LA-UR-16-20300

Approved for public release; distribution is unlimited.

Title:	Collision of Physics and Software in the Monte Carlo Application Toolkit (MCATK)
Author(s):	Sweezy, Jeremy Ed
Intended for:	Presentation to West Point Physics Dept. Staff
Issued:	2016-01-21

Collision of Physics and Software in the Monte Carlo Application Toolkit (MCATK)

Jeremy Sweezy

January 22, 2016

UNCLASSIFIED

Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- Recent Areas of Research
- Summary and Future Work

Outline

- **MCATK Overview**
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- Summary and Future Work

What is MCATK?

- A C++ component-based Monte Carlo neutron-gamma transport software library
  - Development began in 2008
- Continuous energy neutron and photon transport
- Reads ACE formatted nuclear data generated by NJOY
- Designed to build specialized applications
- Designed to provide new functionality in existing general purpose Monte Carlo codes like MCNP
- Developed with Agile software engineering methodologies

MCATK Motivation: Reduce Costs

- Reduce code duplication
- Reduce code complexity
- Reduce time to deliver custom solutions
- Leverage external packages
- Provide re-usable components
- Find defects earlier in the development cycle

MCATK: Reusable Components

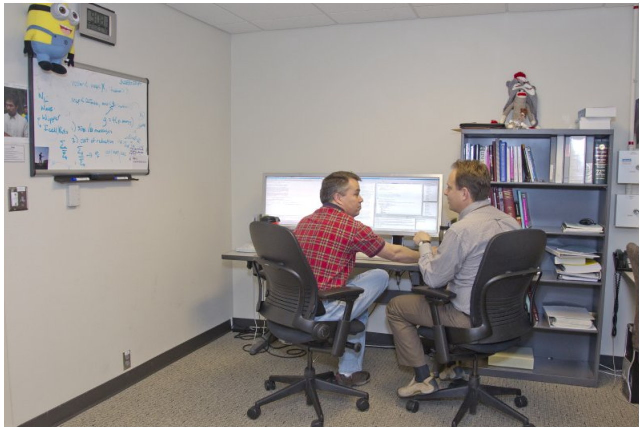
- Provide components for:
  - Existing codes (example: MCNP)
  - Stand alone applications (example: MCATK K_{eff} solver)
- Components are well tested
  - Unit testing – verification
  - Integrated physics tests – validation
- Flat API available for C and Fortran codes
- Object Oriented API for C++ codes

Outline

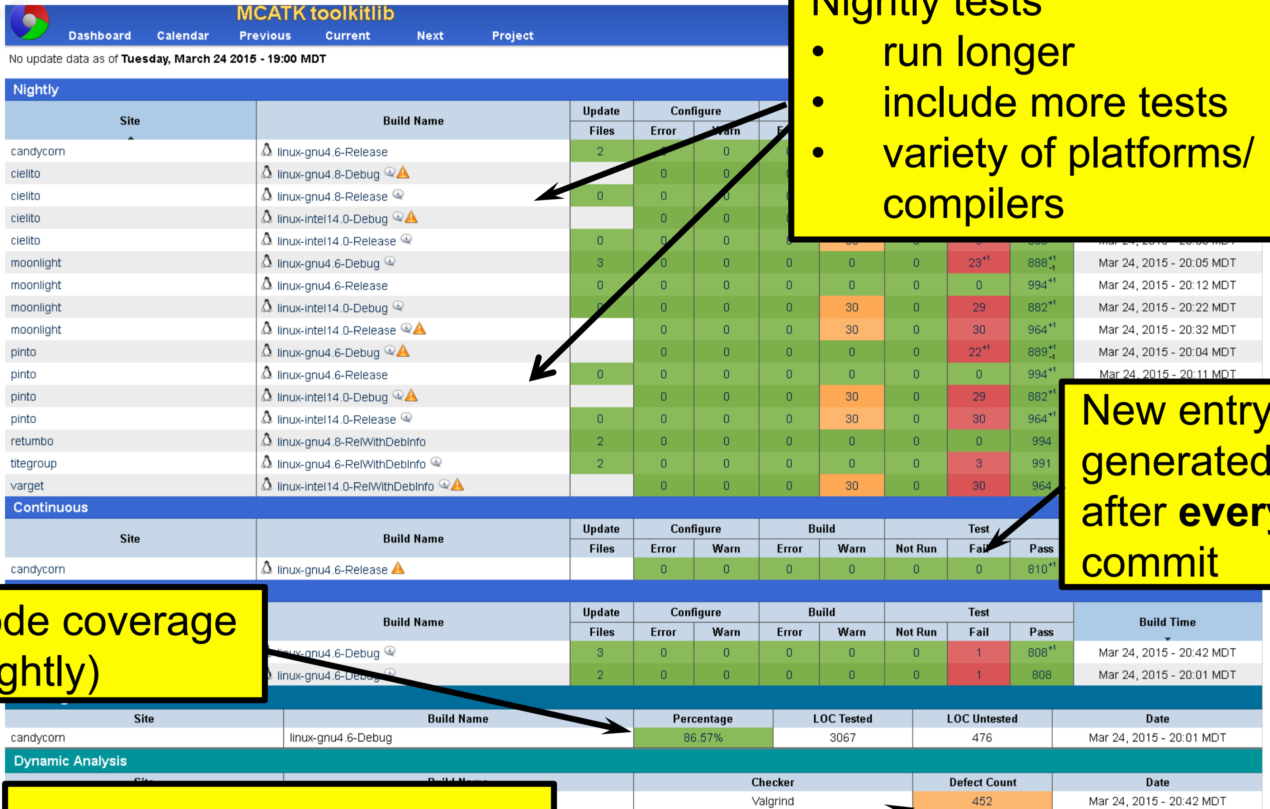
- MCATK Overview
- **Development Strategy**
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- Summary and Future Work

MCATK: Development

- Agile software development methodology
  - Incremental development
  - Short iteration cycles (2 weeks)
- Test driven development (TDD)
  - Test first philosophy
  - Unit testing with UnitTest++
  - Unit tests built and executed continuously (with each commit)
  - Goal: ~10 minute build
- Pair-programming, Colocation
  - Improves design and testing
  - Promotes knowledge sharing and collective ownership of code



MCATK: Continuous Testing and Reporting



Nightly tests
- run longer
- include more tests
- variety of platforms/ compilers

New entry generated after **every** commit

Code coverage (nightly)

Memory leaks etc. (even 3rd party libraries!) (nightly)

Outline

- MCATK Overview
- Development Strategy
- **Available Algorithms**
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- Summary and Future Work

What does MCATK solve?

$$\left[\frac{1}{v}\frac{\partial}{\partial t} + \Omega\cdot\Delta + \Sigma(r,E,t)\right]\phi(r,\Omega,E,t)$$

$$= \int_0^\infty dE' \int_{4\pi} d\Omega' \Sigma_s(r,\Omega'\cdot\Omega,E' \rightarrow E,t)\phi(r,\Omega',E',t)$$

$$+ \frac{\chi(E)}{4\pi}\int_0^\infty dE' \int_{4\pi} d\Omega' \nu(E')(\Sigma_f(r,E',t)\phi(r,\Omega',E',t)$$

$$+ Q(r,\Omega,E,t)$$

NOT REALLY!

Power Iteration

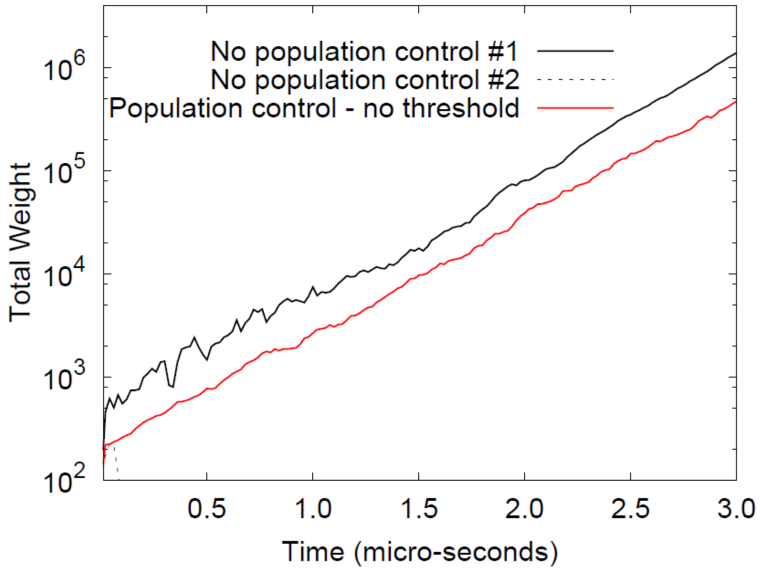
- k_{eff} Eigenvalue solver
- α-Eigenvalue solver
  - α is the inverse of reactor period
  - Uses "time absorption" i.e. α/velocity
- Validation suite covers all of the criticality test problems from the MCNP validation suite
- Test the MCNP & MCATK differences with Shapiro-Wilk test for normality. At the 5% significance level, the difference between MCNP & MCATK cannot be shown to be be non-normally distributed.

