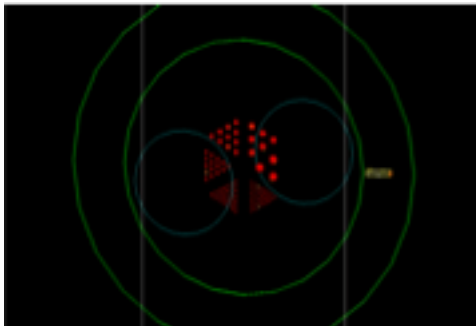
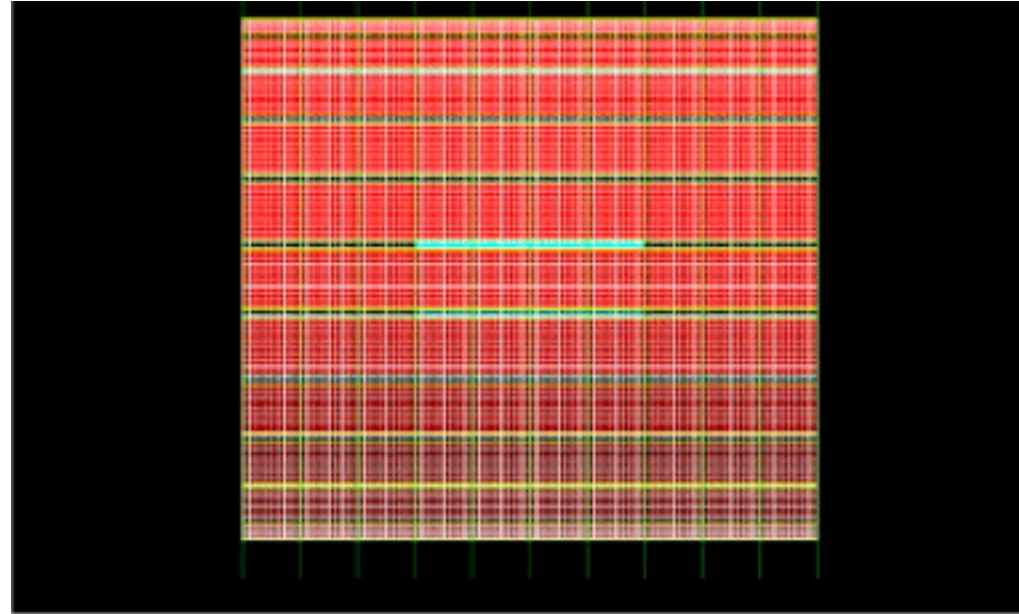
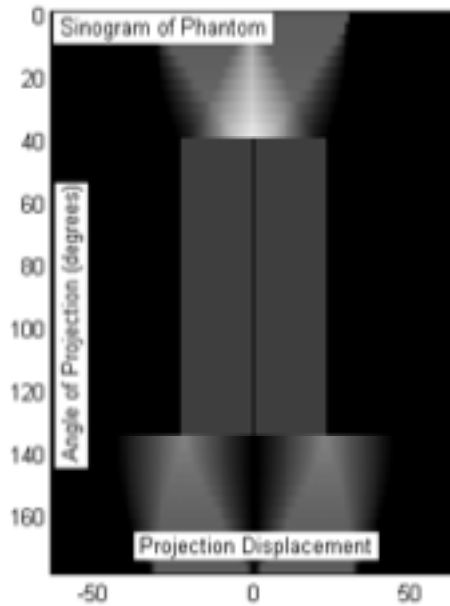


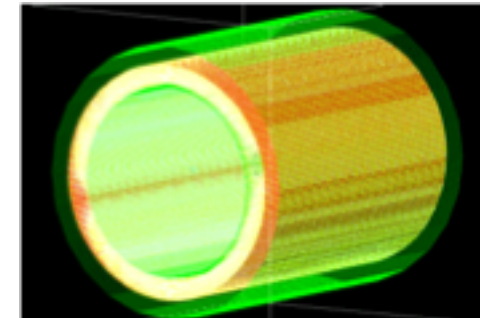
Precision Monte Carlo Measurements using MIXMAX generator



Alexander Howard

*presented by
Gabriele Cosmo*

4th July 2018



Precision Monte Carlo Measurements using MIXMAX generator

1. Introduction
2. Application and requirements for a PRNG
3. Application testing Summary
4. Defaults in Geant4/CLHEP
5. Conclusions

Alexander Howard, Imperial College London

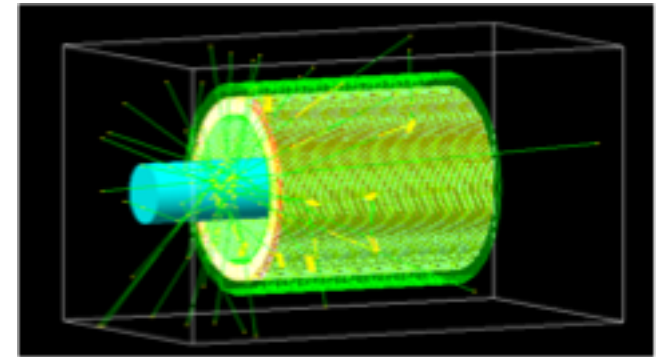
presented by Gabriele Cosmo (CERN)

Summary from Previous Meeting

- Dieharder studies presented last time were **erroneous**
 - Incorrect combination of bits
- Geant4 (CLHEP) now has **MixMax** as it's default
- MixMax Benefits:
 - Quality of random numbers
 - High granularity, un-correlated, lack of artefacts
 - No seeding issues, particularly with multi-threading
 - Performance: faster than other generators (e.g. Ranlux)

Application Testing

- Results are presented at the application level:
 - Not just **MixMax**, but **CLHEP**, **Geant4** and **the application code**
- One “run” (only 1 second of experimental data...) corresponds to more than 9 years of sequential CPU time
 - CSCS (Swiss Computing Centre) “Mönch” cluster
 - More than 700 nodes with a selection of 32GB, 64GB, 256GB RAM
 - Each node has 20 cores with hyperthreading
 - <http://www.cscs.ch/computers/moench/index.html>
- 500 million events create a lot of random numbers
- Critical is the timing distribution between events (time of the annihilation)
 - Otherwise randomness over/underestimated
 - Scattering and other flight time effects have bias



Testing of MIXMAX

- “Simple” problem: continuously uniform (Poisson) distribution between event times
- All tests performed at the Geant4 level in MT mode
MixMax → CLHEP → Geant4
- Multi-threaded mode
 - Seeds distributed between workers and master
- Bunches of events per thread
 - Either 4 or 40 threads per node were tested
 - Risk: if problems were observed it's a coupled system
 - But everything seems okay



