PPC (2020 update)

PPC Notes
==========
October 16, 2020 D. Chirkin

- Added back CUDA version of the code. It now supports full functionality of the OpenCL code. The CUDA version is 15-35% faster than the OpenCL version.

  This is not yet integrated into the CMake build system, you may need to compile the CUDA library by hand (replacing the OpenCL library) and possibly add load(cudart) statements

- Significant code re-working for running on systems with heterogeneous sets of GPUs. An improved default workload distribution (proportional to advertised resources)

- Full documentation is at https://docs.icecube.aq/combo/trunk/projects/ppc/.
  A number of documentation tickets/requests were addressed, which includes tilt, cable, hole ice, DOM oversize discussion. Some of the newest features (mainly for single LED data simulation) are not yet exposed there.

PPC (2020 update)

PPC Notes
==========
October 16, 2020 D. Chirkin

- All icecal-approved ice models and many others are supported. This includes BFR-v1 (birefringence-based anisotropy description), EMRM (absorption-based), all SPICE 3.X (scattering-based) and other legacy models (Lea, Mie, 2.X, 1, AHA, WHAM, etc.) It is possible to configure and combine multiple anisotropy effects simultaneously.

- Split configuration directory is possible (feature added by Juan Carlos): PPCTABLESDIR and ICEMODELDIR directories, and PPCHOLEICE file. Falls back to PPCTABLESDIR if unused.

PPC (2020 update)

PPC Notes
==========
October 16, 2020 D. Chirkin

- Some functionality for kernel sharing to clsim (by JVS)
- DirectFit improvements: reading bad time windows, total charge for QSAT check (Tianlu)
- PPC project also hosts llh/DirectFit (flasher-fitting and cascade/track reconstruction software) and BFR (detailed photon propagation through birefringent crystals)
- Added script for reading in flasher data, which is now processed by standard IceCube software (for all ice models from SPICE 3.X on) (Tianlu).
- More precise flasher LED model, including angular and timing distributions, positioning inside DOM, photon blocking by internal structure (some), harness belt and cable.

Running ppc

```
#muongun
time python $I3_SRC/simprod-scripts/resources/scripts/muongun.py --UseGSLRNG --emin 1e3 --emax 1\
e6 --nevents 100000 --outputfile muons.i3 --gcdfile /cvmfs/icecube.opensciencegrid.org/data/GCD/\
GeoCalibDetectorStatus_AVG_55697-57531_PASS2_SPE_withScaledNoise.i3.gz --no-propagate-photons

#clsim
time python $I3_SRC/simprod-scripts/resources/scripts/clsim.py --UseGSLRNG  --inputfile muons.i3\
 --outputfile mcpes-clsim.i3 --gcdfile /cvmfs/icecube.opensciencegrid.org/data/GCD/GeoCalibDetec\
torStatus_AVG_55697-57531_PASS2_SPE_withScaledNoise.i3.gz --no-PropagateMuons --UseGPUs --no-Run\
MPHitFilter

real      54m42.949s
user      40m56.495s
sys       13m43.625s

#ppc OpenCL
time python $I3_SRC/simprod-scripts/resources/scripts/ppc.py --UseGSLRNG  --inputfile muons.i3 -\
-outputfile mcpes-ppc-opencl.i3 --gcdfile /cvmfs/icecube.opensciencegrid.org/data/GCD/GeoCalibDe\
tectorStatus_AVG_55697-57531_PASS2_SPE_withScaledNoise.i3.gz --no-PropagateMuons --no-volumecyl

real    36m58.612s
user    8m5.803s
sys     0m32.680s

#ppc CUDA
time python $I3_SRC/simprod-scripts/resources/scripts/ppc.py --UseGSLRNG  --inputfile muons.i3 -\
-outputfile mcpes-ppc-cuda.i3 --gcdfile /cvmfs/icecube.opensciencegrid.org/data/GCD/GeoCalibDete\
ctorStatus_AVG_55697-57531_PASS2_SPE_withScaledNoise.i3.gz --no-PropagateMuons --no-volumecyl

real    29m23.894s
user    8m24.614s
sys     0m30.172s
```

CLsim

PPC OpenCL

PPC CUDA

25% faster than OpenCL version

Juan Carlos pointed me to some awesome scripts that are very simple to run and similar between clsim and ppc