	MCNP		MCATK		
Case	k_{eff}	Std. Dev.		Std. Dev.	
BAWXI2	1.0092	0.0013	1.0064	0.0013	-1.52
BIGTEN	0.9919	0.0009	0.9944	0.0010	1.86
FLAT23	0.9992	0.0010	0.9982	0.0011	-0.67
FLAT25	1.0014	0.0010	1.0030	0.0010	1.13
FLATPU	1.0003	0.0010	0.9998	0.0010	-0.35
FLSTF1	0.9838	0.0017	0.9868	0.0018	1.21
Godiva	1.0000	0.0008	0.9998	0.0009	-0.17
GodivaR	1.0197	0.0011	1.0170	0.0012	-1.66
HISHPG	1.0115	0.0010	1.0113	0.0009	-0.15

Work performed by Jesse Giron, ASU

Time-Dependent Solver

- Like MCNP fixed source mode with discrete time steps
- Particle are added to “census” if they survive to the end of the time step
- Tallies for time-dependent α
- Population control
  - Applied to the census between time steps
  - Prevent exceeding memory limits for supercritical systems
  - Prevents the population from dying off in subcritical or stochastic systems (i.e. systems with small neutron populations)



Total weight vs. time for a super-critical analytic problem using the MCATK time-dependent algorithm

Time-Dependent Solver

- We can calculate alpha the old fashion way, just like in a physical experiment:

$$N(t_1) = N(t_0)e^{\alpha(t_1 - t_0)}$$

or:

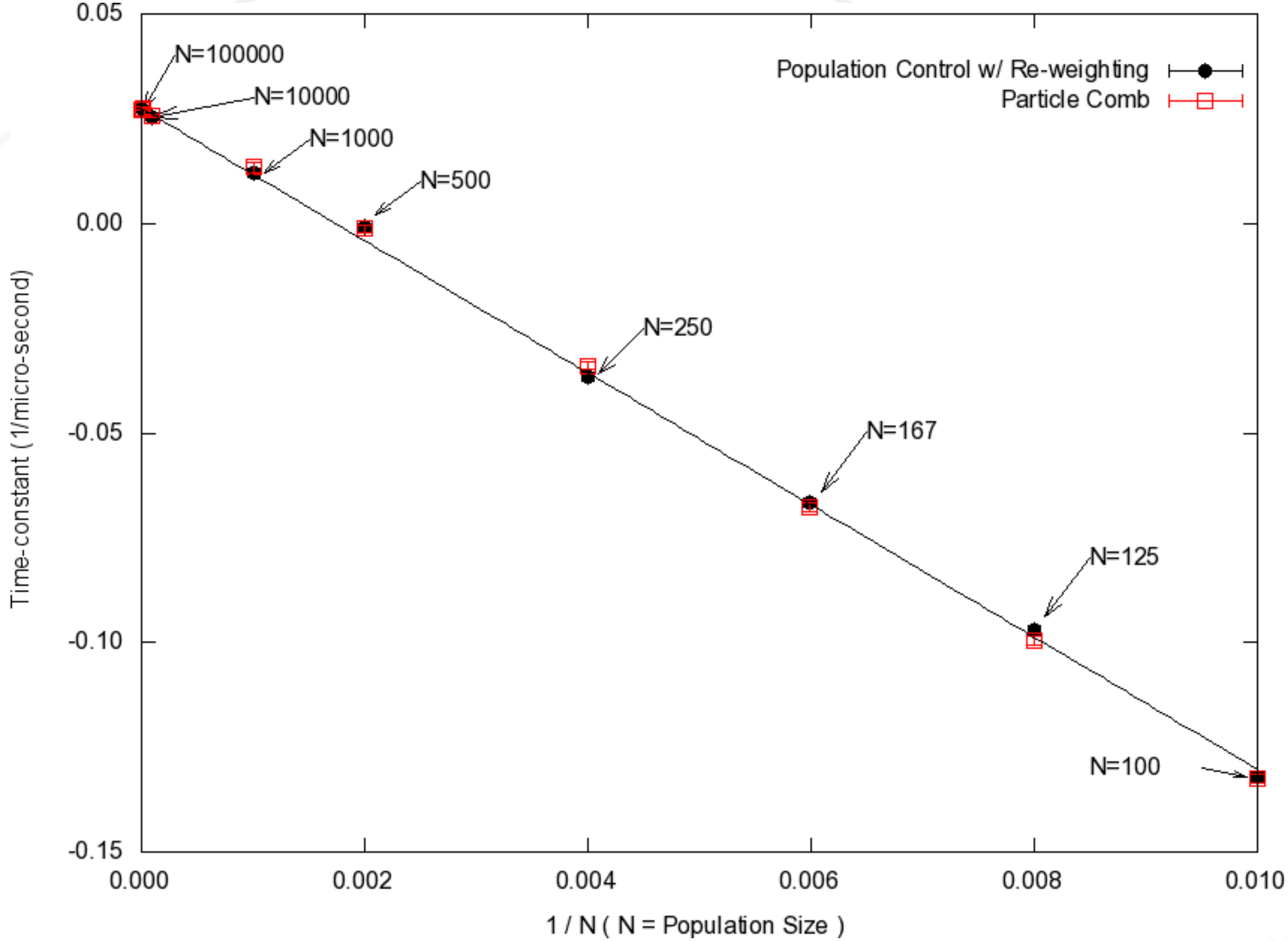
$$E[\alpha] = \frac{1}{t_1 - t_0} E\left[\ln \frac{N(t_1)}{N(t_0)}\right]$$

- Since the logarithm function is concave, Jensen's inequality shows that E[ln(X)] ≤ ln(E[X]), therefore:

$$E[\alpha] \leq \alpha_{actual}$$

- Bias is due to taking the logarithm of a random value.

Time-Dependent Solver: Bias in α estimation



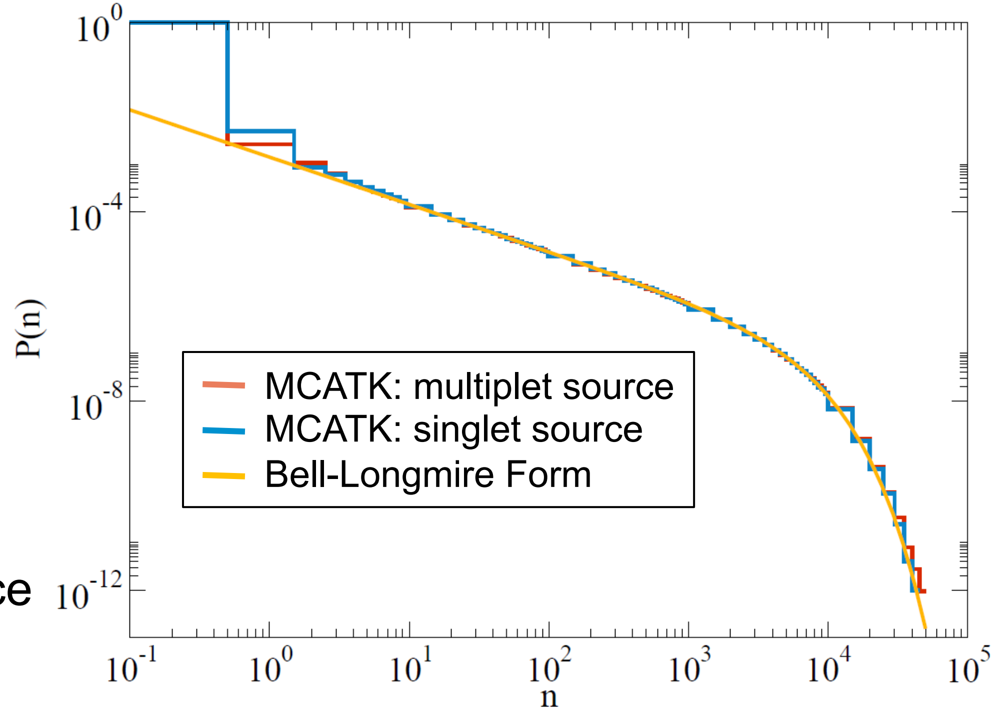
UNCLASSIFIED

Fission Chain Analysis

- Simulate the random histories of complete fission chains and observe their stochastic behavior
- For simulation of:
  - Pulsed nuclear reactors (e.g., Godiva, Caliban)
  - Criticality accident scenarios
- Tally:
  - Probability of initiation (POI)
  - Probability of extinction (POE)
  - Probability of survival (POS)
  - Total / net / leakage multiplication
  - Moments of the neutron population distribution resulting from single neutrons or neutron sources

