Improvements to Bertini-style Intranuclear Cascade

Michael H. Kelsey
SLAC

7th GEANT4 Space Users Workshop, Seattle, WA

19 August 2010

Outline : Since 9.3(.p01)

- Cascade Model in GEANT4
- Physics-related Improvements
- Physics Validations
- Software Improvements *(backup slides)*
- Performance Improvements (**memory, CPU**)
- Known Issues (9.4-beta)
- Future Directions

Note: Page numbers reflect backup slides integrated into flow of presentation

Bertini-esque Cascade in GEANT4

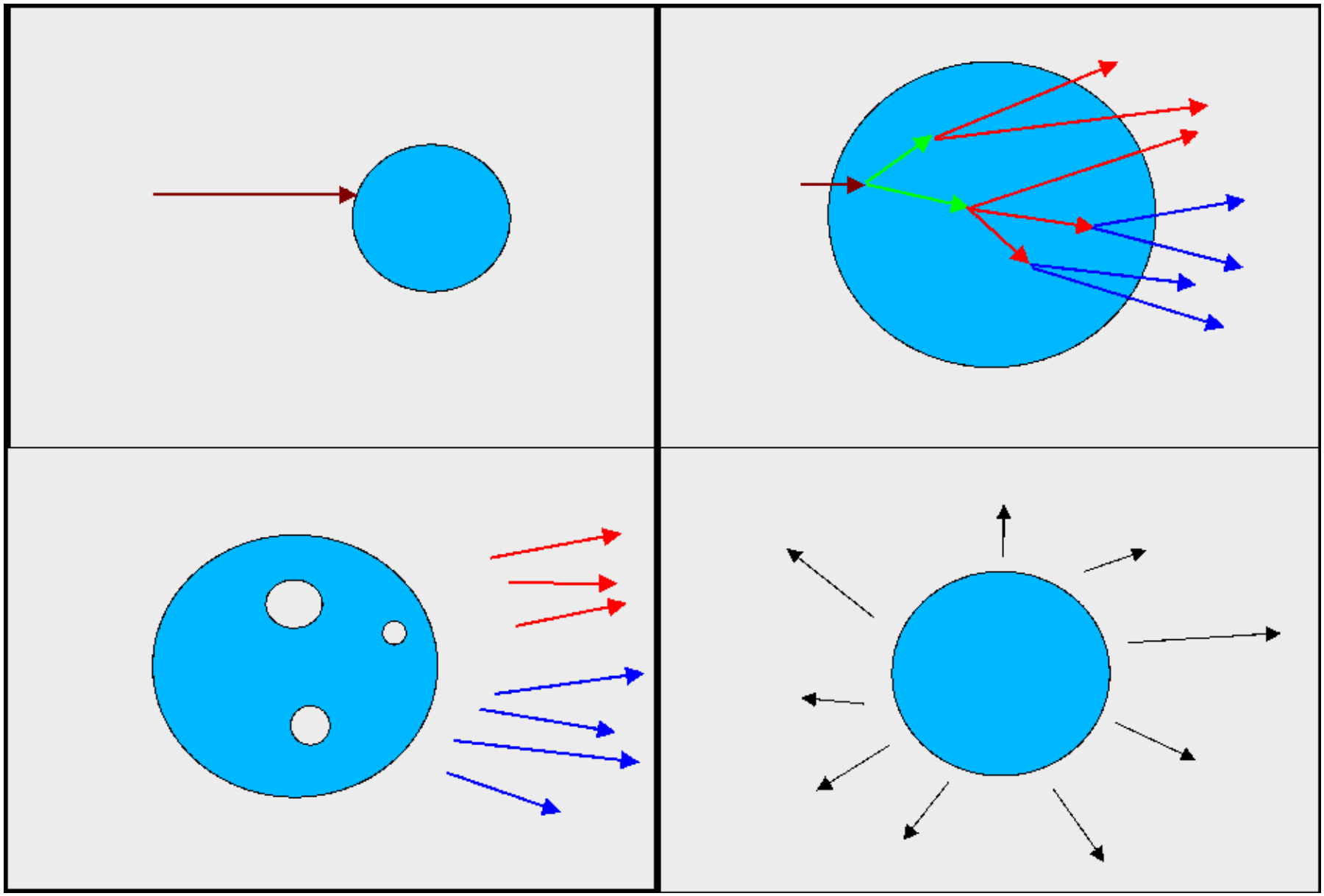
Classical particle cascade

- Solution to the Boltzmann equation on average
- No scattering matrix
- "Frozen nucleus" – no nucleon motion
- Can be traced back to early codes (1960s)

Components of Cascade Model

- Detailed 3-D model of nucleus (density and potential)
- Nucleons don't move, but momenta drawn from Fermi dist.
- Elementary particle collider: uses free-space cross sections to generate secondaries from collisions with each nucleon
- Secondaries propagate in nuclear medium
- Pre-equilibrium and equilibrium decay of residual nucleus

Cascade Sequence



Cascade Sequence

1. Incident particle penetrates nucleus, propagates in a density-dependent nuclear potential
2. Hadron-nucleon interactions based on free-space cross sections, angular distributions; no interaction if Pauli exclusion
3. Each secondary from interaction propagates in nuclear potential until interaction or leaves nucleus
4. During cascade, particle-hole exciton states are collected
5. Each nucleon interacts only once; cascade terminates when all nucleons have been touched
6. Pre-equilibrium decay occurs using exciton states
7. Residual nucleus may break up, evaporate, or fission

Performance Envelope

Used for nucleon, hyperon, and light meson projectiles

$$p,\ n,\ \pi^{\pm},\ \pi^{0},\ K^{\pm},\ K^{0}_{L},\ K^{0}_{S},\ \Lambda,\ \Sigma^{\pm},\ \Xi^{-},\ \Xi^{0}$$