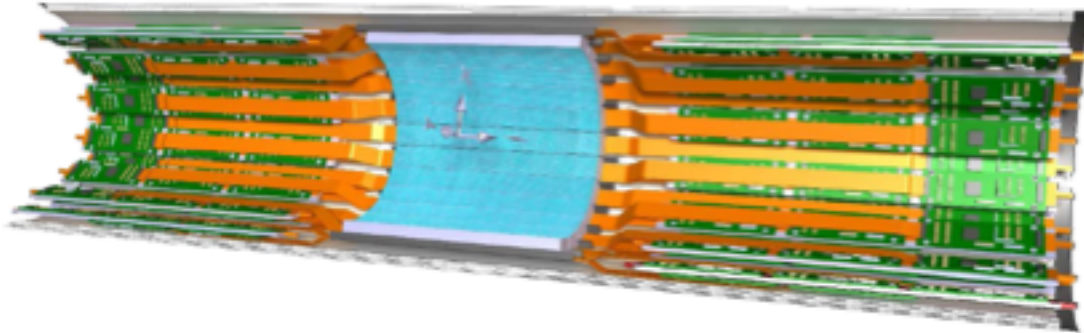
The SAFIR project (goal)

ETH zürich

SAFIR: Evaluation of the Performance of ASICs at High Count Rate

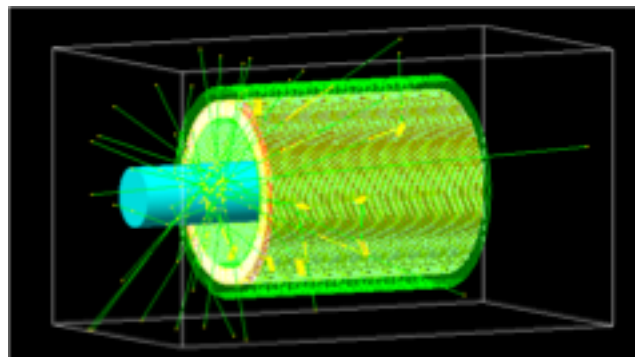
SAFIR – Small Animal Fast Insert for MRI

Robert Becker, Alfred Buck, Chiara Casella, Simon Corradi, Günther Dissertori, Jannis Fischer, Alexander Sinclair Howard, Mikiko Ito, Parisa Khateri, Werner Lustermann, Ulf Röser, Geoffrey Warnock, and Bruno Weber

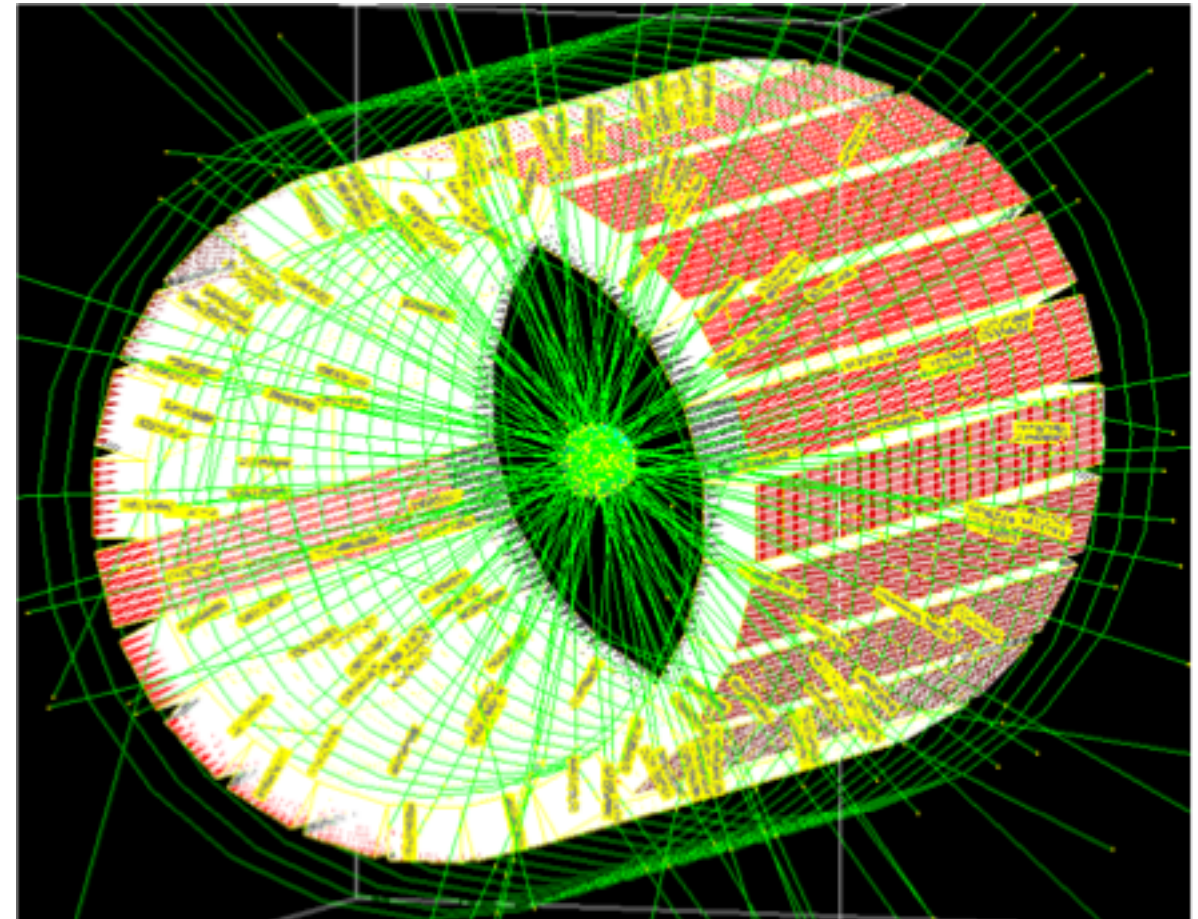


SAFIR

PSMRI 2016 Conference — Mikiko Ito | 23 May 2016 | 1

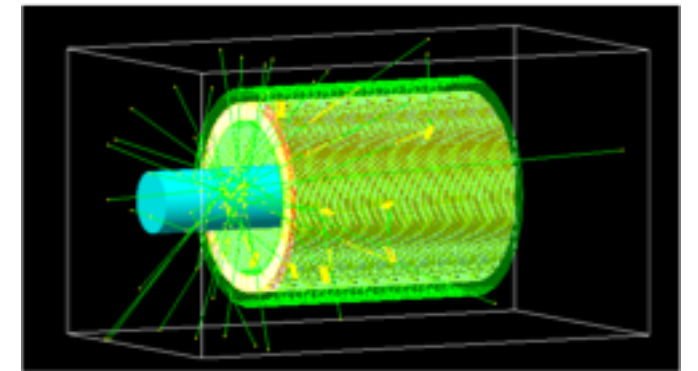


Fast (temporal) Small Animal PET with
simultaneous MRI
 $>100\text{MBq}/\text{cm}^3$



Application Testing – PET scanners

- PET relies on the simultaneous measurement of annihilation gammas
- Random contribution and estimation is **critical**
 - Image quality, Signal:Noise and normalisation (activity quantification)
- Due to the (unusually) high activities in our proposed scanner the Coincidence Time Window (CTW) needs to be very short
- Performance of detector needs to be optimised
 - CTR, scintillator, geometry
- Continuity (granularity) in event times is **crucial**



