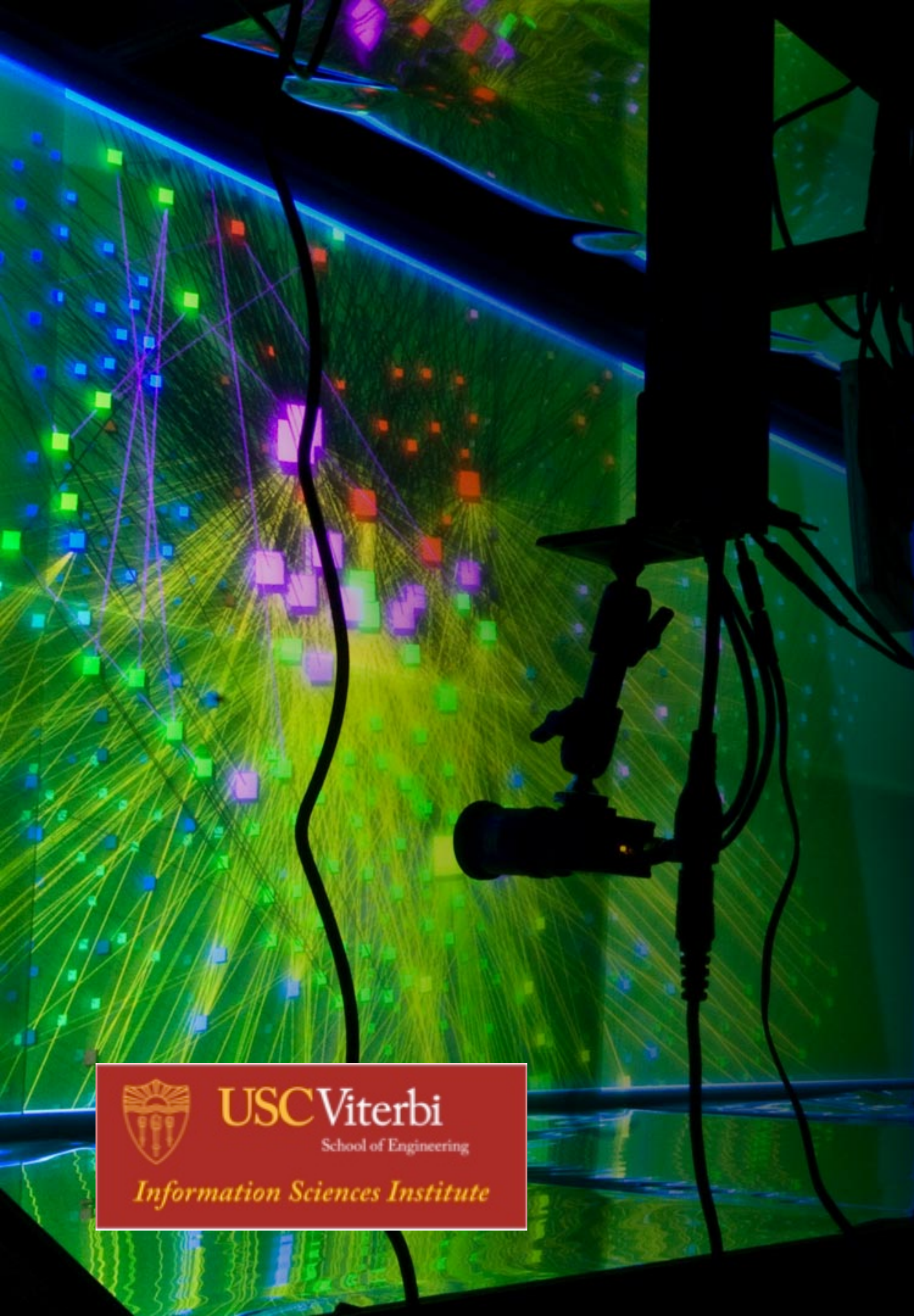


GEANT4 Hadronic Cross Section Optimizations

Robert Fowler and Paul Ruth
RENCI / UNC Chapel Hill

Pedro Diniz
ISI / USC

USC Viterbi School of Engineering
Information Sciences Institute

renci
RESEARCH ENGAGEMENT INNOVATION

Background CrossSection Calculation

*51 real events simulated in ~2 hours (provided by Soon)