PPC

Update 2017

PPC Release Notes
==========
August 31, 2017 D. Chirkin

Release V02-05-00

- merged CUDA and OpenCL versions into single OpenCL version of code
- removed multiple define statements (permanently enabling optional code)
- multiple code optimizations, improved method of copying of photon data to the GPU
- fully merged particle/flasher code into single method, flasher parameters now passed in a particle segment
- added Poisson/binomial sampling of generated number of photons (from mean prediction)
- added cable shadow and DOM tilt code and corresponding flasher-fitted files
- removed obsolete scripts and most of the ice directories; one (most recent and comprehensive) ice directory remains
- added documentation that explains run time parameters and data files

Most recent code is now once again on the trunk. Release V02-05-00 coming soon.

All features of OpenCL, CUDA, and branch versions (i.e. monopole and SMP) are now on trunk.

Only one ice model is maintained in resources/ice, the fully-featured latest SPICE, with flasher-based angular sensitivity, cable shadow and DOM tilt files.

The emitting segments now have “photon number” in them, which means less data transmitted to the GPU. They are unpacked in the first pass into the GPU memory

- → this is actually a few percent faster
- → no longer sampling photons in multiples of 10 to save memory
- → allowed to merge flasher/particle interface, flasher configuration now passed in the “photon segment”

Have my own wait code that avoids spinning CPU.

Only the latest light yield parametrizations (by Leif R.) are left in the code.

Geometry and RDE tables will replace the input from GCD, when present, will produce a warning. Fixed a problem in using RDE from icetray with wavelength dependence file in the ice directory.

New ppc functionality

Depth-dependent anisotropy, passed in the icemodel.dat file (2 new columns)

DOM tilt and cable position, configurable with dx.dat and cx.dat files

Poisson sampling of the photon number determined by light yield parametrizations

Binomial downsampling of CLStep photons (when ang. sens. curve always below 1)

Direct hole ice is now in the OpenCL version, always compiled in

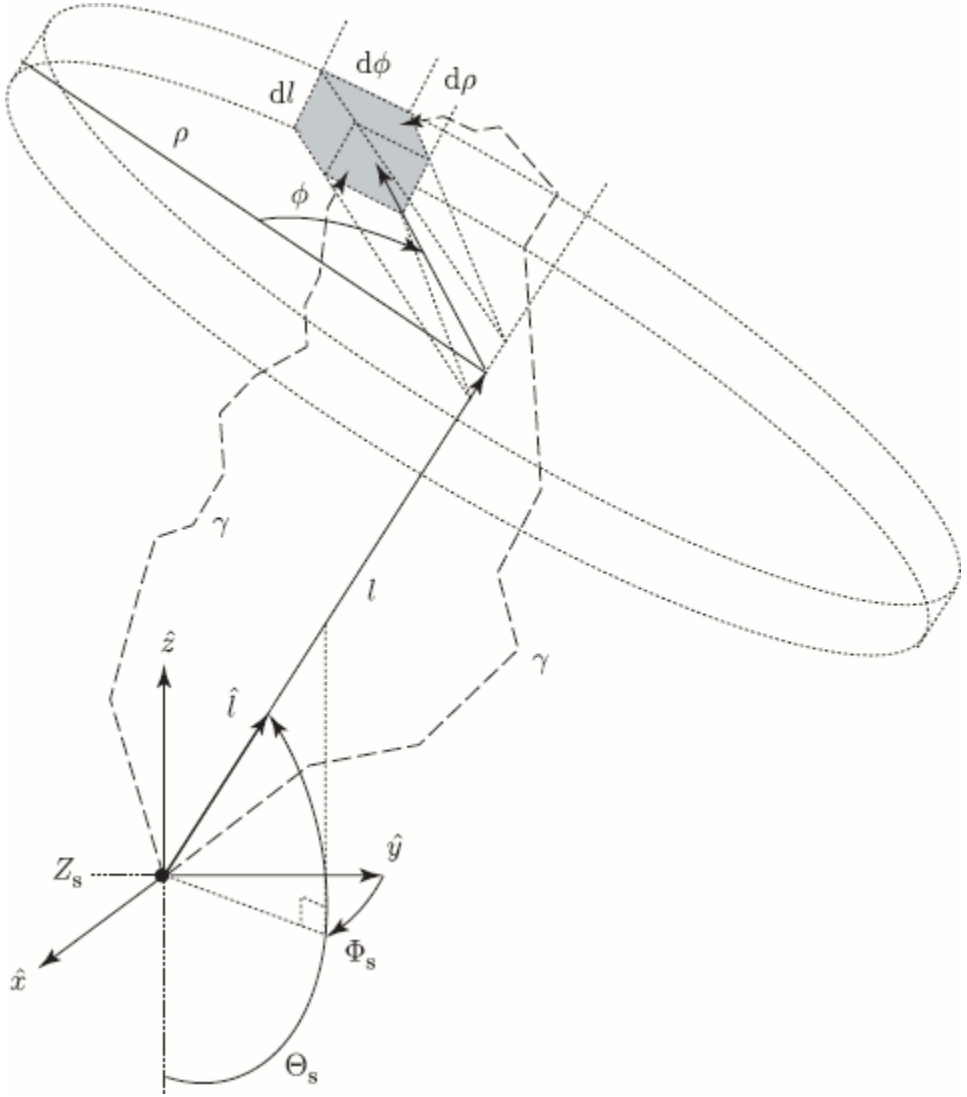
Increased number of wavelength bins passed to GPU to 512 (up from 32)