Application PRNG Sensitivities

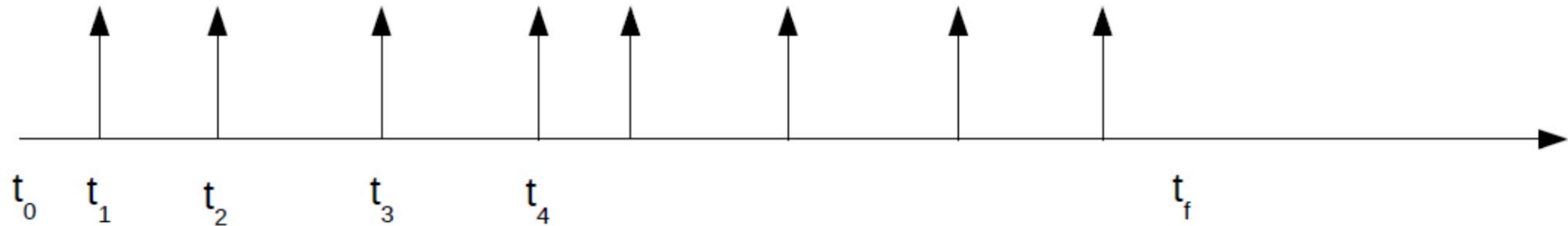
- Seeding between runs → `/dev/urandom`
 - **Unnecessary for MixMax!**
- Granularity vs. time interval
- Multithreading (seeding, warm-up, distribution between workers)
- Geant4 Radioactive Decay Module:
 - Internal memory/thread-race
 - Events are biased to occur within a measurement window

Radioactive Decay Time Window

- Set decay time window to 1s
- Record the decay time of ^{22}Na
- Geant4 RDM uses `G4UniformRand()` across the time window and weighted by the half-life
- Essentially flat for ^{22}Na (>2 year half-life)
- Alternative approach would be to time-order events, however, might lead to unforeseen structure and requires modification of Geant4 model (RDM)

Time between events

- Should follow the radioactive decay law (only one isotope)
- Poisson probability distribution



t_i follows a uniform distribution between t_0 and t_f

$\Delta t_i = t_{i+1} - t_i$ follows a exponential distribution between 0 and t_f

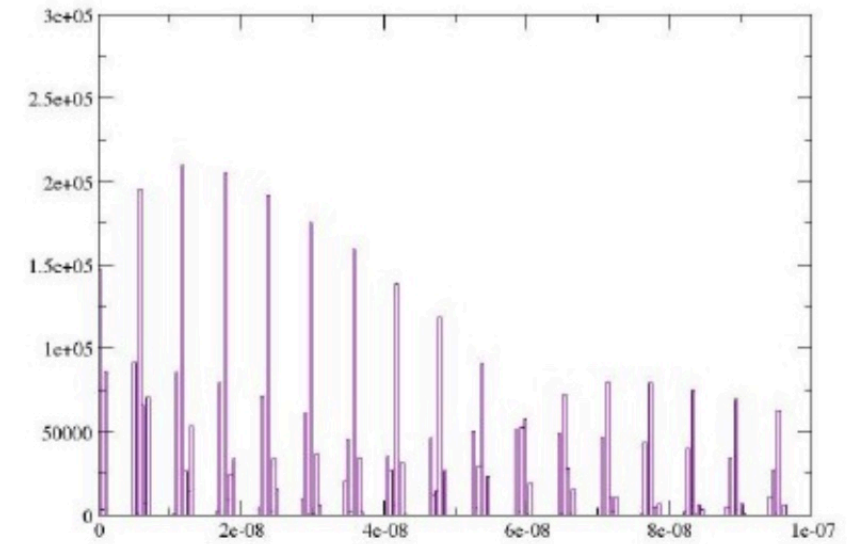
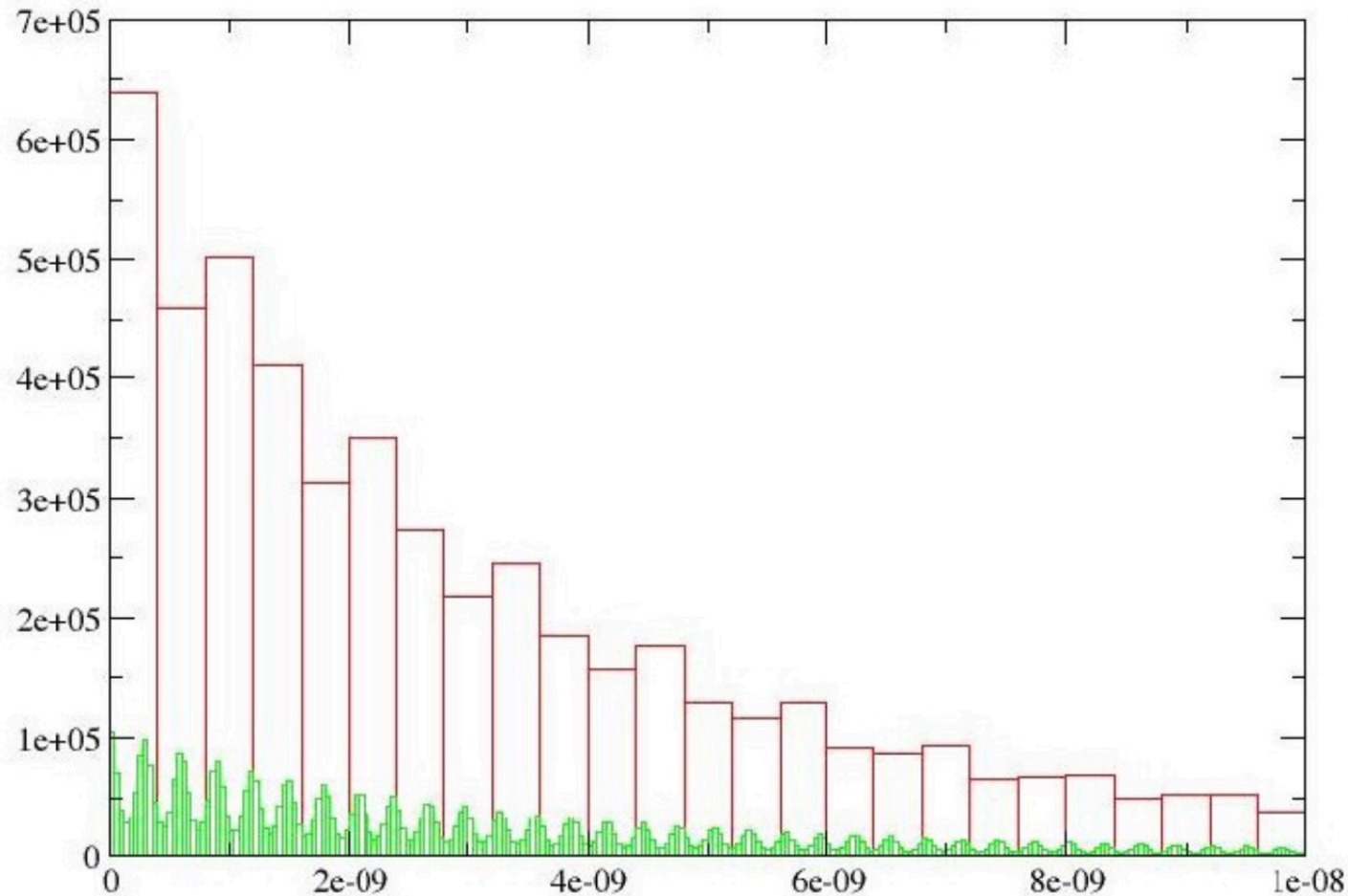
Expectation