- Hadronic CrossSections
  - ~10% of total wall clock time*
  - Deep call chain with no hot spots
    - Reduce call chain length
    - Reduce time spent in calculation

Scope	WALLCLOCK (us).[0,0] (E)	WALLCLOCK (us).[0,0] (I)
Experiment Aggregate Metrics	1.29e+09 100 %	1.29e+09 100 %
main		1.29e+09 100 %
131: G4UImanager::ApplyCommand(char const*)		1.28e+09 99.3%
422: G4UIcommand::DoIt(G4String)		1.28e+09 99.3%
210: G4UIcontrolMessenger::SetNewValue(G4UIcommand*, G4String)		1.28e+09 99.3%
277: G4UImanager::ExecuteMacroFile(char const*)		1.28e+09 99.3%
234: G4UIbatch::SessionStart()		1.28e+09 99.3%
215: G4UIbatch::ExecCommand(G4String const&)		1.28e+09 99.3%
170: G4UImanager::ApplyCommand(char const*)		1.28e+09 99.3%
422: G4UIcommand::DoIt(G4String)		1.28e+09 99.3%
210: G4RunMessenger::SetNewValue(G4UIcommand*, G4String)		1.28e+09 99.3%
287: G4RunManager::BeamOn(int, char const*, int)		1.28e+09 99.3%
155: G4RunManager::DoEventLoop(int, char const*, int)		1.26e+09 97.1%
237: G4RunManager::ProcessOneEvent(int)		1.26e+09 97.1%
264: G4EventManager::DoProcessing(G4Event*)	9.34e+05 0.1%	1.26e+09 97.1%
185: G4TrackingManager::ProcessOneTrack(G4Track*)	2.98e+06 0.2%	1.25e+09 96.6%
125: G4SteppingManager::Stepping()	5.61e+06 0.4%	1.17e+09 90.8%
180: G4VProcess::AlongStepGPIL(G4Track const&, double, double, double&, G4GPILSelection*)	2.27e+05 0.0%	5.64e+08 43.6%
458: G4Transportation::AlongStepGetPhysicalInteractionLength(G4Track const&, double, double, double&, G4GPILSelection*)	9.47e+06 0.7%	5.20e+08 40.2%
321: G4PropagatorInField::ComputeStep(G4FieldTrack&, double, double&, G4VPhysicalVolume*)	1.00e+07 0.8%	3.45e+08 26.7%
286: G4ChordFinder::AdvanceChordLimited(G4FieldTrack&, double, double, CLHEP::Hep3Vector, double)	2.51e+06 0.2%	2.29e+08 17.7%
202: G4MagInt_Driver::AccurateAdvance(G4FieldTrack&, double, double, double)	2.40e+06 0.2%	1.17e+08 9.1%
185: G4ChordFinder::FindNextChord(G4FieldTrack const&, double, G4FieldTrack&, double&, double, double*, CLHEP::He	4.28e+06 0.3%	9.35e+07 7.2%
323: G4MultiLevelLocator::EstimateIntersectionPoint(G4FieldTrack const&, G4FieldTrack const&, CLHEP::Hep3Vector const	2.29e+06 0.2%	1.02e+08 7.9%
210: G4ChordFinder::ApproxCurvePointV(G4FieldTrack const&, G4FieldTrack const&, CLHEP::Hep3Vector const&, double	1.02e+06 0.1%	6.01e+07 4.6%
298: G4VIntersectionLocator::IntersectChord(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&, double&, double	8.82e+05 0.1%	1.52e+07 1.2%

renci

USC Viterbi School of Engineering Information Sciences Institute

Cross Section Usage

- Cross section calculation is used to:
  - Determine (probabilistically) whether an interaction occurs in traversing a particular geometric volume.
  - Then determine reaction and outcomes.
- CrossSectionDataStore instances created for 65 different processes.
  - Each of these uses different models for different energy domains, particles, materials.
  - Data is (usually) represented in sub-classes of G4PhysicsVector.
- We have been working on increasing the performance of the CrossSection calculations.