Fission Chain Analysis

- POI: Probability that a fission chain persists for an infinite time
- Cannot track an infinitely long chain
- Instead calculate POE (POI=1-POE)
- Importance method for POE:
  - Give each chain an importance and a weight
  - Larger chains are less important to POE
  - Play Russian Roulette on entire fission chains



PDF of neutron number, ***n***, for a Pu sphere

Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- **Problem Modeling (Sources, Geometry, Data, Tallies)**
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- Summary and Future Work

Sources

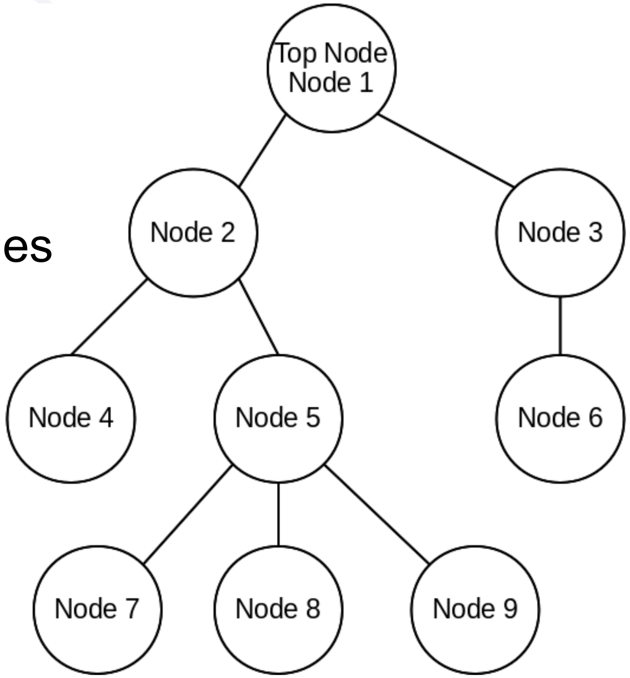
- Point sources
- Mesh sources
- Time-varying distributions
- Spontaneous fission
  - Can use average multiplicity or sample from actual multiplicity data
- Surface source files
  - Link to MCNP

MCATK Geometry

- LNK3DNT mesh files
- Mesh specification through the API
- Mesh types supported:
  - Spherical: R
  - Cylindrical: R, RZ, RZθ
  - Cartesian: X, XY, XYZ
- 3-D Solid Body Geometry
  - Available geometric primitives:
    - Spheres
    - Cylinders (right finite cylinder)
    - Boxes(right parallel piped)
    - Cones (truncated circular cones)
  - Each primitive can have a rotation and translation

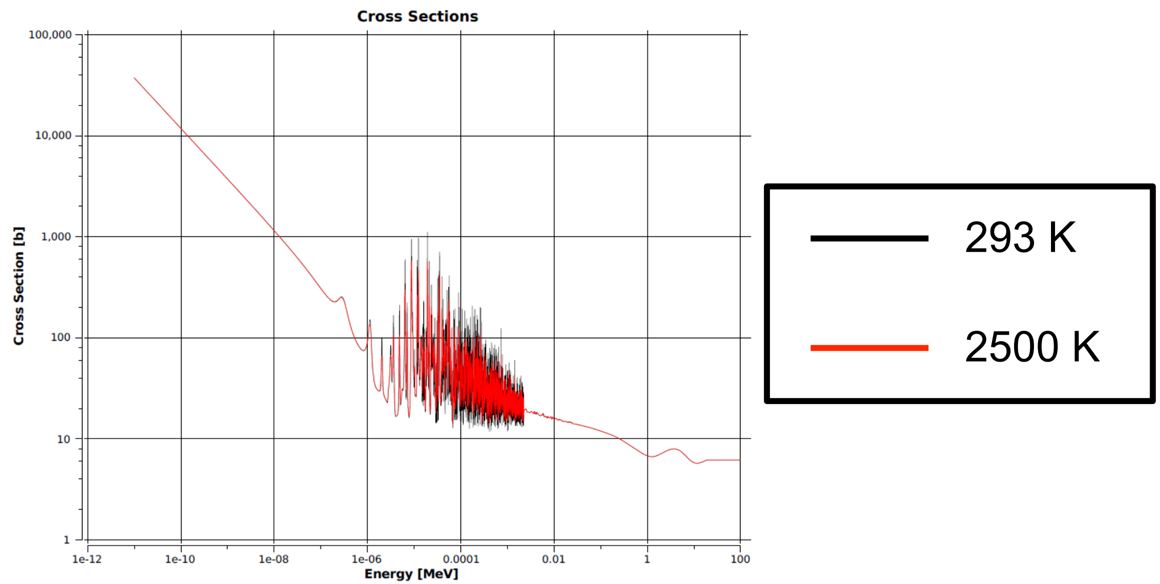
More on MCATK Geometry

- MCATK uses a scene graph hierarchy
  - Each solid is assigned to a node in a tree
  - Each node is a child of (contained by) the node above
  - Similar to CERN's ROOT and GEANT4 geometries
- Nodes have the following properties:
  - Assigned a geometric primitive or mesh
  - Material ID and density
  - Assigned children nodes
  - Children are allowed to overlap, but MCATK applies an order of precedence
  - An optional transformation inherited by its children
  - Parent containment and child precedence can be used to replicate combinatorial operators



Cross Section Data and Multi-temperature Treatment

- MCATK uses continuous energy data read from ACE data files
- Multi-temperature treatment:
  - MCATK is able to read in cross sections processed at more than one temperature
  - It picks the cross section table that was processed at the temperature that is closest to, but does not exceed the cell temperature
  - This capability is expected to be a step towards Doppler broadening on-the-fly and other multi-temperature treatments



Photon Physics

- MCATK uses the same continuous energy photon data as used in MCNP
- Photo-electric is treated as a terminating event
- Pair-production produces two 0.511 MeV photons at the collision site
- Incoherent (Compton Scattering):

$$p(\mu) = I(Z, \alpha, \mu) K(\alpha, \mu)$$

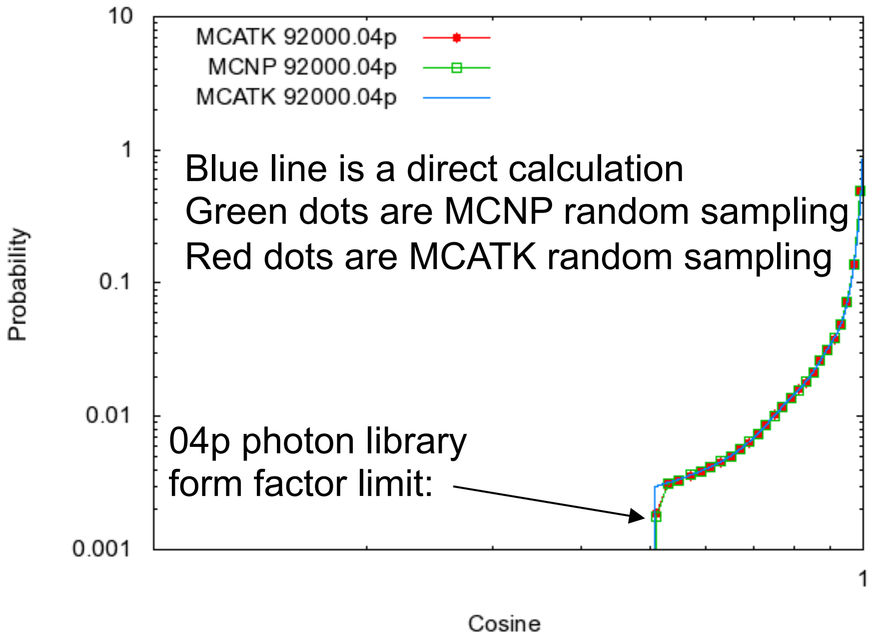
I - Incoherent form factor & K - differential Klein-Nishina cross-section