$\Longrightarrow$ Extending support to include Ω^{-} and antinucleon projectiles

Kinetic energies 0–10 GeV, extending to 12∼15 GeV

$\Longrightarrow$ Not recommended for primaries below $\sim$ 800 MeV

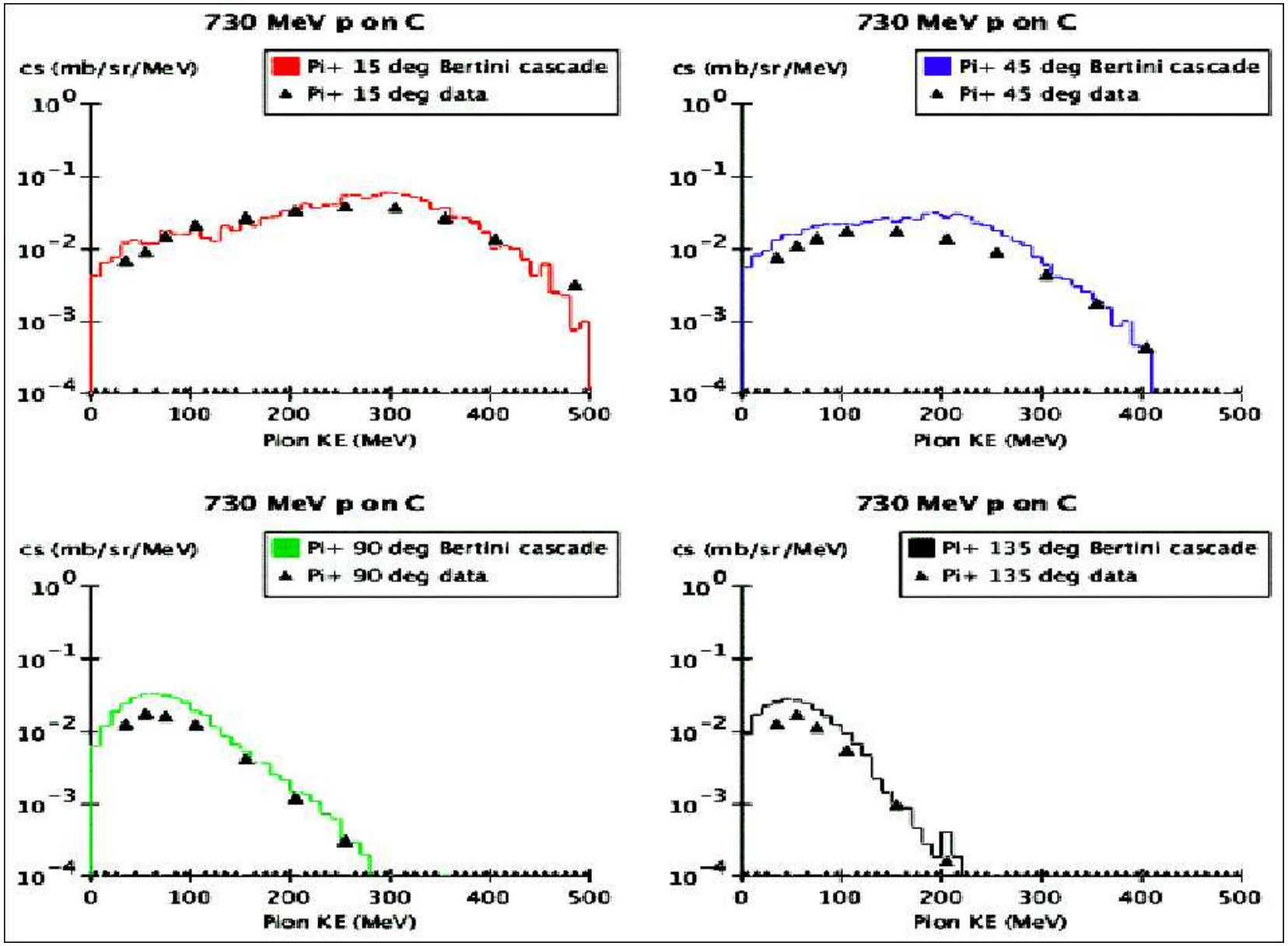
Hadron-nucleon interactions up to 8-body final states

Best for "non-light" targets (A $\gtrsim$ 15), where potential and nuclear density are good assumptions

Model is simpler (faster) than binary cascade, results not really appropriate for sub-GeV projectiles