Two strategies

- Improved caching of CrossSection results
  - 1 cache entry per ***triple*** (process/particle/material)
  - Completed (In the pipeline toward production code)
- Surrogate model for CrossSection calculations
  - Prototype completed
  - Initial results are promising

Caching CrossSection Results

- Currently, there is a 1-entry cache per process for XC calculations.
- Observation
  - There is an interleaving of recent calls to GetCrossSection with the same sets of particle, material, process, and energy.
  - Results in same cross section value
  - True even though energy is a double! (The physics is causing this.)
- Optimization
  - Expanded cache recent the most recent cross section for particle, material, process triple.
- Measurements
  - 17% of calls would benefit from this cache
  - 29% of GetCrossSection cycles are from these calls.
  - ~18k triples total
  - ~3k triples would benefit

Caching CrossSection Results

- Implementation
  - Hashtable per process (i.e. per CrossSectionDataStore)
    - std::unordered_map
  - One cache entry for each particle/material pair
    - Key
      - material
      - particle definition
    - Value
      - particle energy
      - cross section (including xsecelm)

renci

USC Viterbi School of Engineering
Information Sciences Institute

Modified CrossSection Calculation

```
G4double G4CrossSectionDataStore::GetCrossSection(part,mat){
  ...
    entry = process_cache_map[(part,mat)];
    if(entry->energy == part->GetKineticEnergy()){
       xsecelm = entry->xsecelm;
       crossSection = entry->crossSection;
    } else
       //Calculate CrossSection the regular way (including xsecelm)
       ...
       entry->xsecelm = xsecelm;
       entry->crossSection = crossSection;
    }
    return crossSection;
}
```

Caching CrossSection Results

- Performance increase
  - 1.8% reduction in wall clock time (51 real events simulated over 2+ hours)
- Presented at Hadronic working group
  - What is the state of this being put in the production code?

Surrogate Model: XS Usage

In the Hadronic section of the code:

- Particle/Material/Process Triples
  - 50% of cycles in ~10 triples
  - 90% of cycles in ~85 triples
  - Total ~18k triples
- Implementing for tens of triples can utilize fast path for nearly all of the calls.

Surrogate Model

- The cross section of an interaction between a particle and a complex material is (re-computed) on each call.
  - Look up each isotope. Use element and isotope abundance tables to weight the result.
- Typical materials
  - Air, Stainless steel, $PbZO_4$, Cu, Teflon, Circuit boards, …
- Stainless ={ ^{12}C, ^{13}C, (^{14}C), ^{54}Fe, (^{55}Fe), ^{57}Fe,^{58}Fe, ^{60}Fe, (^{50}Cr), ^{52}Cr, ^{53}Cr, ^{54}Cr, ^{55}Mn, ^{58}Ni, ^{60}Ni, ^{61}Ni, ^{62}Ni, ^{64}Ni, ^{28}Si, ^{29}Si, ^{30}Si}
- Previous versions of G4 used "pseudoelements" based on natural abundances. This was judged in adequate for the "what happened" calculations.

Building the Surrogate Model

- Create fast path for CrossSection calculations (offline or in initialization).
  - Identify common (particle, material, process) triples.
    - The number chosen depends on how much extra storage can be used.
  - Create Surrogate Model
    - Over sample using existing physics model.
    - Down sample to a simpler model with bounded error.
      - Evenly spaced sample points in E, ln(E), or log(E)
      - Or piecewise linear with adaptively placed nodes (Douglas-Peucker method).
    - Solve a linear system to adjust Y values to remove bias from the sign of the errors in each interval.
    - Represent the reduced model using existing G4PhysicsVector sub-classes.
- For triples where the fast path exists, use it for the mean free path (interaction length) calculations.
  - Single query of a G4PhysicsVector
- When an interaction occurs (rare)
  - Perform original physics calculations