- 1 second simulation with a 64-bit generator should give a granularity (minimum spacing between events) of **$1e^{-19}s$**
- **465.7ps** for a 32-bit generator (signed integer)
- The results for any generator (32- or 64-bit) should be a smooth exponential distribution

Correlations vs. Uncorrelations

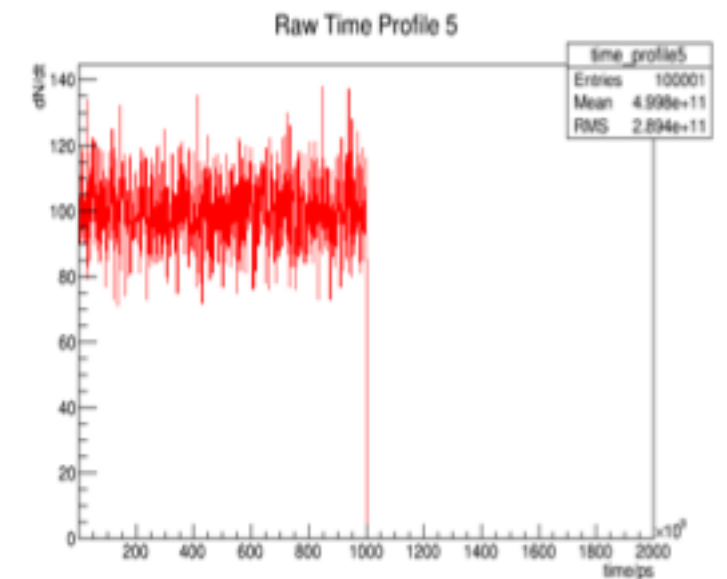
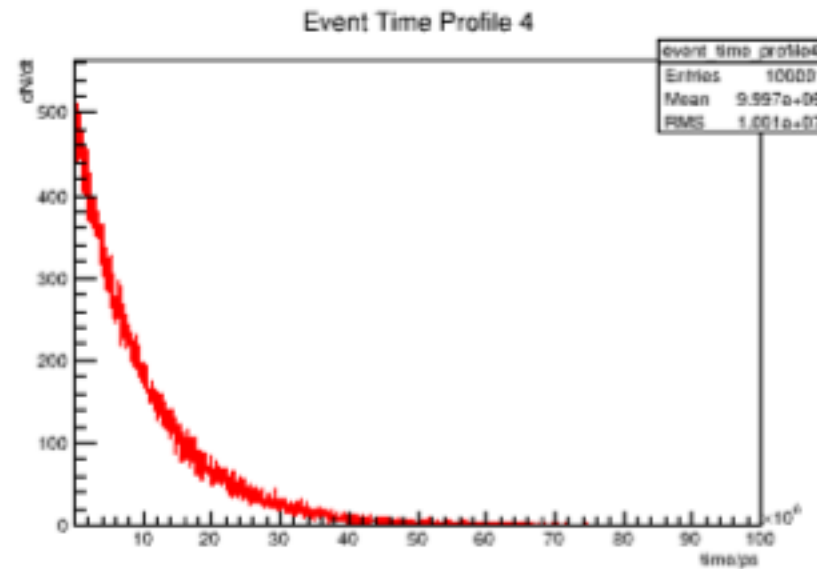
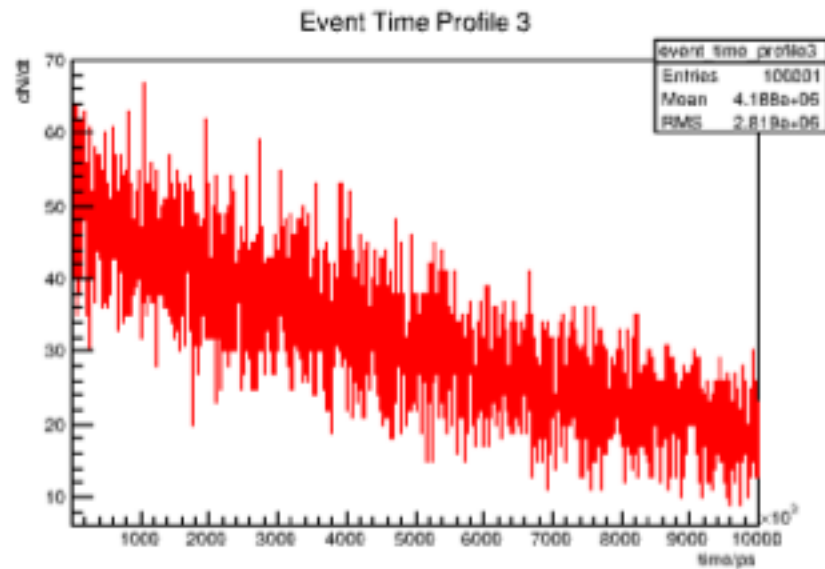
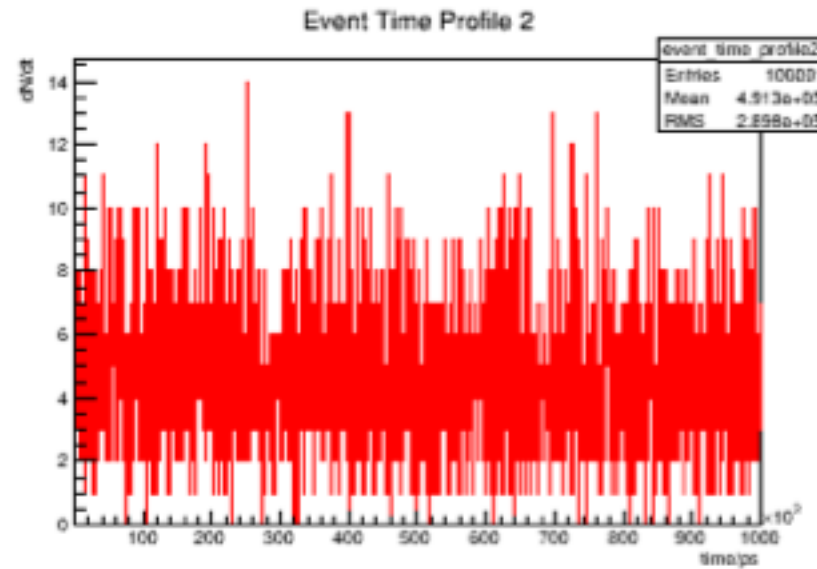
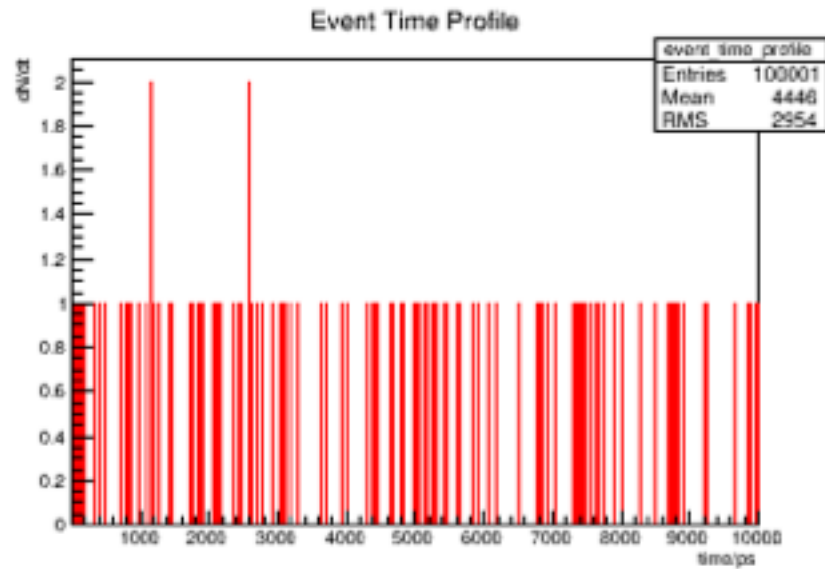
- In PET imaging lines of response (LOR) are generated by creating coincidence pairs of 511 keV photopeak events (positron annihilation)
- To estimate performance and correctly combine multiple hits it is crucial to be able to determine uncorrelated (random) pairs as well as true lines of response
- This problem is particularly acute for high activities (acquisition/imaging times)
 - Useful for high-throughput (cost)
 - Required for time propagation (functional/4-D images)
- The following plots show the situation with two random number generators:
Ranlux64 and **MixMax**