Full documentation of all parameters and input files

PPC

Update 2017

Photon tracking with tables



- First, run photonics to fill space with photons, tabulate the result

- Create such tables for nominal light sources: cascade and uniform half-muon

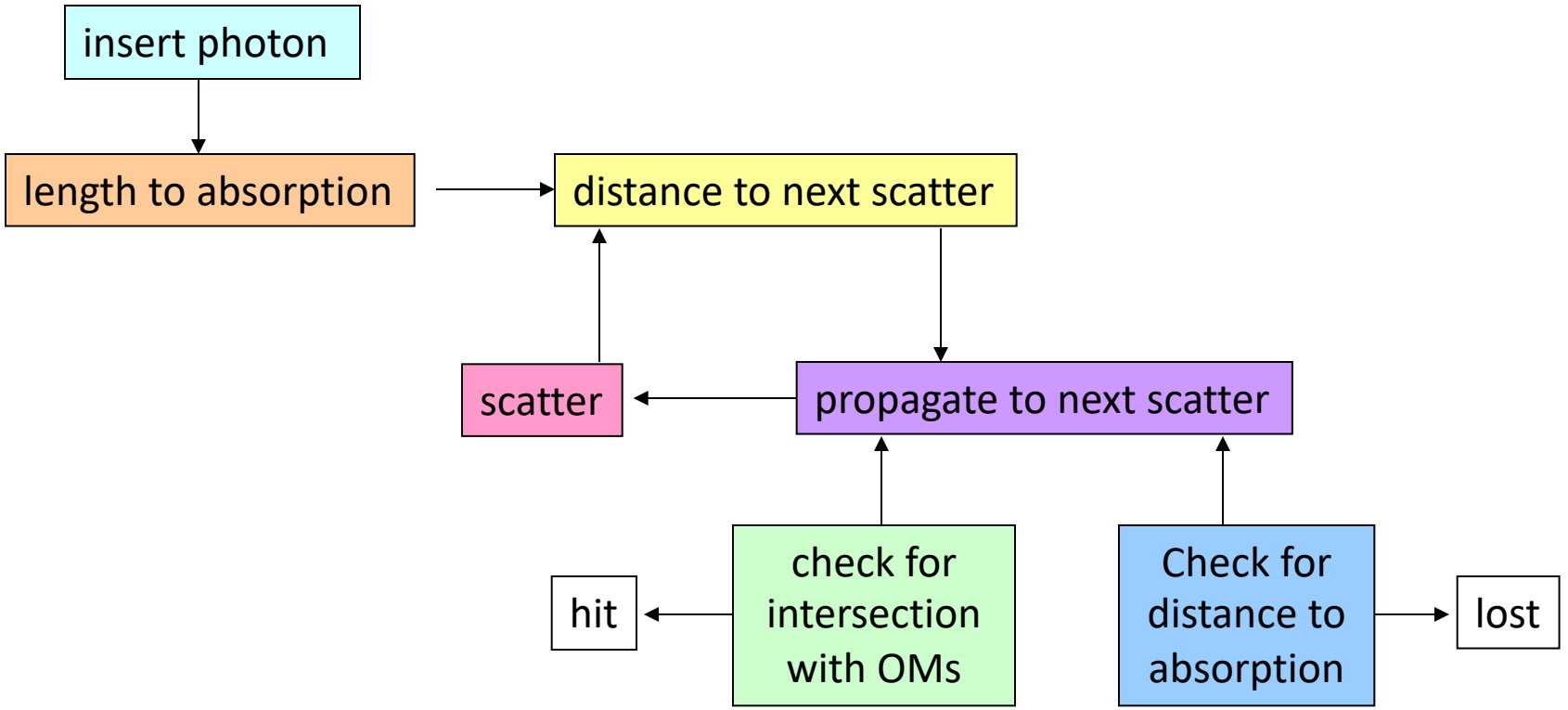
- Simulate photon propagation by looking up photon density in tabulated distributions