Building the Surrogate Model

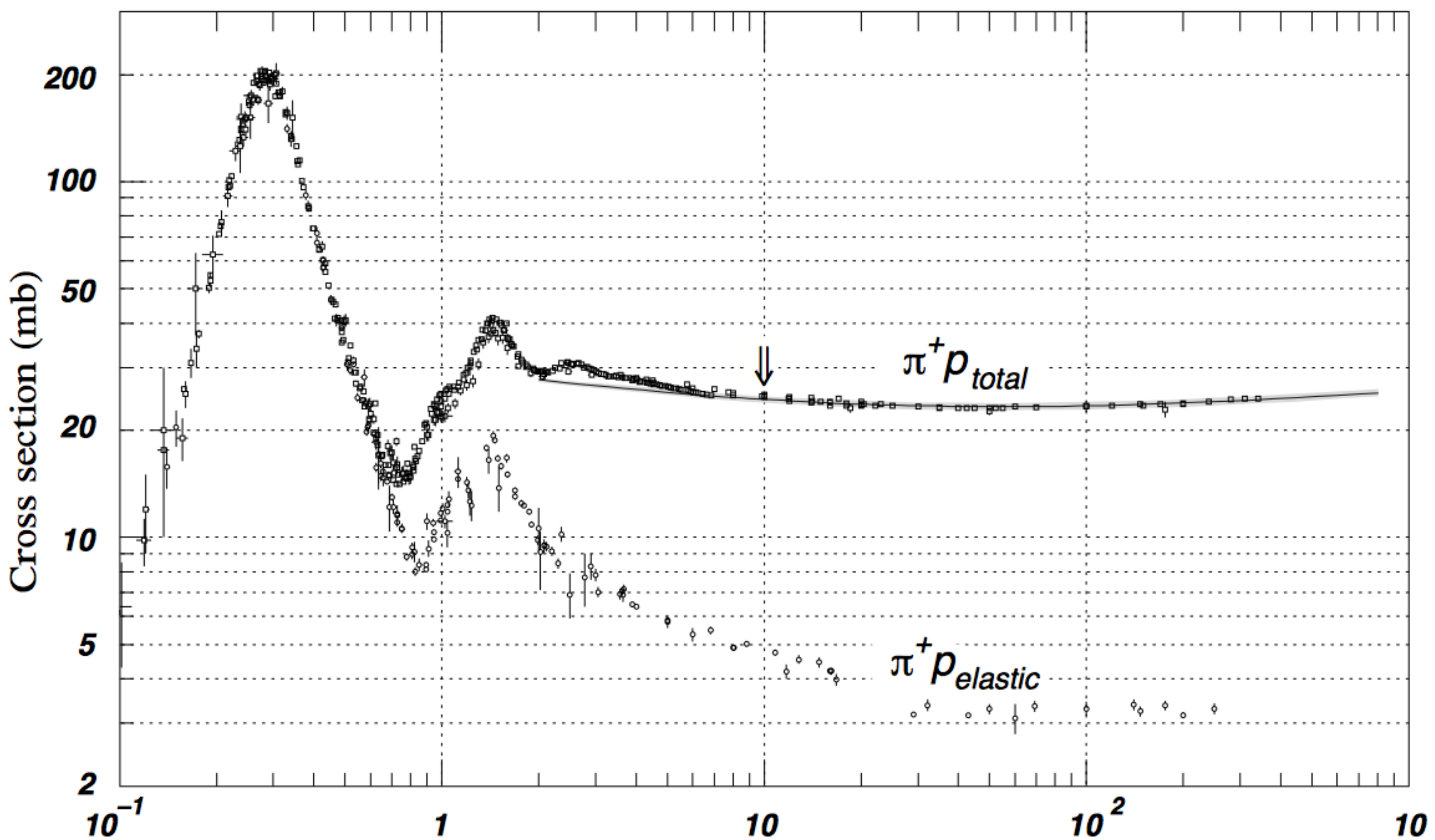
- Create fast path for CrossSection calculations (offline or in initialization).
  - Identify common (particle, material, process) triples.
    - The number chosen depends on how much extra storage can be used.
  - Create Surrogate Model
    - Over sample using existing physics model.
    - Down sample to a simpler model with bounded error.
      - Evenly spaced sample points in E, ln(E), or log(E)
      - Or piecewise linear with adaptively placed nodes (Douglas-Peucker method).
    - Solve a linear system to adjust Y values to remove bias from the sign of the errors in each interval.
    - Represent the reduced model using existing G4PhysicsVector sub-classes.
- For triples where the fast path exists, use it for the mean free path (interaction length) calculations.
  - Single query of a G4PhysicsVector
- When an interaction occurs (rare)
  - Perform original physics calculations