- Coherent (Thompson Scattering):

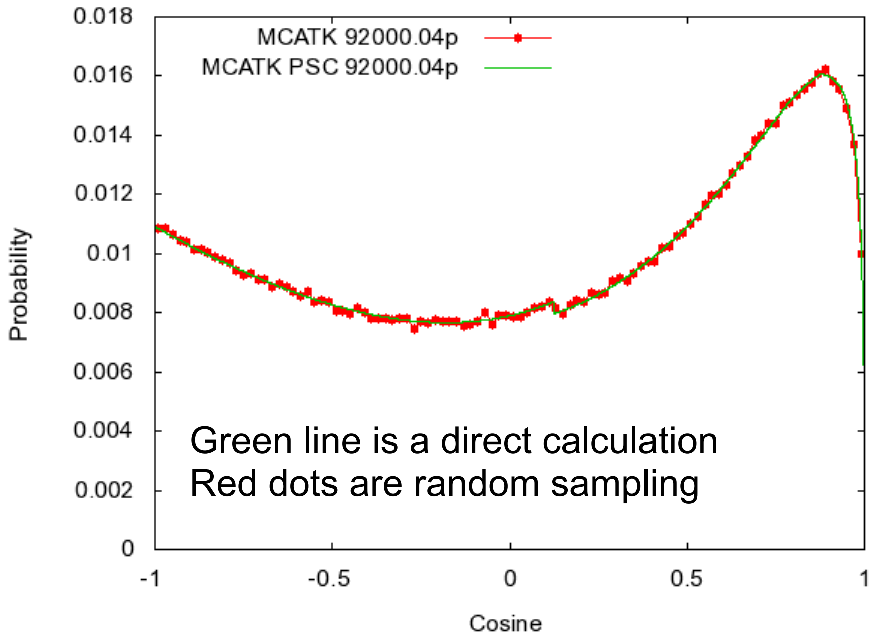
$$p(\mu) = C^2(Z, \alpha, \mu) T(\mu)$$

C - Coherent form factor & T - differential Thomson cross-section

Validation of Photon Scattering



Test random sampling of of coherent scattering angular distribution against a direct calculation. 0.15 MeV photon on Uranium.

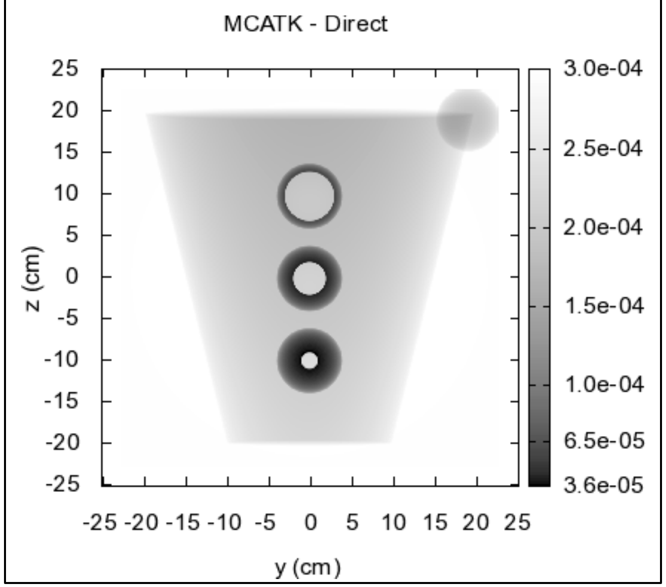


Test random sampling of of incoherent scattering angular distribution against a direct calculation. 0.15 MeV photon on Uranium.

Tallies

- Neutron and photon track length tallies
- k_{eff} and α track length tallies
- Leakage tallies
- Photon next-event estimators

Simulated radiograph of a test object using next event estimators



Photon Next-event estimators

Detector Response

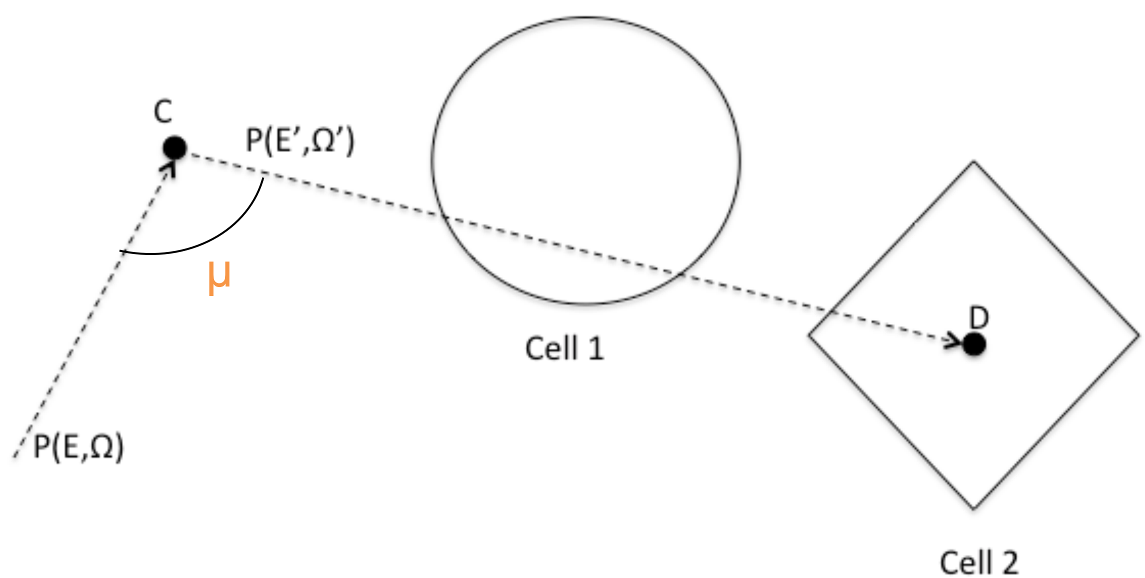
Reaction Cross-Section

$$S(R,E) = \frac{w}{2\pi R^2} \sum_{i=1}^{N} \frac{\sigma_i(R,E)}{\sigma_T} p_i(\mu, E \rightarrow E') e^{-\int_0^R \Sigma_T(s,E')ds}$$