→ Table generation is slow
→ Simulation suffers from a wide range of binning artifacts
→ Simulation is also slow! (most time is spent loading the tables)

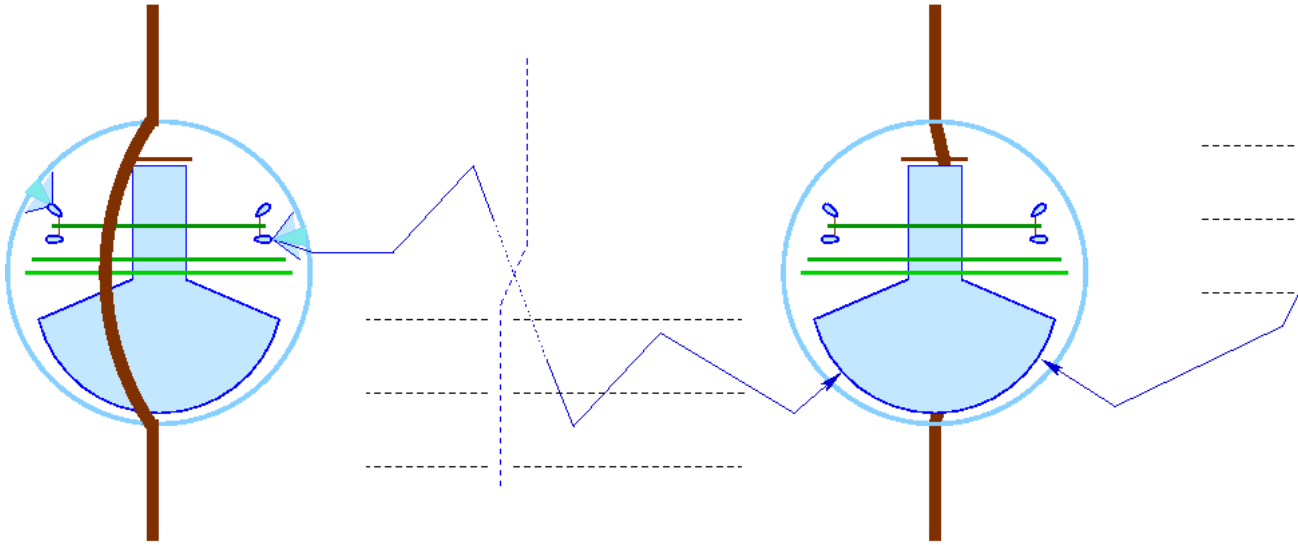
Direct photon tracking

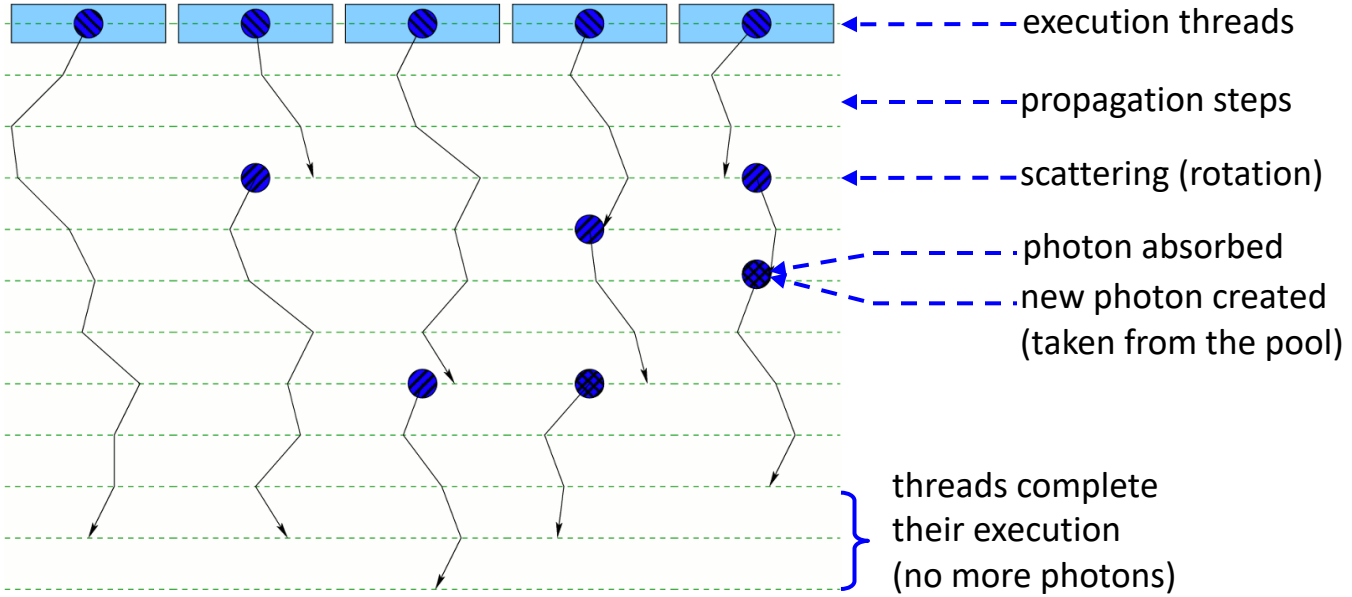
photon propagation code (ppc) or OpenCL simulation (clsim)