Building the Surrogate Model

- Create fast path for CrossSection calculations (offline or in initialization).
  - Identify common (particle, material, process) triples.
    - The number chosen depends on how much extra storage can be used.
  - Create Surrogate Model
    - Over sample using existing physics model.
    - Down sample to a simpler model with bounded error.
      - Evenly spaced sample points in E, ln(E), or log(E)
      - Or piecewise linear with adaptively placed nodes (Douglas-Peucker method).
    - Solve a linear system to adjust Y values to remove bias from the sign of the errors in each interval.
    - Represent the reduced model using existing G4PhysicsVector sub-classes.
- For triples where the fast path exists, use it for the mean free path (interaction length) calculations.
  - Single query of a G4PhysicsVector
- When an interaction occurs (rare)
  - Perform original physics calculations

Building the Surrogate Model

Over sample the existing code.

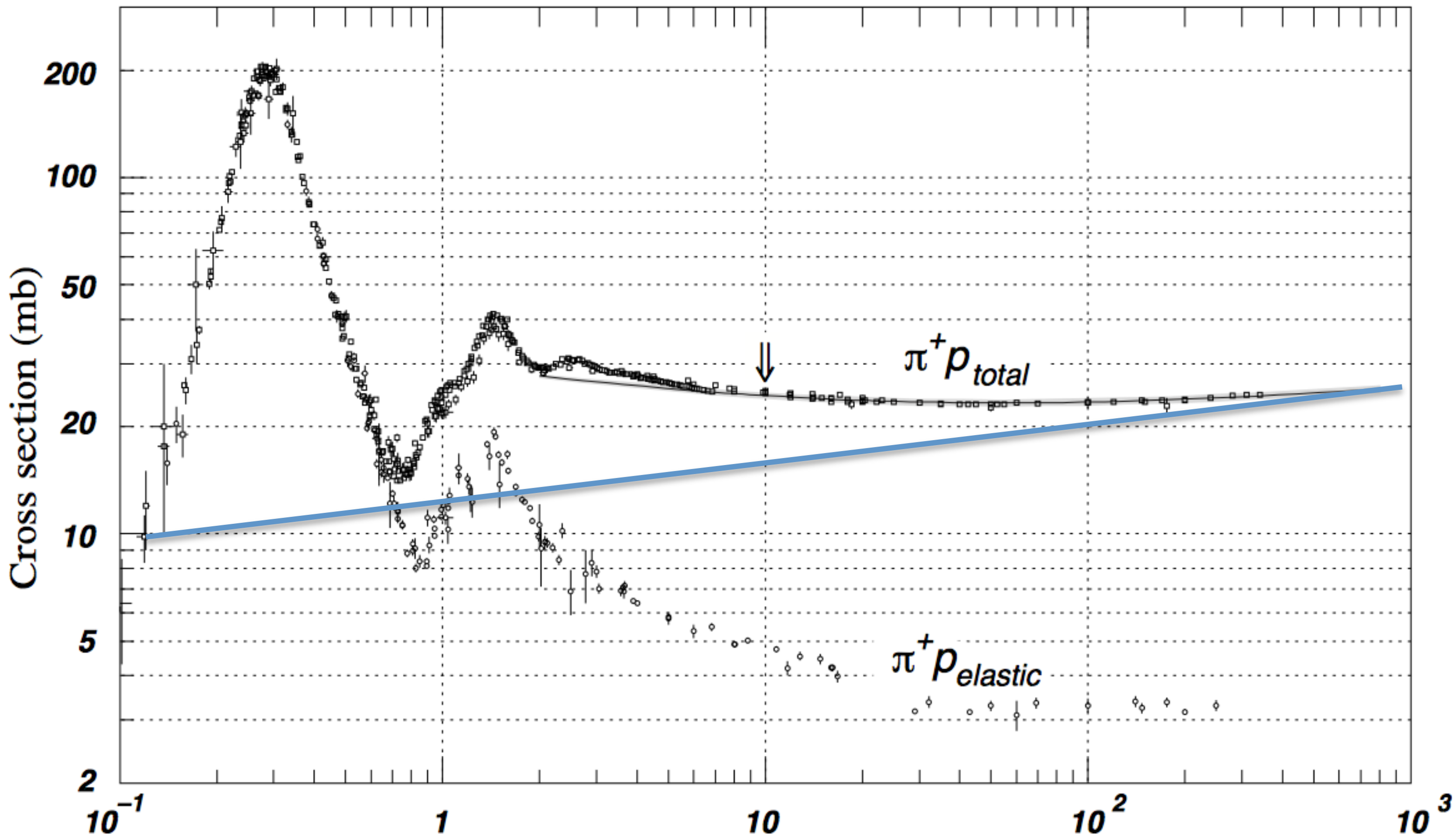


renci

USC Viterbi School of Engineering Information Sciences Institute

Douglas-Peucker Approximation.

Start with a piecewise-linear approximation.
In each segment, Add the point at the largest error.
Until total error is within the required bound.