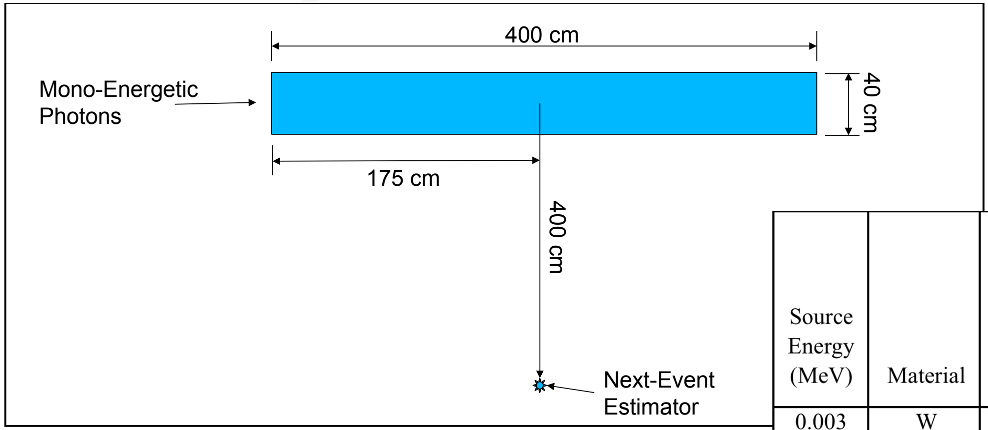
Reaction

Probability of Scatter into μ

Ray-Trace



Photon Next-event Estimators



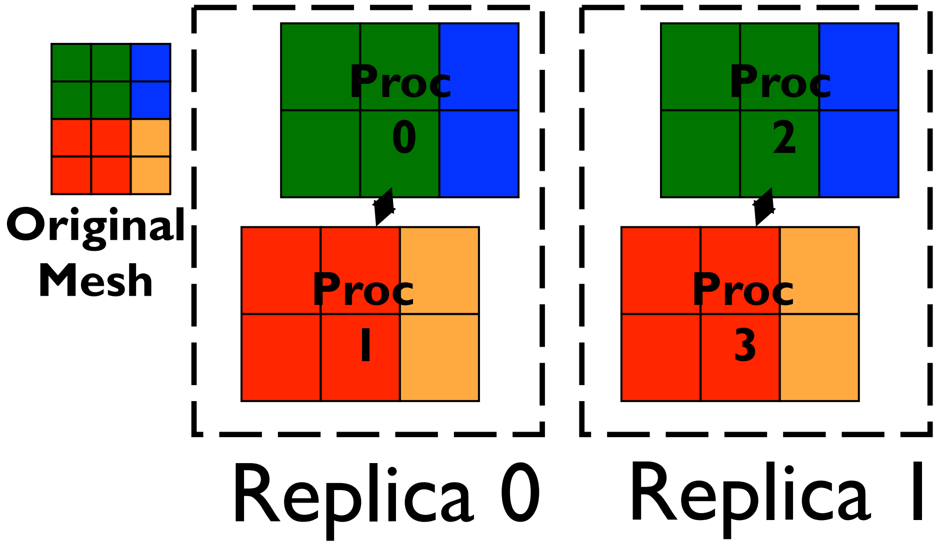
Source Energy (MeV)	Material	Density (g/cc)	MCNP Flux Estimate (particles/ cm^2)	MCNP Flux Relative Uncertainty	MCATK Flux Estimate (particles/ cm^2)	MCATK Flux Relative Uncertainty	Relative Difference (MCNP-MCATK) / MCNP
0.003	W	7.5E-5	8.60E-11	6.8E-5	8.60E-11	6.8E-5	-0.00019
0.015	W	1.0E-3	3.93E-10	0.00028	3.93E-10	0.00028	-0.00011
0.130	W	5.6E-2	7.74E-10	0.0018	7.76E-10	0.0018	-0.0026
2.0	W	2.6	8.12E-09	0.0031	8.16E-09	0.0032	-0.0060
60.0	W	2.0	8.16E-09	0.0029	8.09e-09	0.0029	0.0074
2.0	Concrete	1.9	1.53E-07	0.00069	1.53e-07	0.00069	0.00011
2.0	H_2O	1.5	2.52E-07	0.00056	2.52e-07	0.00056	0.00013
2.0	Fe	2.3	7.67E-08	0.00094	7.66e-08	0.00094	0.0016

Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- **Parallelism**
- Miscellaneous Tools/Features
- Example MCATK Application
- Summary and Future Work

Parallel Models

- MPI (distributed memory)
  - Domain replicated (particle decomposed)
  - Domain decomposed
  - Hybrid (decomposed and replicated)



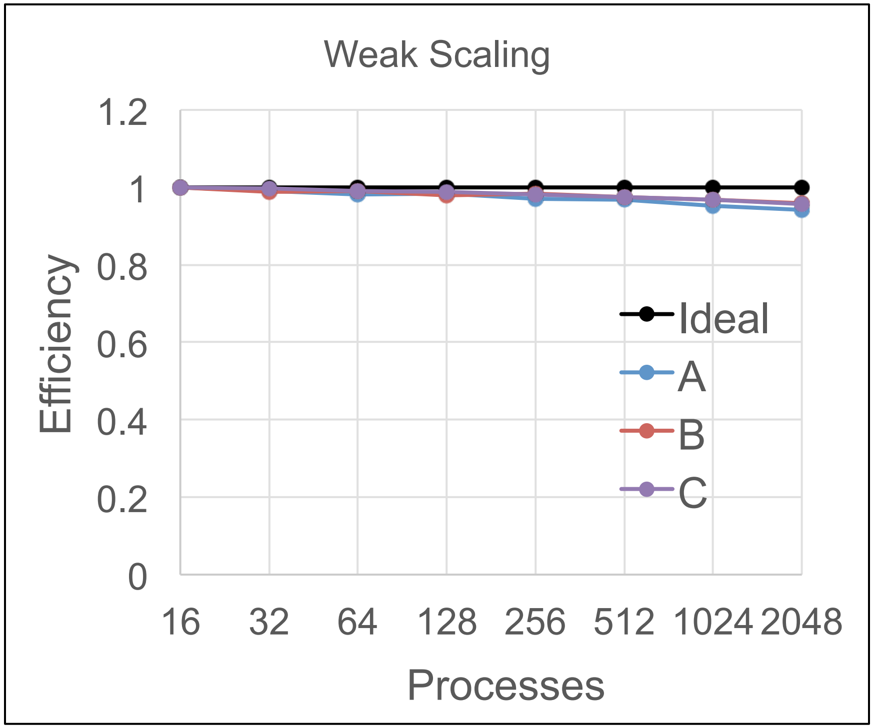
- Threading (shared memory) models currently being implemented

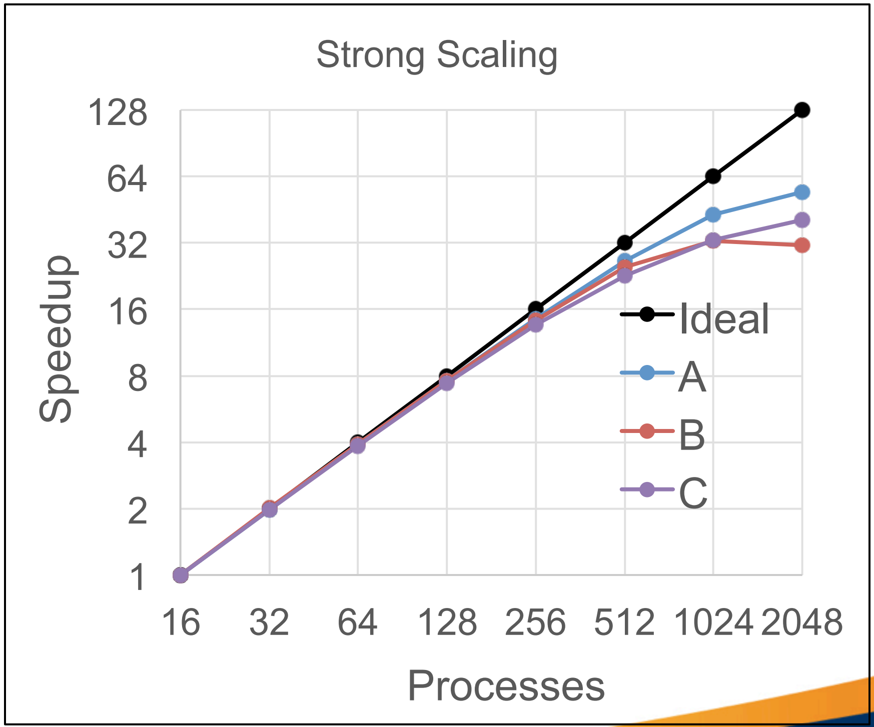
Parallel Performance

- A - Godiva - 1-D spherical calculating k_{eff} with the *power-iteration algorithm*
  - 1,000 Active/50 Settles
  - 10,000 particles/generation/process (weak)
  - 500,000 particles/generation (strong)
- B - Godiva - 1-D spherical calculating alpha time constant and neutron fluence rates with the *time-dependent algorithm*:
  - 250,000 particles/proc (weak) : 500 steps @ 0.1sh
  - 1.5 M particles (strong) : 2500 steps @ 0.1 sh
- C - ZEBR8H - 2-D cylindrical calculating k_{eff} with the *power-iteration algorithm*:
  - 1,000 Active Cycles/50 Settles
  - 5,000/generation/proc (weak)
  - 100,000 particles/generation (strong)
- Performance tests are run as part of the weekly test suite
  - Tests may not involve enough work for all processors

Parallel Performance

- Good weak scaling up to 2048 procs
- Strong scaling tests may not involve enough work for all processors
- To date, the focus has been on capability rather than performance
- Performance tests automatically each week