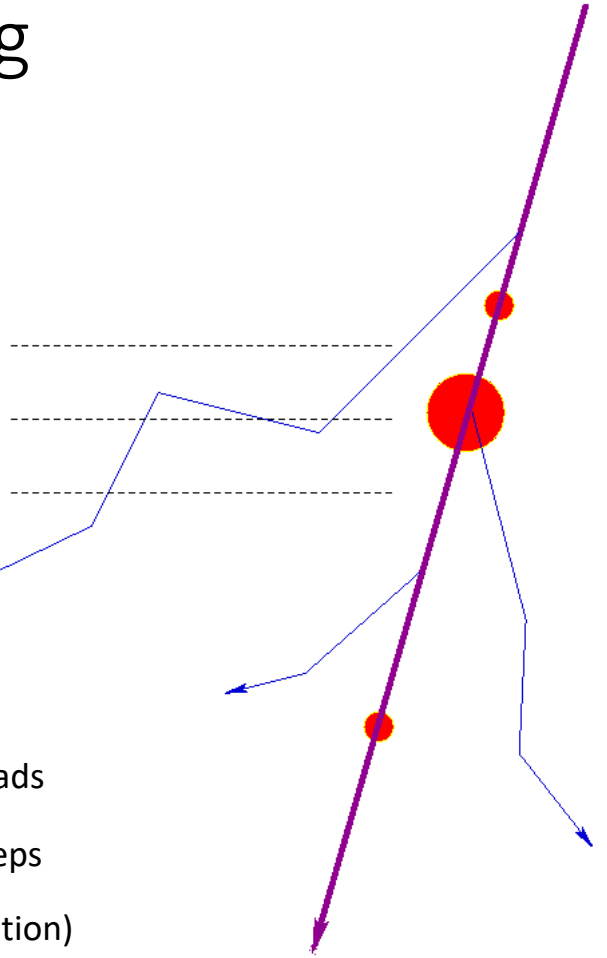
propagate photons directly when needed



Simulation: Direct photon tracking







Same code used for both flasher simulation/ice calibration and muon/physics data simulation

New ppc

Most recent code has been moved to the “dima” branch.