$\Longrightarrow$ *See R. Reed's talk [8], Wed 9:30*

Performance Validation



Physics-related Improvements

- π-N, K-N, N-N angular distributions

  Use exponential for quasielastic scattering

- Nuclear binding energy

  Use standard GEANT4 function, rather than custom

- Evaporation parametrization

  Fix bug which set parameters to zero on bin edge

- Baryon and charge non-conservation

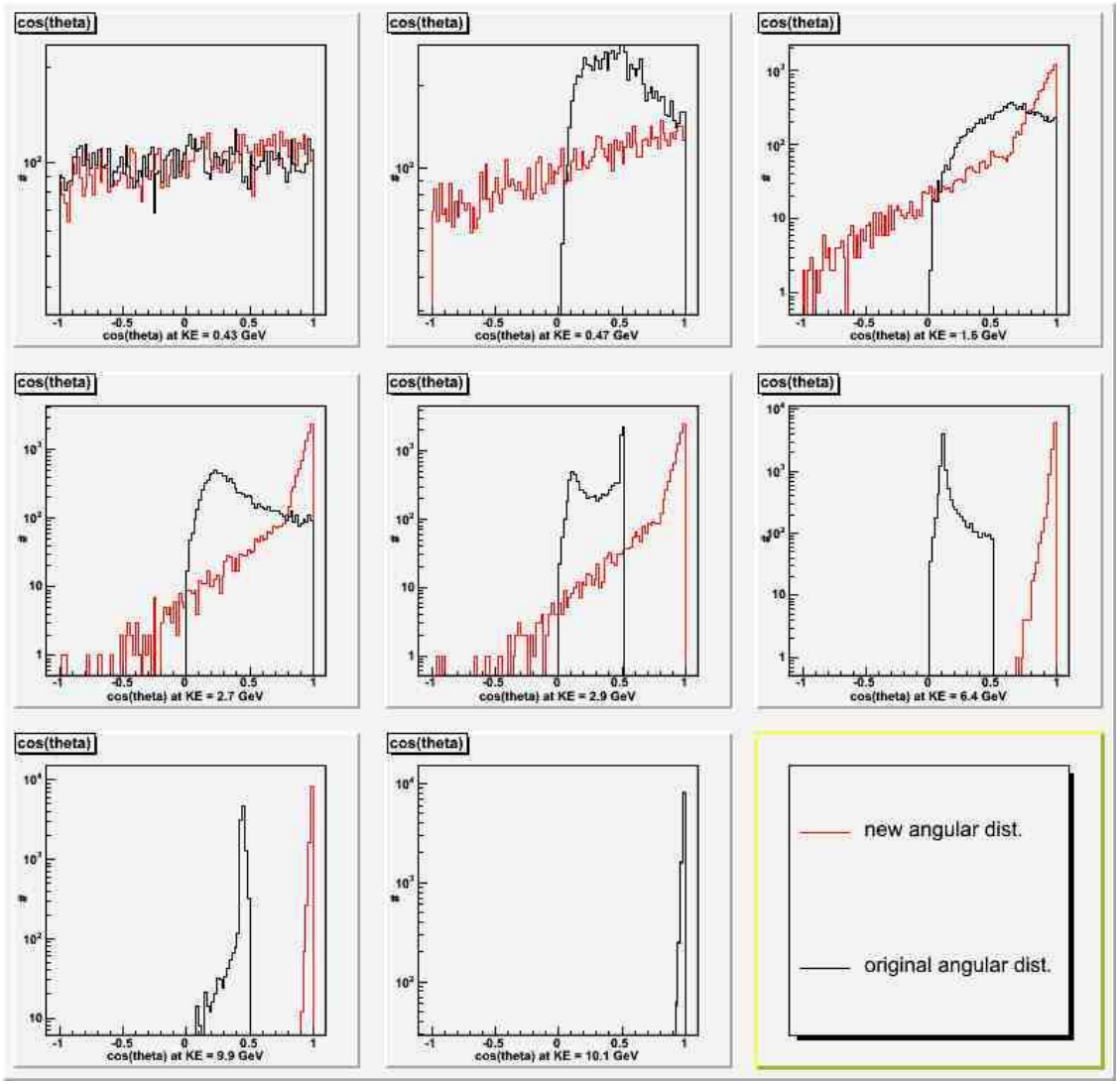
  Pending cascade secondaries included in final state

- Energy/momentum conservation

  Use proper four-vector operations everywhere, nuclear excitation is mass, discard non-conserving cascades

N-N Quasielastic Scattering

RED: New **BLACK: Old**



Energy and Momentum Conservation

Most code performed three-vector manipulations, applied masses "after the fact" to get energies

Now: Proper covariant calculations used throughout, with mass assignments "up front"

Kinetic energy used inappropriately in several places (nuclear excitation, inelastic collisions)

Now: Nuclear excitation treated as mass contribution

Energy/momentum imbalance "fixed" ad-hoc after cascade

$\Longrightarrow$ Not stable, can get non-physical results (50 TeV deposited from 10 GeV $\pi+$)

Now: Cascades which don't balance are discarded and regenerated

Remaining Non-conservation

Secondaries change direction and momentum travelling through potential (at zone boundaries)

Nuclear potential is treated as fixed, infinite mass

Without proper nuclear recoil, final state particles can't balance initial four-momentum