Granularity and gross structure an issue with “Ranlux64”



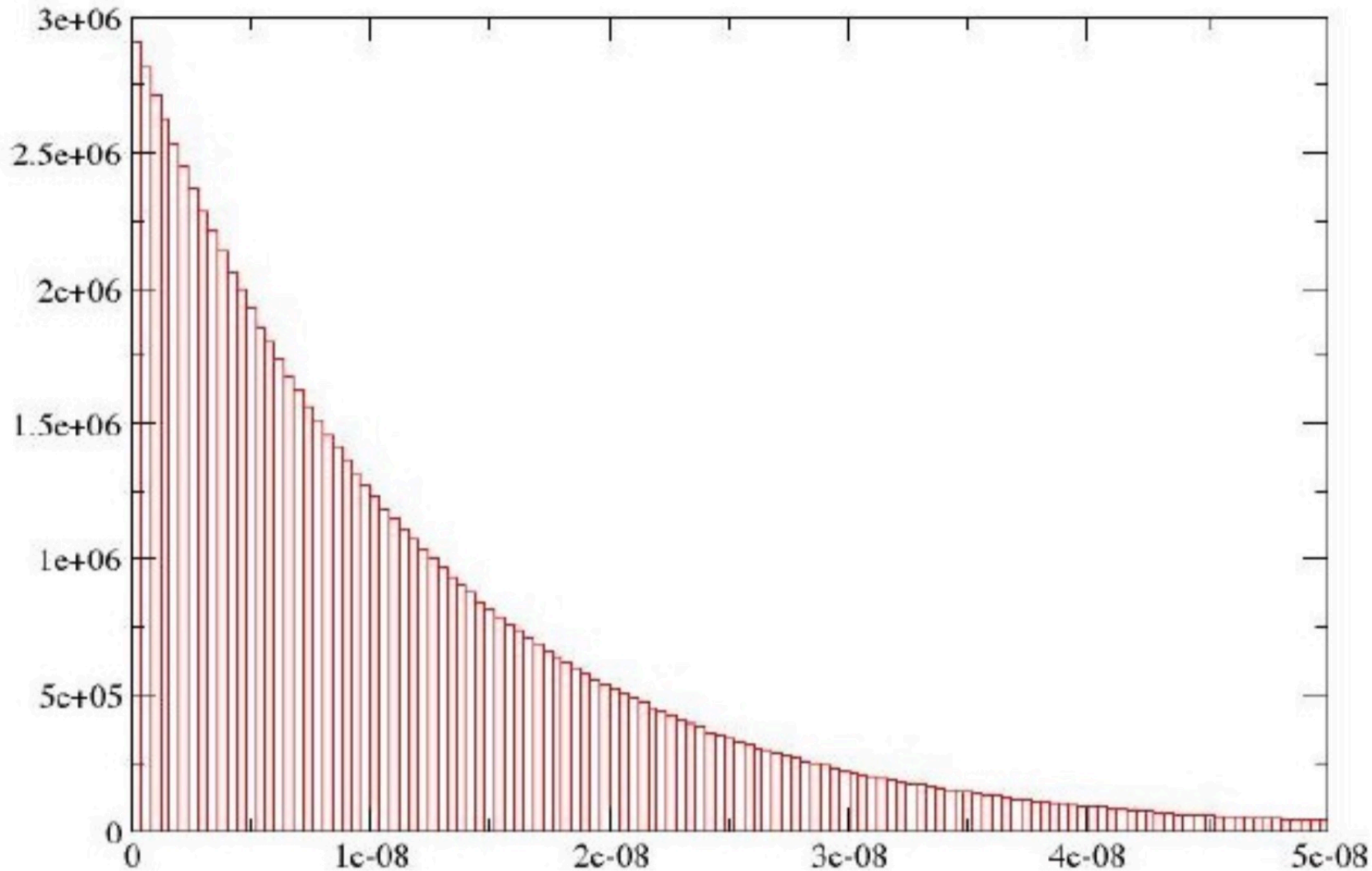
- A seeding issue?

100k G4UniformRand() events - MixMax



MixMax – 500 million events

- Totally smooth distribution!