Currently only OpenCL version is there (likely to remain like that, w/o the CUDA code)

→ have my own wait code that avoids spinning CPU

Many optional code option defines are now gone, all of the code is always compiled.

→ monopole code is always “in”

Only one ice model is maintained in resources/ice, the fully-featured latest SPICE, with flasher-based angular sensitivity, cable shadow and DOM tilt files

The emitting segments now have “photon number” in them, which means less data transmitted to the GPU. They are unpacked in the first pass into the GPU memory

→ this is actually a few percent faster

→ no longer sampling photons in multiples of 10 to save memory

→ allowed to merge flasher/particle interface, flasher configuration now passed in the “photon segment”

New ppc functionality

Depth-dependent anisotropy, passed in the icemodel.dat file (2 new columns)

DOM tilt and cable position, configurable with dx.dat and cx.dat files

Poisson sampling of the photon number determined by light yield parametrizations

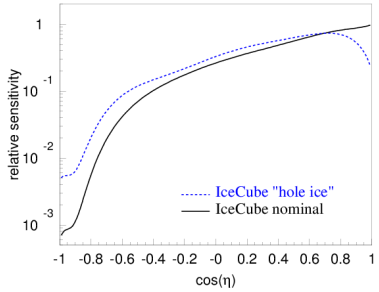
Binomial downsampling of CLStep photons (when ang. sens. curve always below 1)

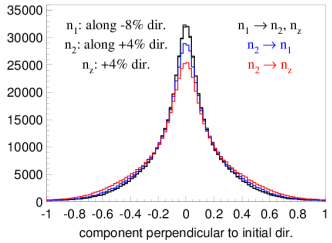
Direct hole ice is now in the OpenCL version, always compiled in

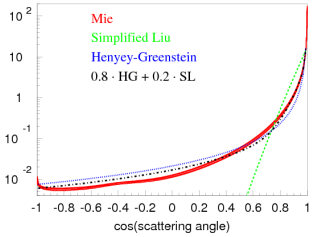
Increased number of wavelength bins passed to GPU to 512 (up from 32)

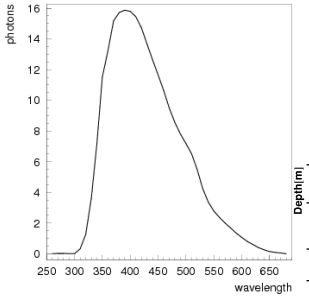
One example script; documentation: in progress

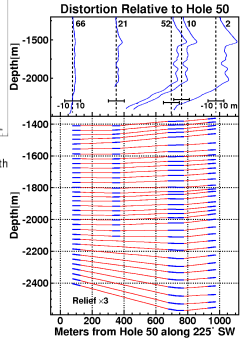
Overview of inputs to PPC











Ice Configuration

Ice is configured in a directory, its location passed to the module via a parameter PPCTABLESDIR (set with os.putenv).

Ice models are now kept in ice-models module. The most recent model is maintained in ppc (“dima” branch).

SPICE 3.2:

as.dat	angular sensitivity curve: flasher-based with p=0.3
cfg.txt	main configuration file
icemodel.dat	ice model table (scattering, absorption vs. depth)
icemodel.par	wavelength dependence coefficients
Icemode.bbl	air bubble parametrization above 1350 m depth
rnd.txt	random number multipliers
tilt.dat	table of ice layer relief values at tabulated points
tilt.par	locations of tabulated points
wv.dat	normalized cherenkov+transmission convolution
cx.dat	table of DOM tilts
dx.dat	table of cable positions

Ice configuration

If running ppc outside icetray (as a stand-alone command-line program or in concert with DirectFit (distributed with ppc in private/ppc/llh directory) the following files are required:

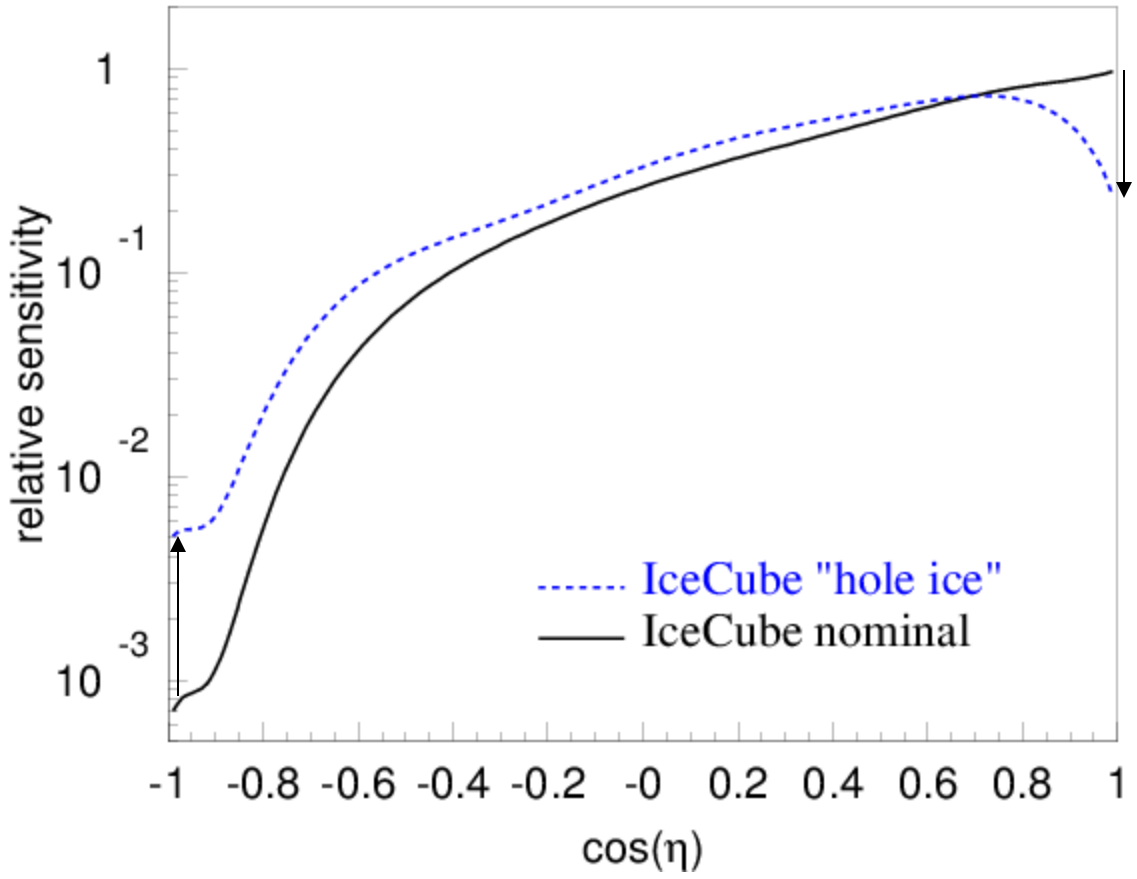
geo-f2k	detector geometry

Optional files:

eff-f2k	table of RDE values
wv.rde	tabulated difference of high-QE/nominal response vs. wavelength
hvs-f2k	table of high voltages (zero voltage disables DOM)