Workaround: If cascade leaves a nuclear remnant, it is assigned recoil $p_{\mathrm{initial}} - p_{\mathrm{final}}$, then de-excited

Light nuclei can be completely disrupted, leaving no remnant to balance four-momentum

Performance Improvements

Main performance issue is "**memory churn**"

Many (millions) cycles creating/deleting small objects

Dominated by passing and copying `std::vector` containers

- Replace by-value pass/return with references
- Use data members for within-class passing
- Pre-allocate fixed-length vectors where possible
- Minimal use of `static` variables (not thread-safe)

Test job (1k p–Pb collisions) allocated 210 MB in 9.3 (∼2,000 page faults), reduced to **8 MB** (no PFs) in latest tag

CPU usage now 99% efficient (was 65–70%)

CMS and ATLAS reported 5–8% reduction in total "churn" in 9.3-ref-05

Known Issues (9.4-beta)

If you try the 9.4-beta release, Cascade code has some issues

- Neutral kaons have odd kinetic energy spectra (GeV vs. MeV bug)
- Final-state lookup tables swapped for $\pi^+ p$ and $\pi^- n$ 8-body channels
- Unphysical energy deposition (e.g., 50 TeV from 10 GeV π^+)
- Occasional charge or baryon non-conservation
- Infrequent FPEs/errors for negative-energy secondaries
- Semi-inclusive cross sections reduced at mid/large angles

All known bugs fixed in `hadr-casc-V09-03-67`

Cross-sections under study to verify consequences of physics improvements

Future Directions

Top priority: Validate changes in cross-sections compared to 9.3

Modify cascade propagation code to treat finite-mass (recoiling) nucleus

Collect and deploy cross-section tables for additional projectile hadrons (Ω^-, $\bar{p}$, $\bar{n}$)

Eliminate "internal" de-excitation models; defer to existing GEANT4 processes

Continue consolidation and rationalization of code base

Replace internal particle classes with standard GEANT4 tools

Backup Slides

Physics Lists

The Cascade is part of the reference physics lists with `BERT` in the name (e.g., `QGSP_BERT`)

Building a custom physics list:

```
G4CascadeInterface* bert = new G4CascadeInterface();
G4ProtonInelasticProcess* pproc = new G4ProtonInelasticProcess();
pproc->RegisterMe(bert);
proton_manager->AddDiscreteProcess(pproc);
```

In upcoming releases, diagnostic reporting:

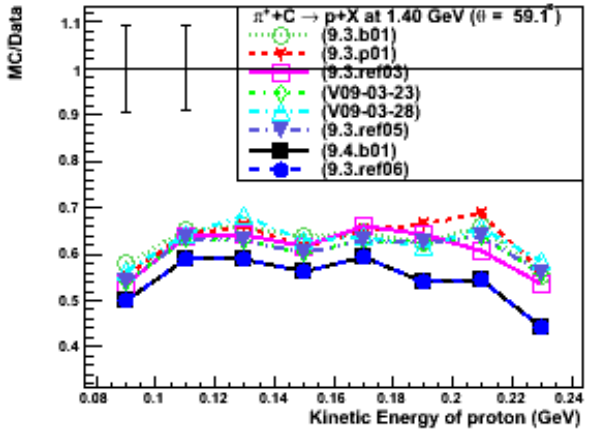
```
bert->setVerboseLevel(N);
```

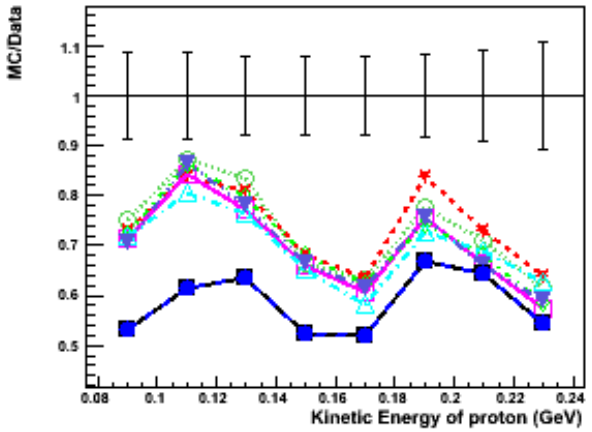
As N increases from 1 to 4, more detailed information