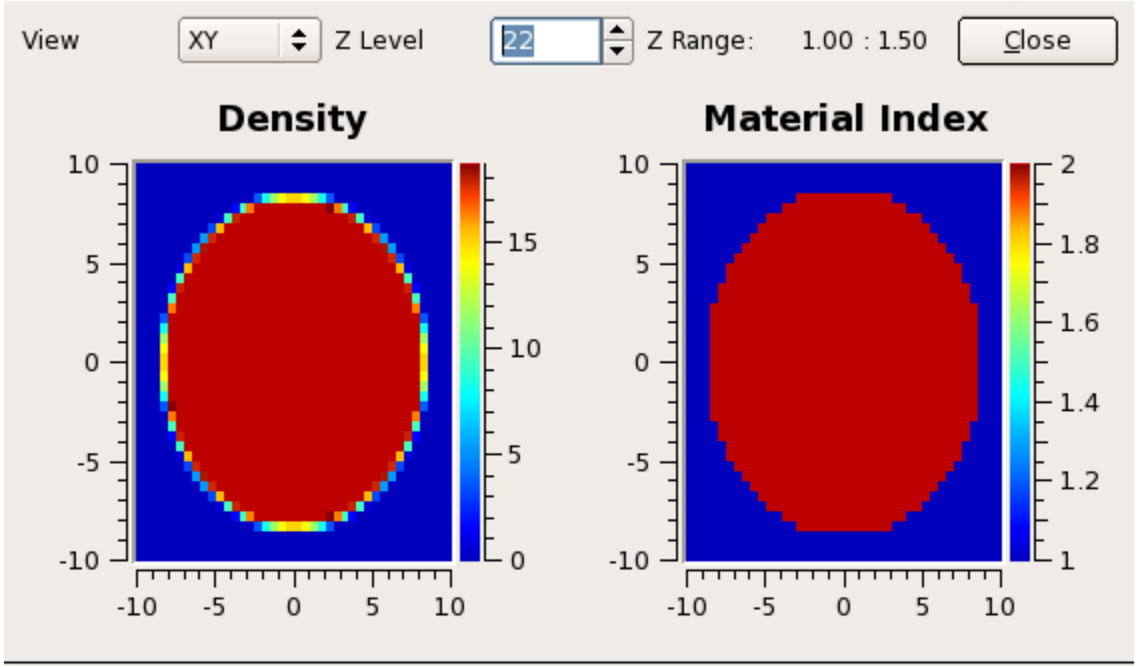


Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- **Miscellaneous Tools/Features**
- Example MCATK Application
- Summary and Future Work

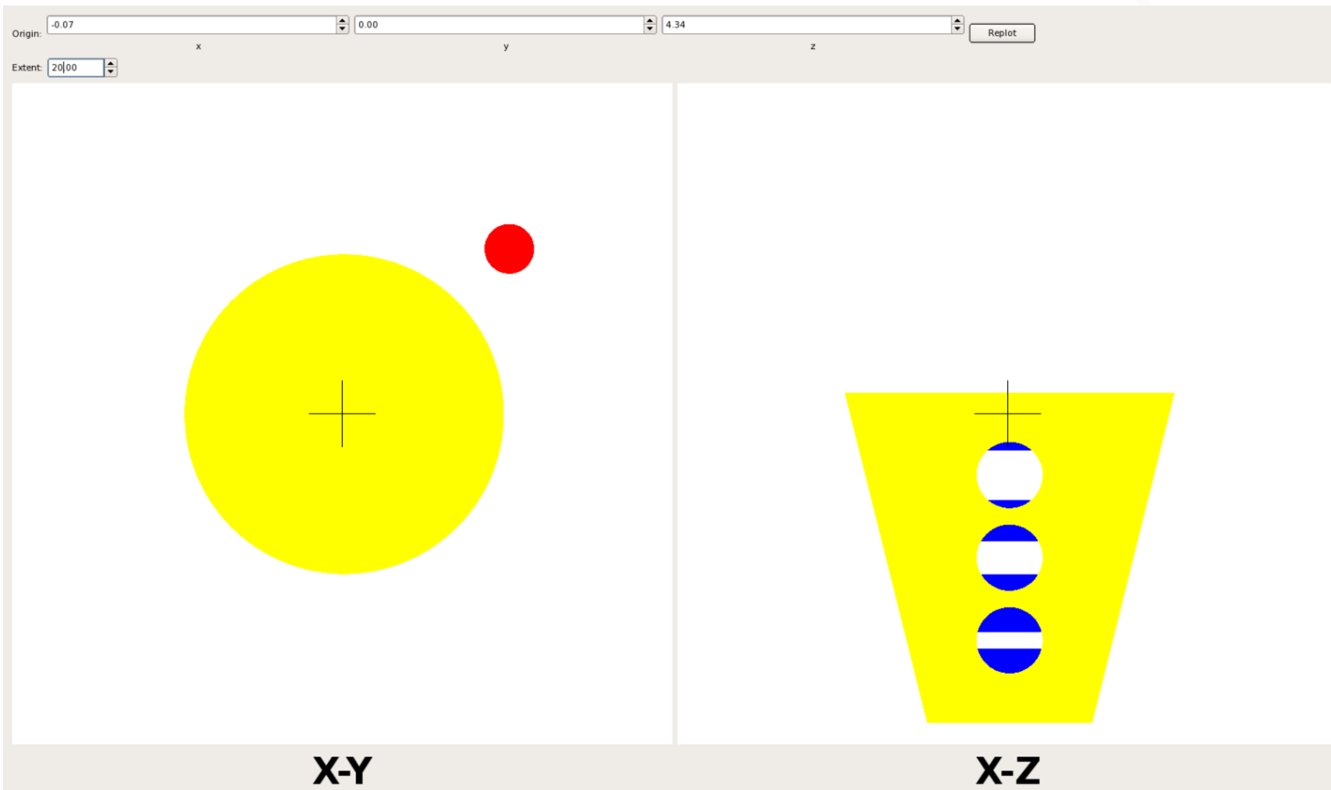
LNK3DNT Viewer

Godiva critical assembly via LNK3DNT file



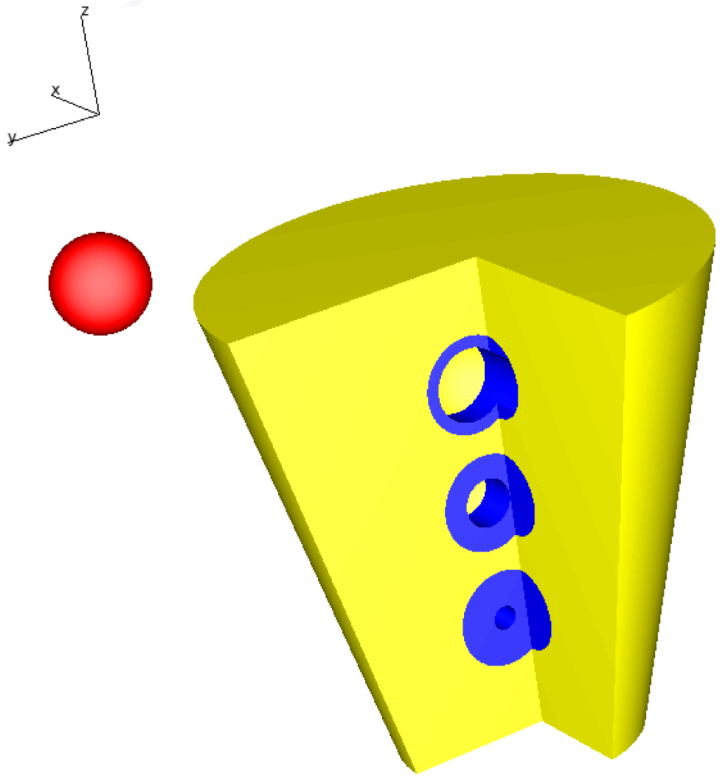
Geometry Plotting Tool

- Similar to MCNP's plotting tool
  - 2 cross-sectional views with a common origin