Summary

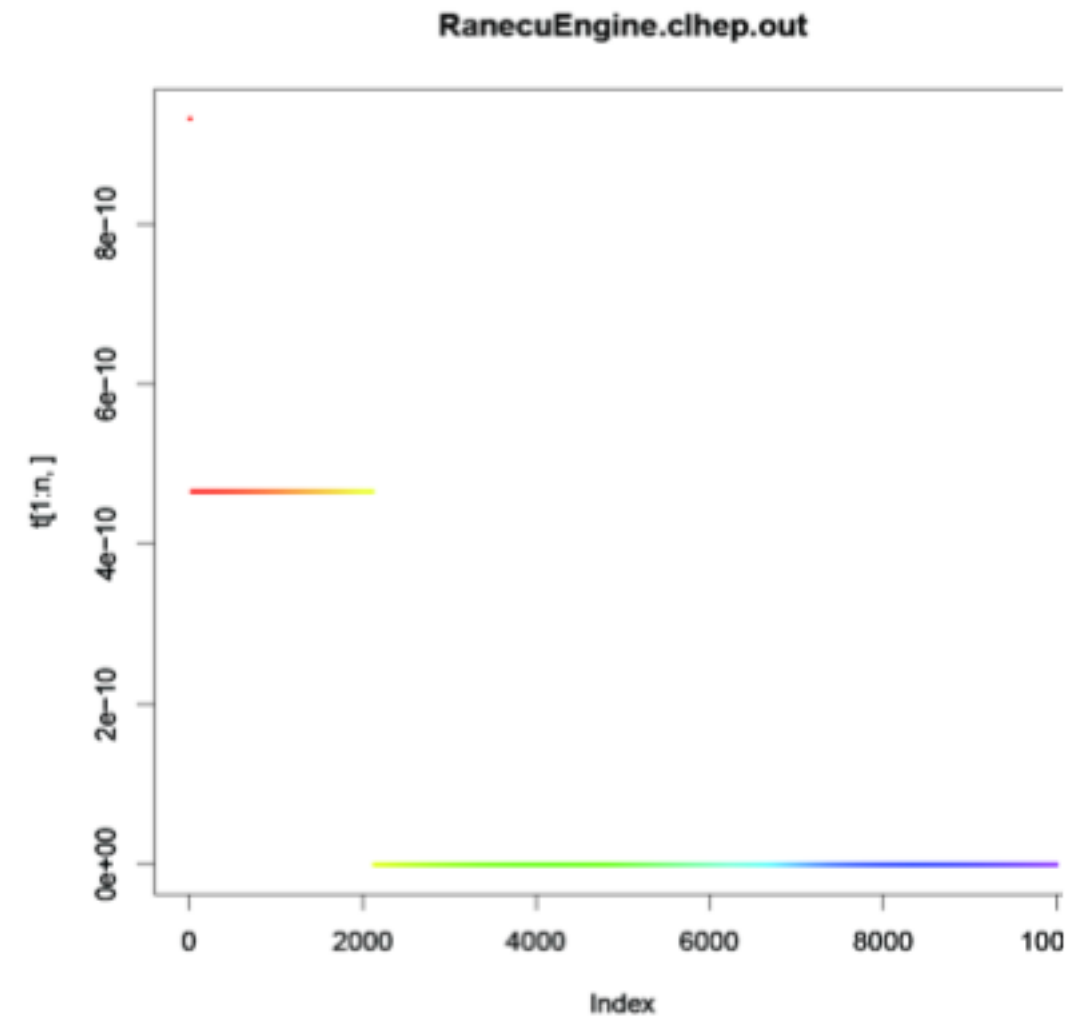
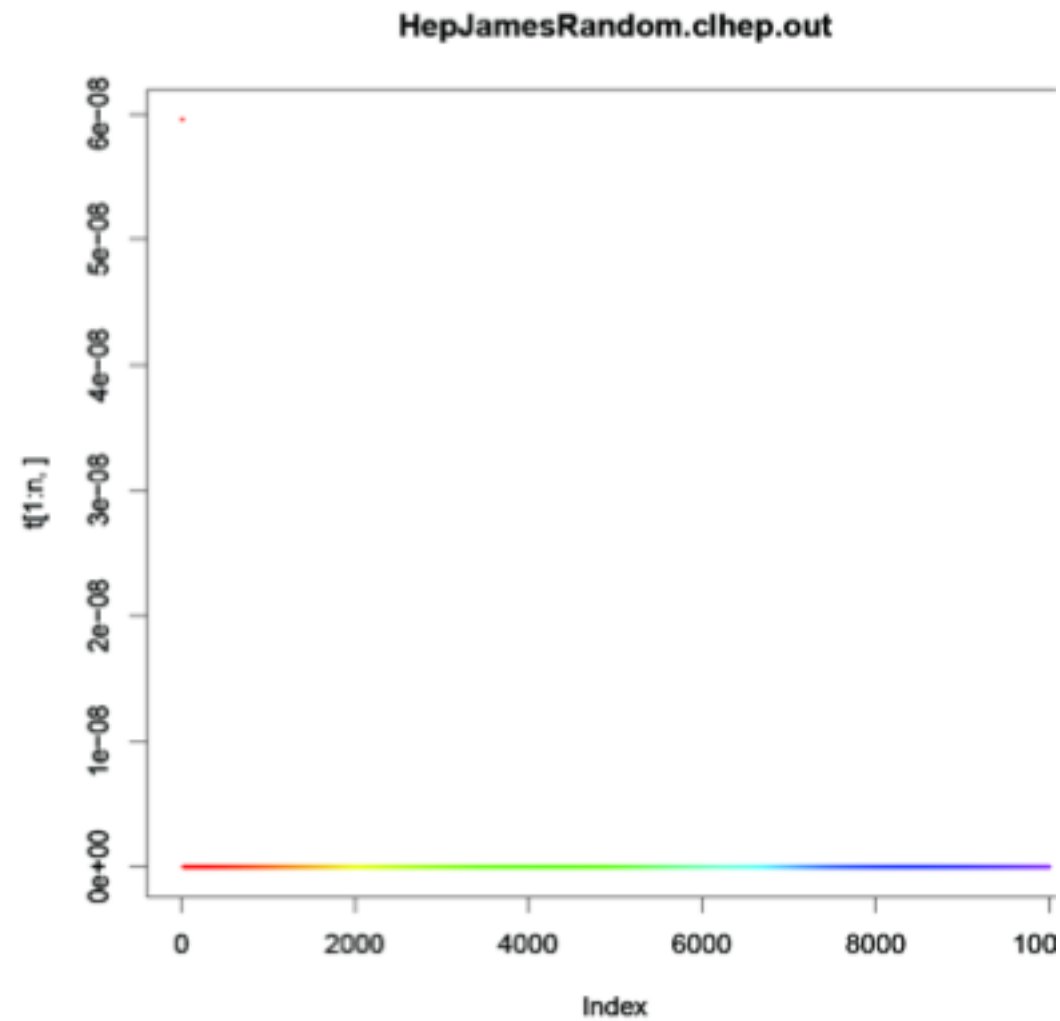
- **MixMax** works very well for this application
- No other generator gave these results in Geant4
- Possible to estimate random correctly
- Crucial for understanding and developing detector timing performance as well as analysis algorithm investigation
- An extension in Geant4:
 - Use run and event number to generate the seeds
 - Makes storing and reproducing random engine status trivial