Performance Validation

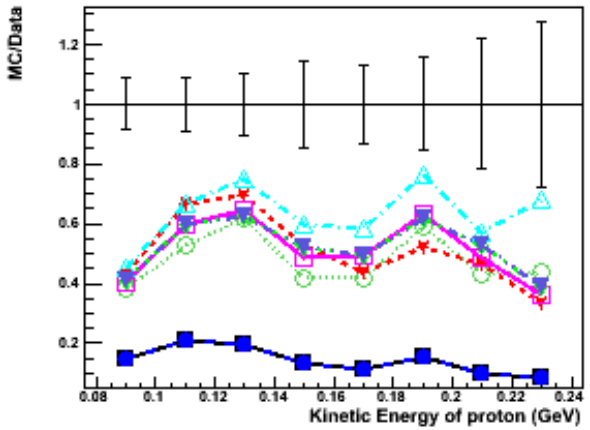
$$\pi^+ + C \Rightarrow p + X$$

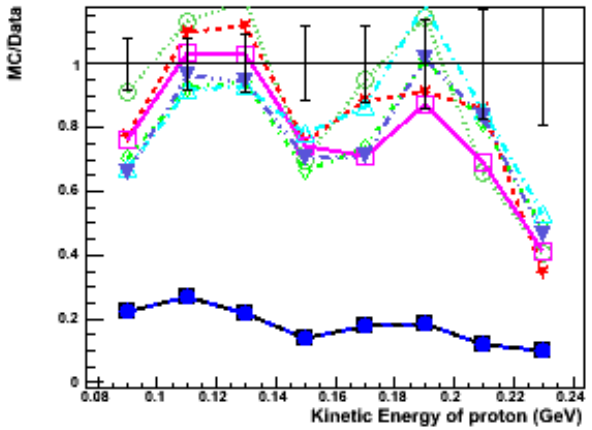
1.4 GeV/c 5.0 GeV/c





59.1°





119.0°

Results courtesy S. Banerjee, FNAL

Two-body Quasielastic Scattering

Four channels: N-N; h-N I=$\frac{3}{2}$, I=$\frac{1}{2}$, charge exchange

Two regions, chosen randomly with fraction F

$0 < \theta < \theta_0$ (small angle): $t_1 = 2\,A\,|\vec{p}_{\rm cm}|^2$

$\theta_0 < \theta < \pi$ (large angle): $t_1 = 2\,C\,|\vec{p}_{\rm cm}|^2$

$$t_2 = e^{-2t_1}, \qquad s = \frac{e^{-t_1(1-T)} - t_2}{1 - t_2}$$

Small angle: $r = (1-s) \times \mathbf{random} + s$

Large angle: $r = s \times \mathbf{random}$

Scattering angle: $\cos\theta = 1 + \dfrac{\log[r(1-t_2)+t_2]}{t_1}$

Parameters A, C, F, T all binned functions of $E_{\rm kin}$

Internal Software Improvements

- Particle properties, kinematics

  Use existing GEANT4 classes, CLHEP `LorentzVector`

- Function encapsulation (ongoing)

  Throwing angular dists, interpolator, drop `std::pow`

- Inelastic scattering final states

  Common base class for lookups, interpolation

- Model process classes

  Common base class with shared utility functions

- Diagnostic messages

  Uniform use of settable `verboseLevel` values, scripts for counting occurrences, `diff`-ing results

Internal Improvements (I)