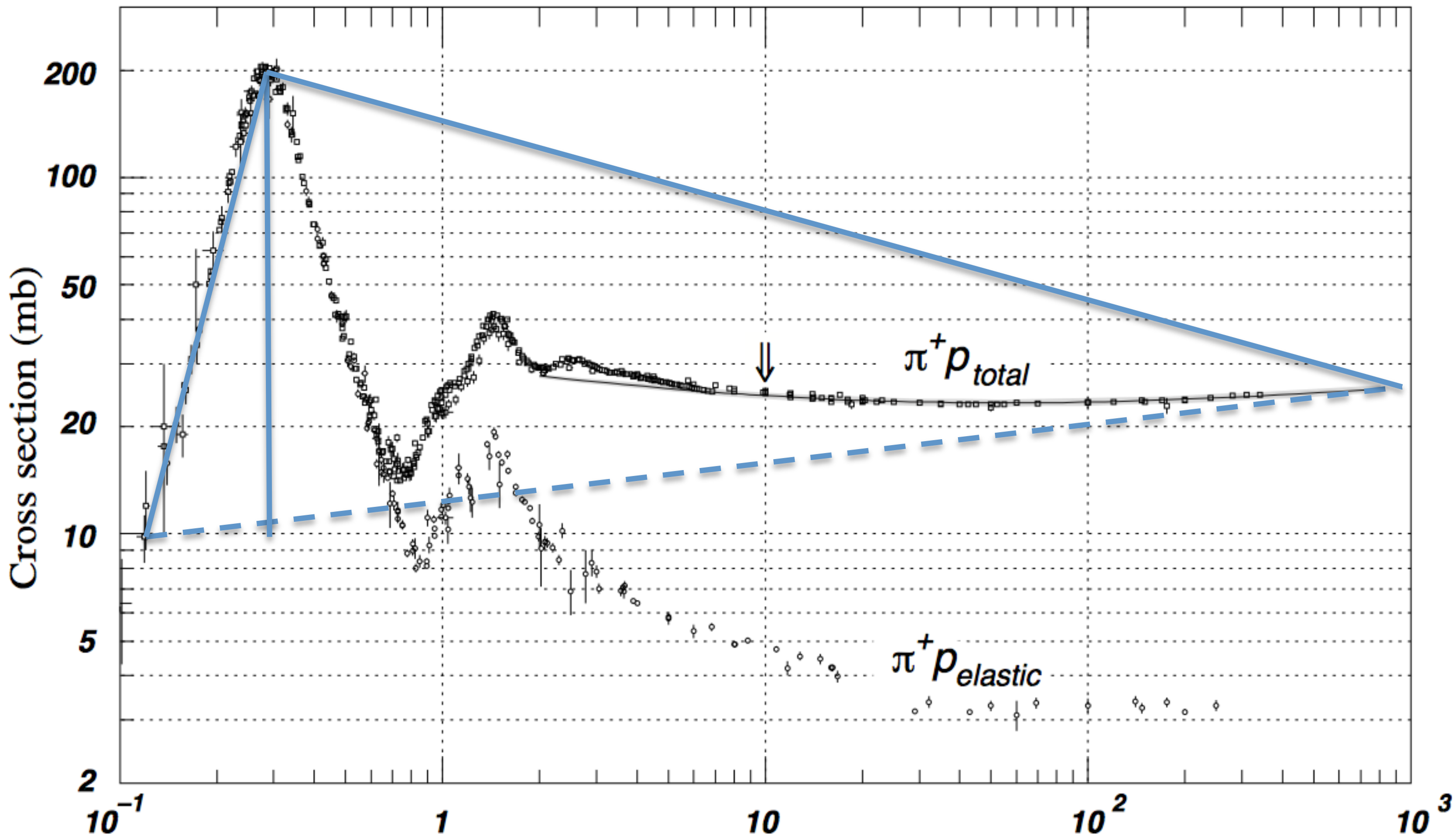


renci

USC Viterbi School of Engineering