Conclusions

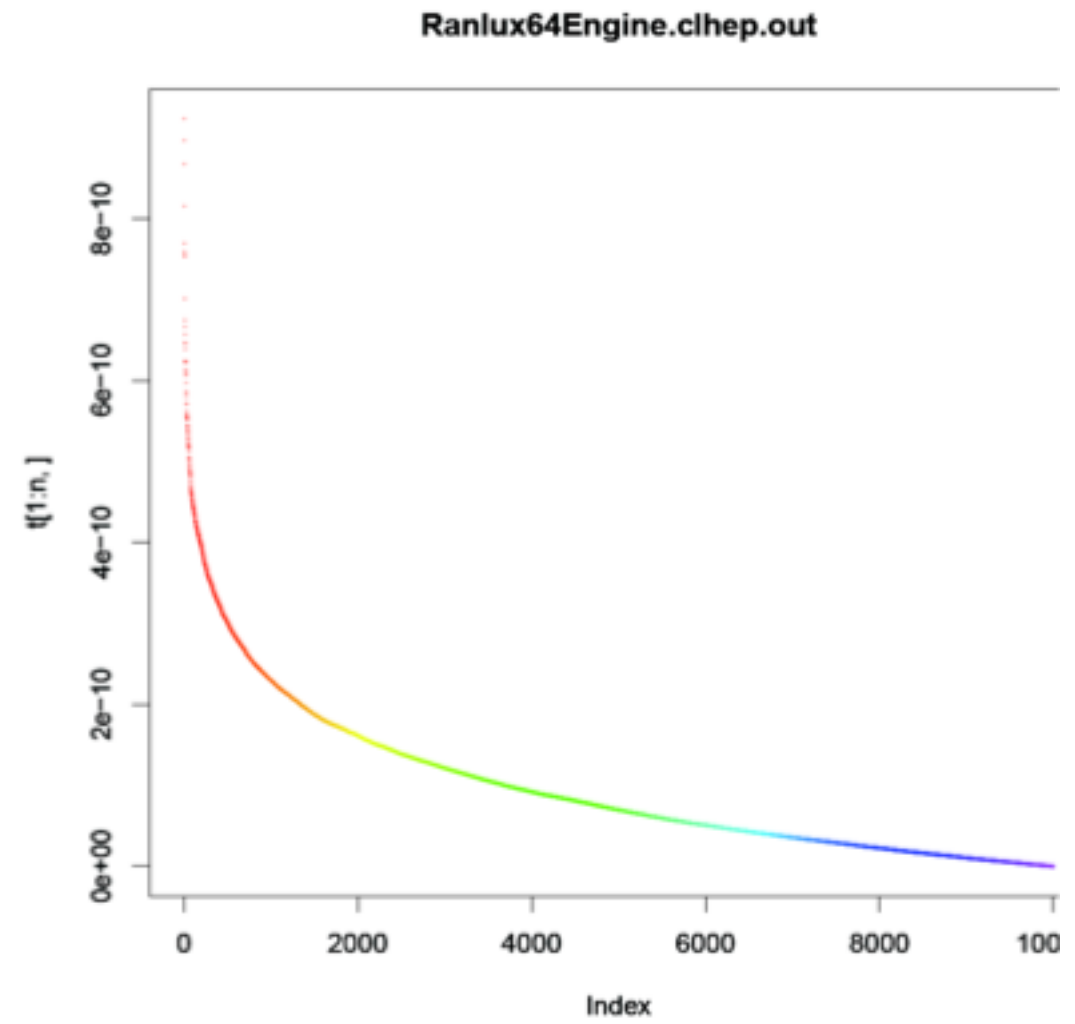
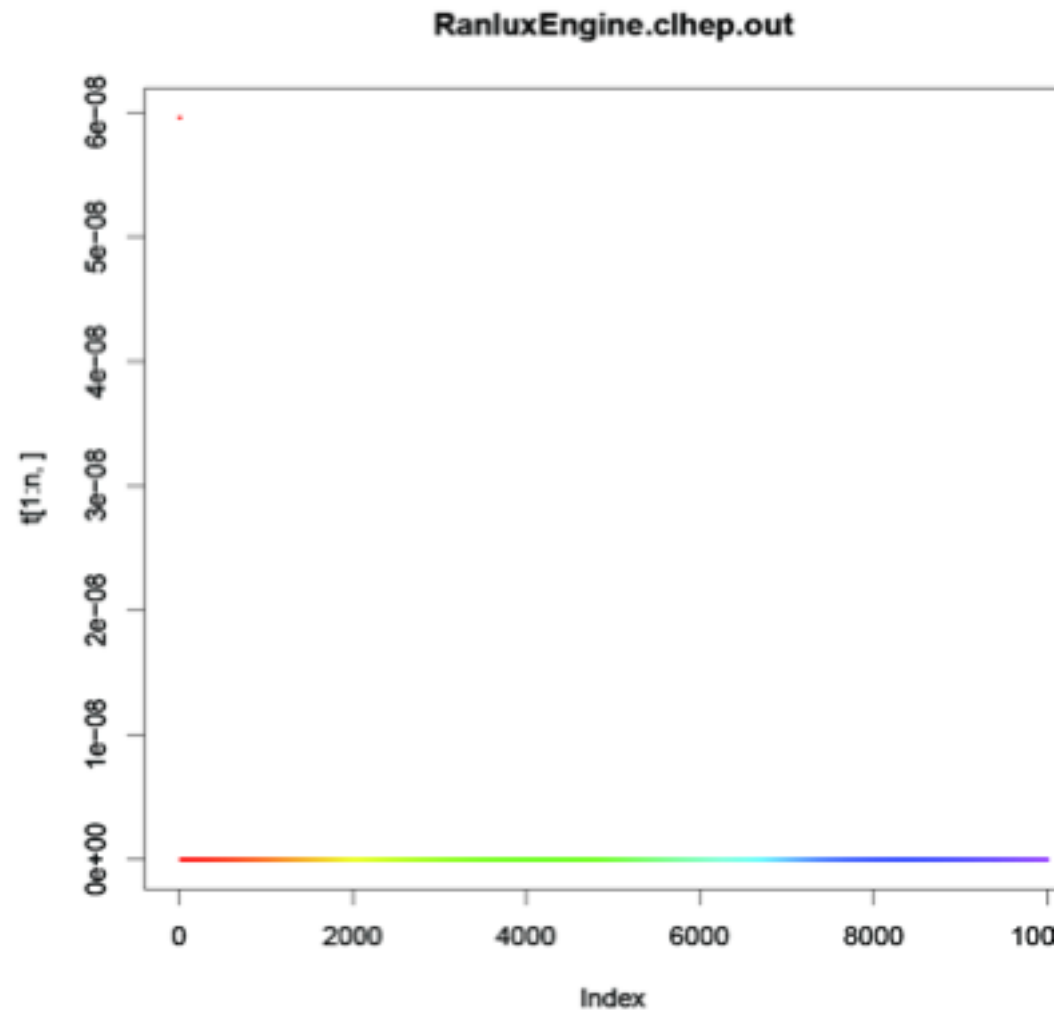
- MixMax solved a number of problems with random numbers in Geant4 applications
- Especially important for **high** statistics **precise** random estimations in PET detectors (decay-time pile-up).
- Seeding issues are also solved by MixMax, particularly important for multi-threading marshalling
- Seed storing and reproducing could be simplified if a combination of event and run number gave the seed (**only** works with MixMax)
- Since release 10.4 of Geant4, the included CLHEP default uses MixMax