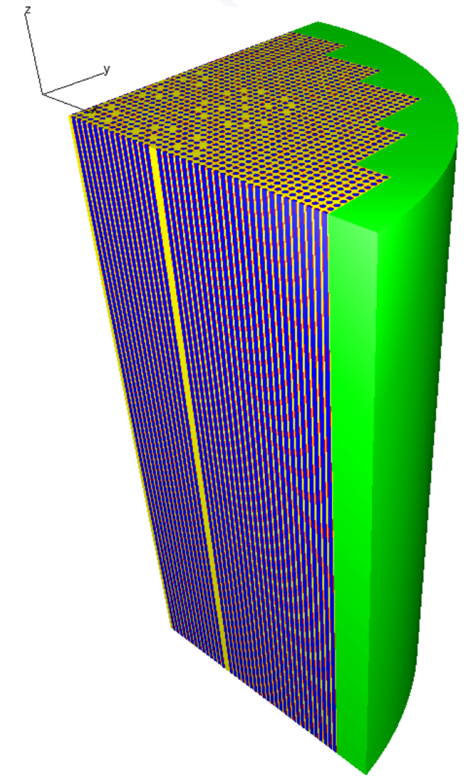


Geometry Rendering Tool

- 3-D Renderer
- Optional chop box for interior visualization

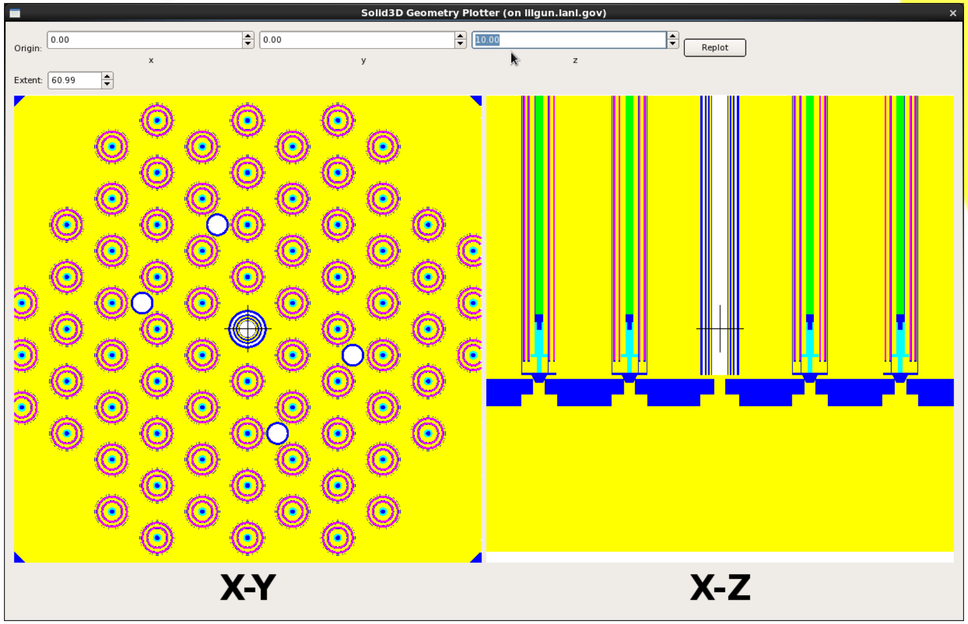


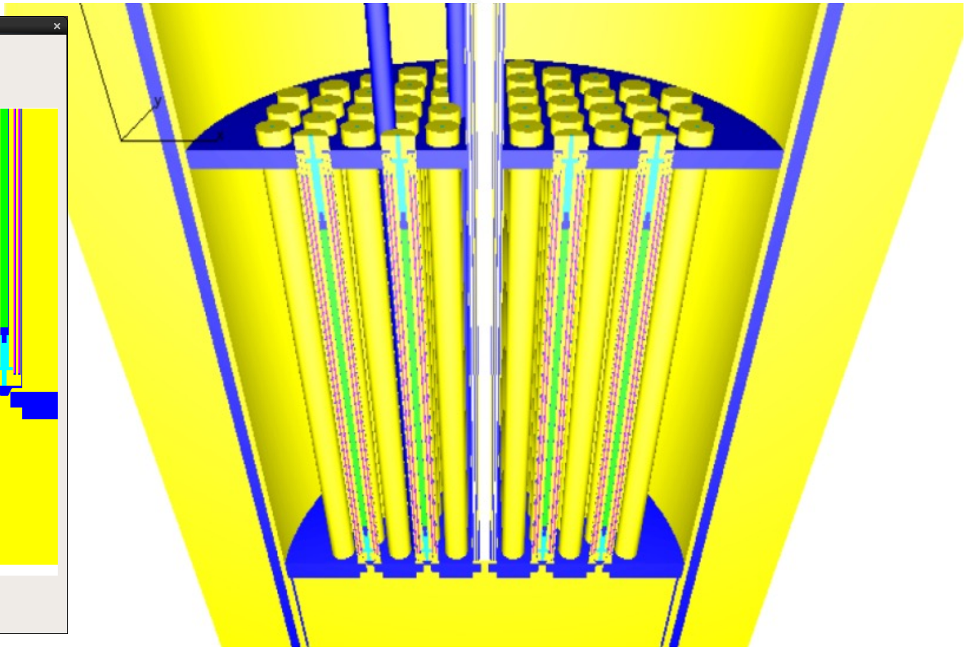
Radiographic Test Object



BAWXI2 Critical Benchmark

Solid Body Geometry Enables Modelling More Complex Systems Than Meshes

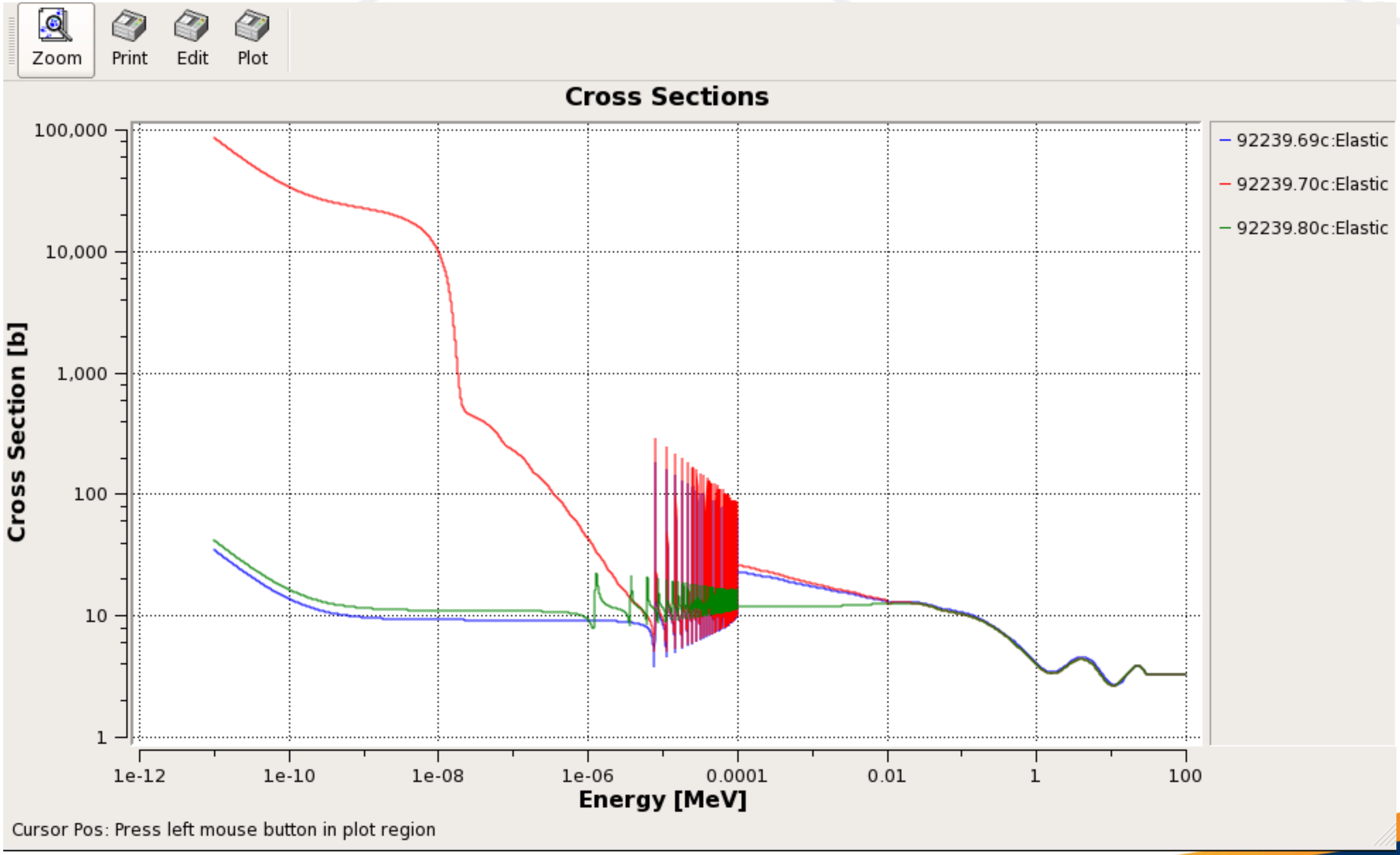




ICT2C3 Critical Benchmark

Work performed by Jesse Giron, ASU

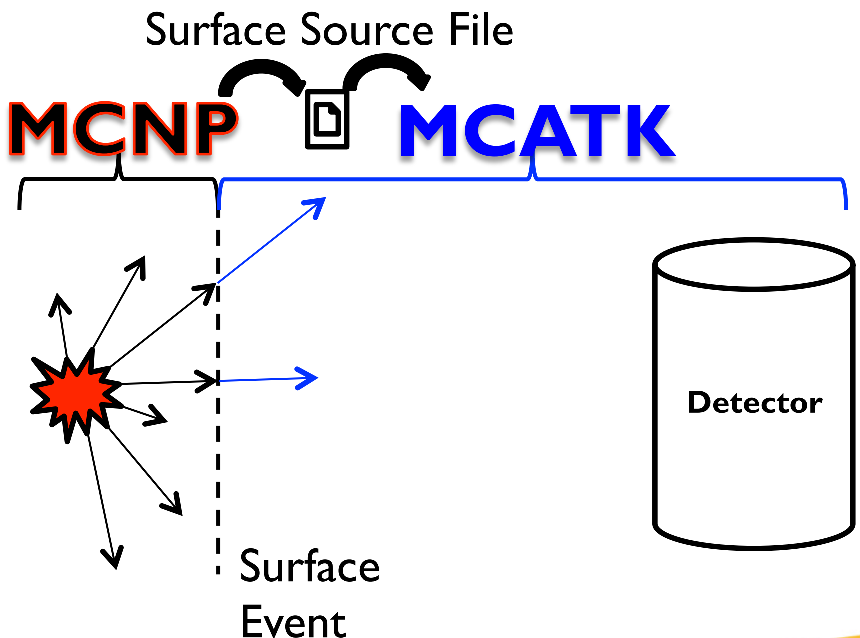
Cross Section Viewer



UNCLASSIFIED

Surface Source Write/Read Capability

- Compatible with MCNP SSW/SSR feature
  - Serves as a link between MCNP and MCATK
- Saves the state of each particle that exits the geometry
  - Could be extended to save particle states at other boundaries (e.g., interior surfaces or time/energy boundaries)
- Particles can be read in and treated as a source



Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- **Example MCATK Application**
- Summary and Future Work

Example Applications

- MCATK GUI driver tool
  - Graphic interface for problem setup
  - Material creation
  - Multiple solution modes:
    - Static k_{eff}
    - Static α
    - Time-dependent α
- Perturbation Applications: setup materials once, then solve a problem on multiple geometries that contain the same materials
- Demo Multiple-Algorithm Solver (shown in the next few slides)
  - Demonstrate how MCATK can be used
  - Use the same materials and geometry with multiple solution algorithms

Multiple-Algorithm Solver

```
namespace mpi = boost::mpi;
int main(int argc, char* argv[]){
    mpi::environment env( argc, argv);
    mpi::communicator world;
    mcatk::ProgramOptions po(argc, argv); // parse command line arguments
    mcatk::Setup driver( po.getInputName(), world.rank() ); // parse input file

    driver.buildMaterials(); // setup all problem materials
    driver.buildMesh(); // setup problem mesh

    driver.solveEigen(); // perform static keff and static alpha calculation
    driver.solveTimeDependent(); // perform time-dependent alpha calculation

    return 0;
}
```

Multiple-Algorithm Solver

```
void Setup::build_materials(){
    std::vector<int> material_id;
    int HEU_id = 0; double HEU_density=18.79;

    ContinuousMat* HEU_mat = new ContinuousMat();
    HEU_mat->addByAtomicFraction( "92235.70c", 0.94 );
    HEU_mat->addByAtomicFraction( "92238.70c", 0.06 );
    multimaterial.add_material( *HEU_mat, HEU_id );
    layer_density.push_back(HEU_density);
    material_id.push_back(HEU_id);

    multimaterial.useTotalNu();
    multimaterial.enableImplicitCapture();
    multimaterial.useThermalScatterTreatment(true);
    pMaterialProperties = new MaterialProperties( material_id, layer_density,
material_id.size() );
}
```

Multiple-Algorithm Solver

```
void Setup::buildMesh(){
    typedef TransportMesh<1,Spherical> Mesh_t;
    Mesh_t::Vertices_t verts;
    // Initialize verts...
    pMesh = new Mesh_t( verts );
}

void Setup::buildSource(){
    pSource.reset( new mcatk::IntrinsicSource("u238") );
    pSource->useIsotropicDirection();
    SourceManager::instance().AddSource( pSource, "U238_SpontaneousFissionSource“ );
}
```