Information Sciences Institute

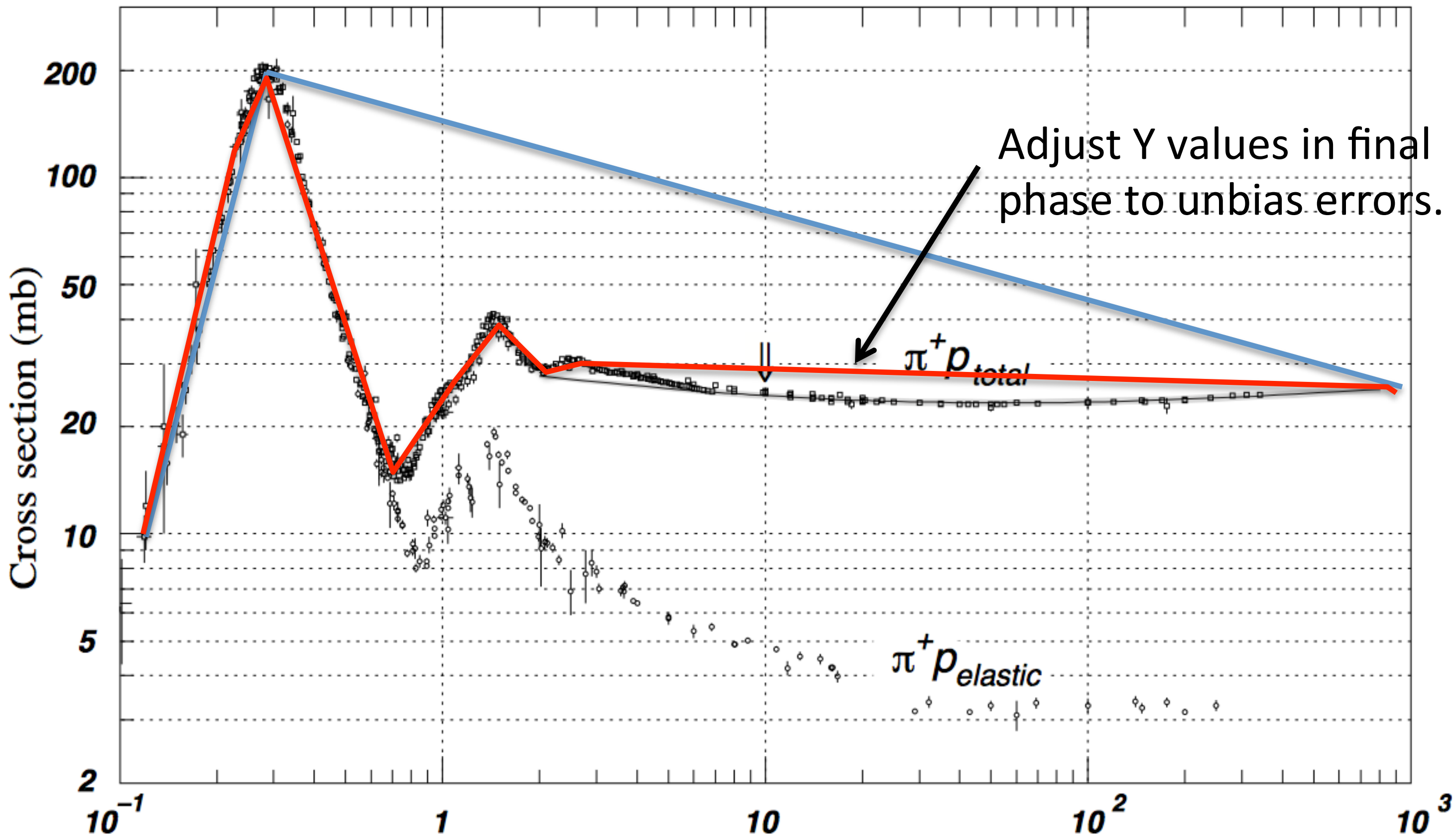
Douglas-Peucker Approximation.