Particle properties, kinematics

Internal classes use index code to identify particle

Masses typed in by hand, may not match GEANT4 definitions

Momenta stored in simple C arrays, by-hand kinematics, index manipulation

Now: Use standard classes `G4ParticleDefinition`, `G4DynamicParticle`, `G4LorentzVector`

Function encapsulation

Lots of identical code blocks duplicated throughout package

Use of `std::pow` with integer arguments, or with 0.333, 0.667 for cube roots

Now: Encapsulated utilities, `G4cbrt` using log(), exp()

Internal Improvements (II)

Inelastic scattering tables

Three separate interfaces, π–N, N–N, and strange–N

Interpolations and lookups done by hand

Now: Templated base class for lookup tables, interpolator class

Model process classes

All classes use same interface `collide(bullet, target)`, returning final-state list by value

Interface enforced administratively (just happen to be the same), copying and pasting with duplications

Now: Abstract base class with required interface, shared utilities provided via member functions

Internal Improvements (III)

Diagnostic messages

Many classes have local `verboseLevel` data member, set only through constructor initialization

Use, and meaning of values, vary widely from class to class

Now: Public `setVerboseLevel` function in base classes

Value passed to utility classes, so that user can set level and get diagnostics from entire cascade procedure

Meanings of levels have been (mostly) rationalized

1. major function calls
2. blocks within functions and output particle lists
3. intermediate state particles, calculations
4. detailed calculations and terms

Performance Improvements

Evaluate performance using **IgProf** tool from CMS experiment

Output like `gprof`, can monitor memory usage as well as calls

Sqlite output format with Web browsing interface

%	Memory	Calls	Function/ctor
100.00	220,714,885	3,178,293	Cumulative
40.62	89,661,000	582,350	`G4InuclElementaryParticle`
23.30	51,429,080	1,666,392	`double` (vector)
13.31	29,372,396	264,785	`std::pair<G4InuclEP, double>`
11.46	25,293,424	192,515	`G4CascadParticle`
3.74	8,249,344	1,007	`iostream`

9.3-ref-03 baseline, 1,000 events 4 GeV p on ^{207}Pb

Performance: 9.3-ref-04

%	Memory	Calls	Function/ctor
100.00	92,727,821	547,728	Cumulative
64.55	59,859,520	114,769	`G4InuclElementaryParticle`
8.90	8,249,344	1,007	`iostream`
5.87	5,439,588	9,860	`G4CascadParticle`
4.13	3,832,400	5,277	`G4InuclNuclei`
4.02	3,729,408	151,282	`double` (vector)

First round of buffer improvements

Performance: 9.4-beta-01

%	Memory	Calls	Function/ctor
100.00	165,218,402	764,666	Cumulative
77.48	128,013,480	192,338	`G4InuclElementaryParticle`
6.10	10,081,696	296,655	`double` (vector)
4.99	8,249,344	1,007	`iostream`
2.69	4,447,240	9,136	`G4CascadParticle`
1.69	2,784,550	43,872	`string`

Substantial reorganization of code, iterations, physics

Performance: hadr-casc-V09-03-55

%	Memory	Calls	Function/ctor
100.00	681,160,264	2,354,839	Cumulative
90.73	618,005,360	1,052,674	`G4InuclElementaryParticle`
5.44	37,032,696	1,036,292	`double` (vector)
1.21	8,249,344	1,007	`iostream`
0.87	5,899,796	10,757	`G4CascadParticle`
0.56	3,799,060	56,957	`string`

Conservation law enforcement, maximum use of retries

Performance: hadr-casc-V09-03-67

%	Memory	Calls	Function/ctor
100.00	16,508,789	218,210	Cumulative
49.97	8,249,344	1,007	`iostream`
12.41	2,048,088	9,018	`long` (vector)
9.02	1,488,497	25,274	`string`
4.85	800,800	100,100	`CLHEP::DoubConv::dto2longs`
4.60	759,220	37,961	`G4HadFinalState::AddSecondary`

All buffers fixed. No Cascade code in top five allocators!