Multiple-Algorithm Solver

```
void Setup::solveEigen(){
    StaticAlpha EigenSolver;
    StaticAlpha::KeffAlgoPtr_t pKeff = EigenSolver.getKeffAlgorithm();
    EigenSolver.setAlphaCycles( numAlphaActiveCycles );
    // set other StaticAlpha options...

    buildSource(); // setup initial source for the first cycle
    AlgorithmAssistant<StaticKeff,TransportMesh<1,Spherical>> helper( *pMesh,
*pKeff, SourceManager::instance() ); // AlgorithmAssistant handles parallelism
and sources

    EigenSolver.setupForParallel( helper );
    EigenSolver.setupAlgorithm( *pMesh, multimaterial, *pMaterialProperties );
    helper.useLoadBalancing();

    // perform static keff and static alpha calculations...
```

Multiple-Algorithm Solver

```
void Setup::solveTimeDependent(){
    TimeDependent TD;
    double start_time=0.0;
    TD.initializeCensusTime( start_time );

    buildSource();
    AlgorithmAssistant<TimeDependent,TransportMesh<1,Spherical>> helper( *pMesh,
TD, SourceManager::instance() ); // AlgorithmAssistant handles parallelism and
sources

    TD.setupAlgorithm( *pMesh, multimaterial, *pMaterialProperties );
    TD.setNTargetParticles( numTDHistories );
    helper.useLoadBalancing();

    // perform time-dependent calculation...
```

Outline

- MCATK Overview
- Development Strategy
- Available Algorithms
- Problem Modeling (Sources, Geometry, Data, Tallies)
- Parallelism
- Miscellaneous Tools/Features
- Example MCATK Application
- **Summary and Future Work**

MCATK: Summary

- MCATK physics:
  - Continuous energy neutron & gamma transport with multi-temperature treatment
  - Static eigenvalue (k_{eff} and α) algorithms
  - Time-dependent algorithm
  - Fission chain algorithms
- MCATK geometry:
  - Mesh geometries
  - Solid body geometries
- MCATK provides:
  - Verified, unit-test Monte Carlo components
  - Flexibility in Monte Carlo application development
  - Numerous tools such as geometry and cross section plotters
- Public availability is a possibility

MCATK: What's Next?

- New physics treatments
  - S(α,β) treatment
  - Unresolved resonance treatment
- Variance Reduction
  - Weight Windows
  - Angular Biasing
- New parallel models
  - Threaded parallel model
  - Many-core parallel model
- New tallies
  - Energy deposition
  - Tallies on a user defined mesh

Where to Learn More

MCATK Citation Paper:

T. ADAMS, S. NOLEN, J. SWEEZY, A. ZUKAITIS, J. CAMPBELL, T. GOORLEY, S. GREENE, R. AULWES, “Monte Carlo Application ToolKit (MCATK),” *Annals of Nuclear Energy*, **82**, 41 (2015).

MCATK at M&C/SNA/MC:

J. SWEEZY, S. NOLEN, T. ADAMS, T. TRAHAN, L. PRITCHETT-SHEATS, “Monte Carlo Applications ToolKit (MCATK): Advances for 2015,” *Proc. M&C 2015*, Nashville, Tennessee, April 19-23, 2015.

S. NOLEN, “Using Fission Chain Analysis to Inform Probability of Extinction/Initiation Calculations with MCATK,” *Proc. M&C+SNA+MC 2015*, Nashville, Tennessee, April 19-23, 2015, American Nuclear Society (2015).

MCATK at ANS:

T.J. TRAHAN, S.D. NOLEN, “A Monte Carlo Algorithm for Fission Chain Analysis of Dynamic Stochastic Systems,” *Trans. Am. Nuc. Soc.*, **113**, 661 (2015).

Contact Info: Jeremy Sweezy, jesweezy@lanl.gov