Start with a piecewise-linear approximation.
In each segment, Add the point at the largest error.
Until total error is within the required bound.



Douglas-Peucker Approximation.

Start with a piecewise-linear approximation.
In each segment, Add the point at the largest error.
Until total error is within the required bound.



renci

USC Viterbi School of Engineering Information Sciences Institute

Building the Surrogate Model

- Create fast path for CrossSection calculations (offline or in initialization).
  - Identify common (particle, material, process) triples.
    - The number chosen depends on how much extra storage can be used.
  - Create Surrogate Model
    - Over sample using existing physics model.
    - Down sample to a simpler model with bounded error.
      - Evenly spaced sample points in E, ln(E), or log(E)
      - Or piecewise linear with adaptively placed nodes (Douglas-Peucker method).
    - Solve a linear system to adjust Y values to remove bias from the sign of the errors in each interval.
    - Represent the reduced model using existing G4PhysicsVector sub-classes.
- For triples where the fast path exists, use it for the mean free path (interaction length) calculations.
  - Single query of a G4PhysicsVector
- When an interaction occurs (rare)
  - Perform original physics calculations

renci

USC Viterbi School of Engineering Information Sciences Institute

Existing CrossSection Calculation

```
G4double G4CrossSectionDataStore::GetCrossSection(part,mat){
  ...
    if(mat == currentMaterial && part->GetDefinition() == matParticle
     && part->GetKineticEnergy() == matKinEnergy)
    { return matCrossSection; }

   //Calculate CrossSection the regular way (including xsecelm)
  ...
}

G4double G4CrossSectionDataStore::SampleZandA(part,mat){
  ...
    G4double cross = GetCrossSection(part, mat);
  ...
}
```

Modified CrossSection Calculation

```
G4double G4CrossSectionDataStore::GetCrossSection(part,mat,requireSlowPath){
  ...
    if(!requireSlowPath){
      fast_entry = (*fastPathMap)[searchkey];
    }
    if (!requireSlowPath && fast_entry != NULL){
      matCrossSection=GetCrossSectonFastPath(fast_entry,part);
    } else {
         //Calculate CrossSection the regular way (including xsecelm)
    }
  ...
}

G4double G4CrossSectionDataStore::SampleZandA(part,mat){
  ...
    G4double cross = GetCrossSection(part, mat, true);
  ...
}
```

Fast Path Usage: Runtime

Triples	Samples	Tolerance	Time	Percent Diff
90%	250K	1E-05	82:36	8.0%
90%	10K	1E-06	85:26	6.1%
90%	500K	1E-05	87:07	5.1%
90%	200K	1E-06	89:03	3.8%
Slow Path Only	N/A	N/A	94:57	0.0%
90%	2M	1E-06	99:18	-2.8%

Fast Path Usage: Cycles

Samples	Tol	Cyc/Op (fast)	Cyc/Op (slow)	Cyc/Op (avg)	Init Cycles	Cyc/Op (avg) w/ init	Total Calls
2M	1E-6	274	3982	839	2.77E+12	2731	1,468,837,903
1M	1E-6	252	3936	811	1.24E+12	1651	1,475,681,237
500K	1E-5	240	3873	786	2.58E+11	962	1,467,516,422
250K	1E-5	230	3981	801	1.23E+11	884	1,486,080,112
10K	1E-6	242	3882	820	6.17E+9	824	1,520,218,543

Questions?