Backup/Previous slides

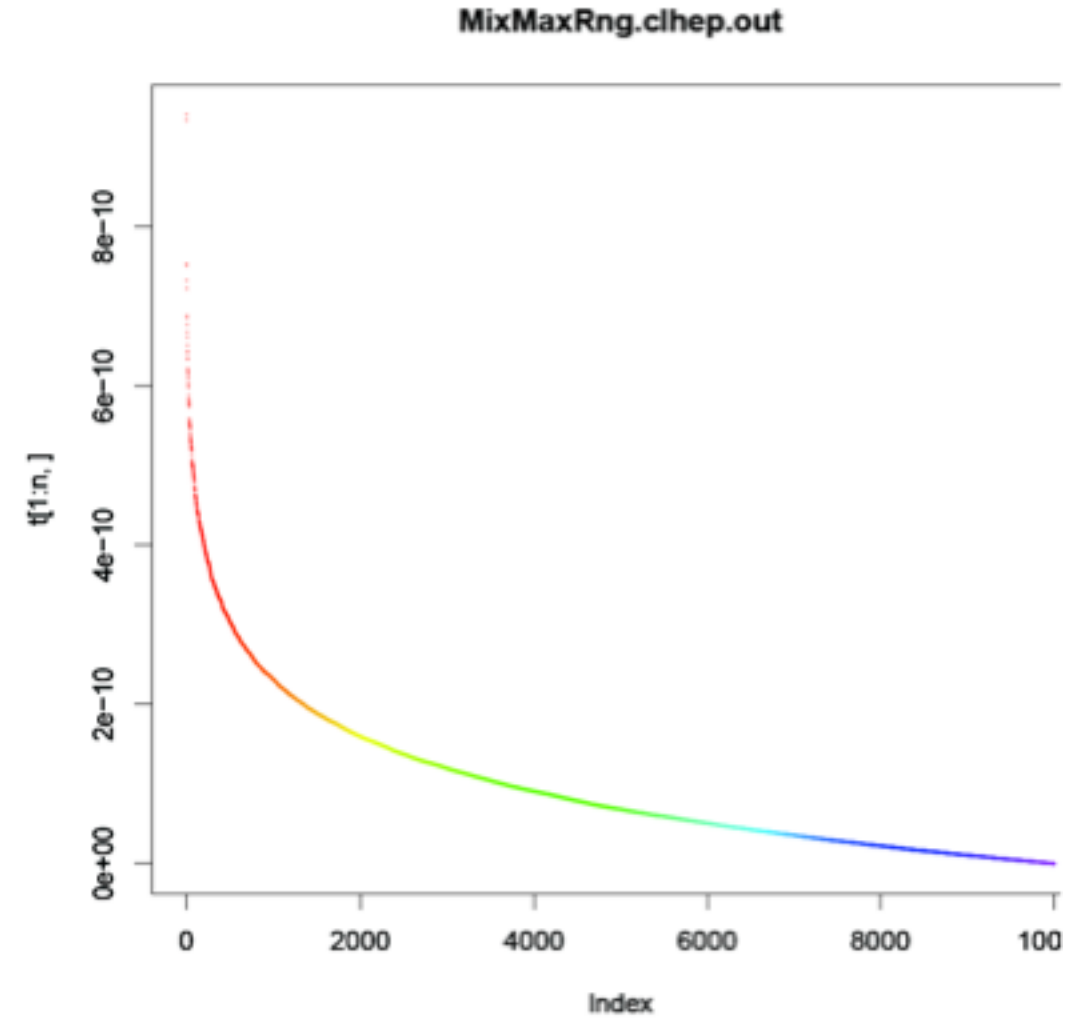
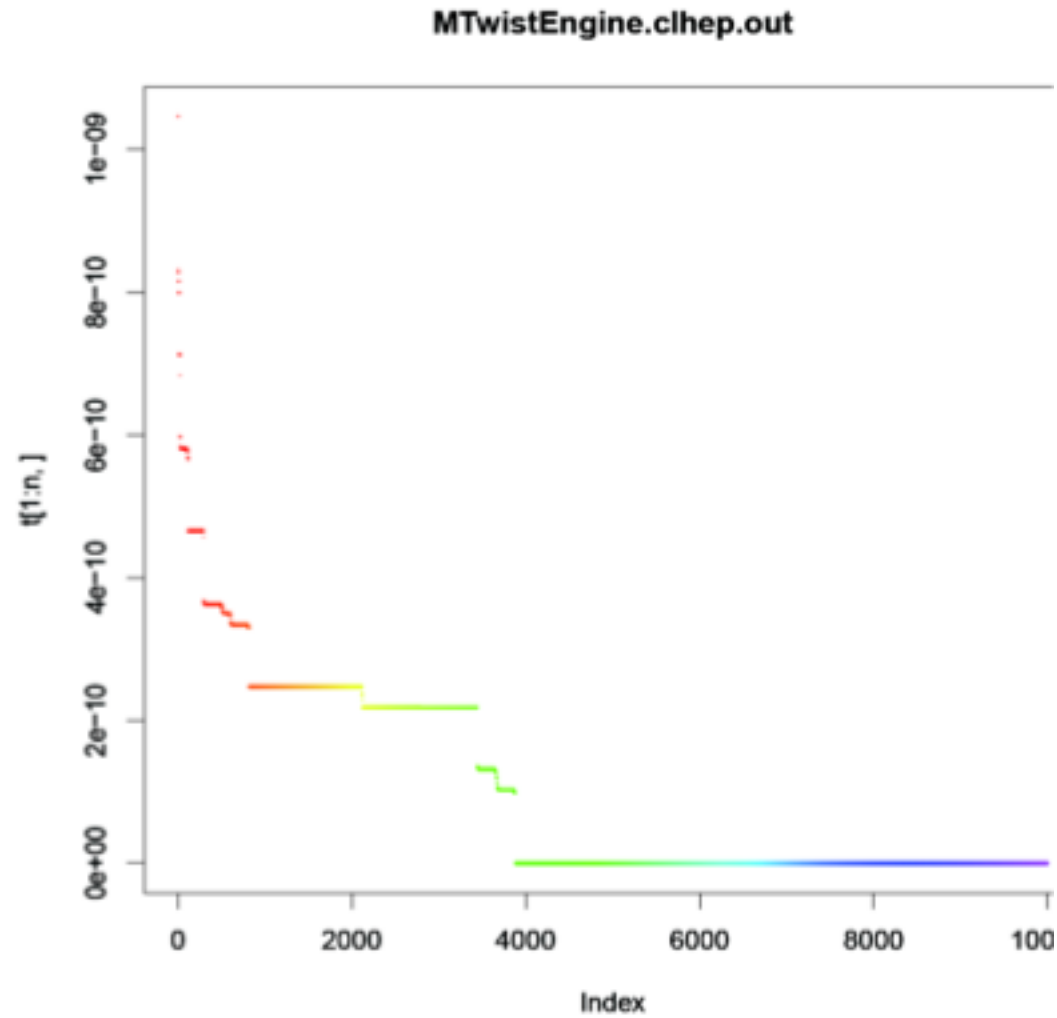
CLHEP Tests (from Kostas)



CLHEP Tests (from Kostas)



CLHEP Tests (from Kostas)



Random Number Testing

Create random numbers in a flat distribution:

- 1) Unit test: RNG.flat()
- 2) Unit test: G4UniformRand()
- 3) G4UniformRand() inside an application
- 4) Time distribution of decay window biased particles from RDM

Outputs:

- 1) Measure time for 100000 trials
- 2) Determine time interval between events (granularity)
 - Record created times (1s interval)
 - Sort in increasing times
 - Plot the distribution of time intervals between events (necessary requirement for random event pile-up studies)

PRNGs tested:

- 1) Ranecu
- 2) Ranlux
- 3) Ranshi
- 4) Ranlux64
- 5) Mersenne Twist
- 6) MixMax

Geant4-10.2-ref02

CLHEP 2.3.1.1 (November 2015)

Geant4-10.2-ref06

CLHEP 2.3.3.0