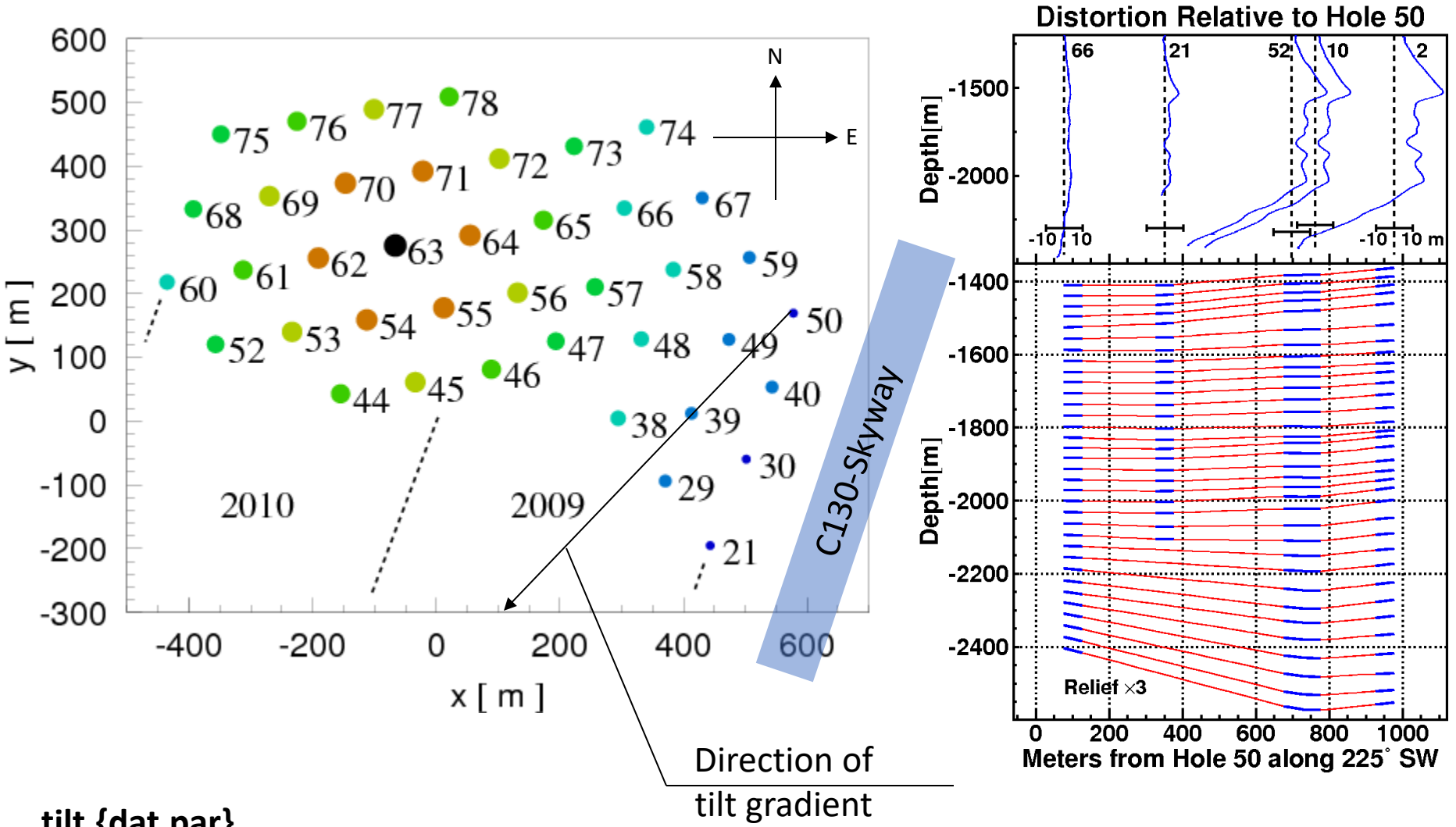
There are also multiple additional input files to llh/DirectFit, these will be described in documentation

Traditional “hole ice” angular sensitivity



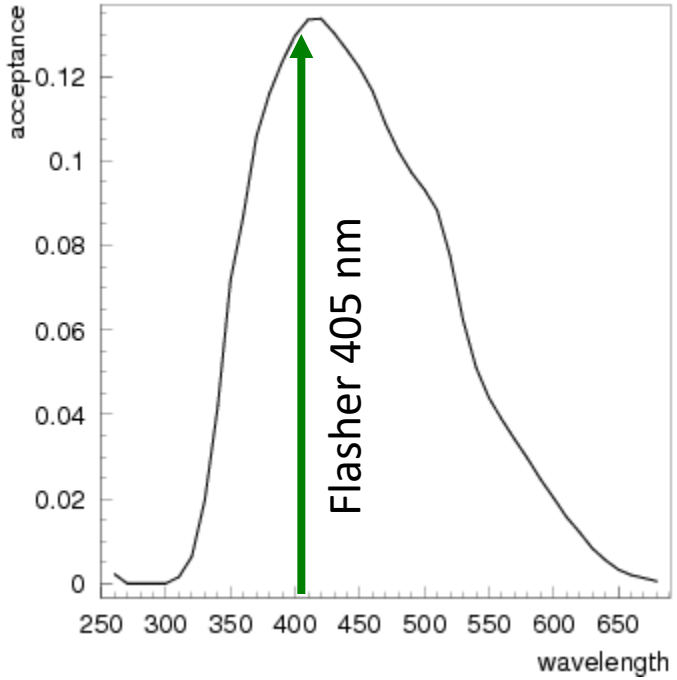
as.dat

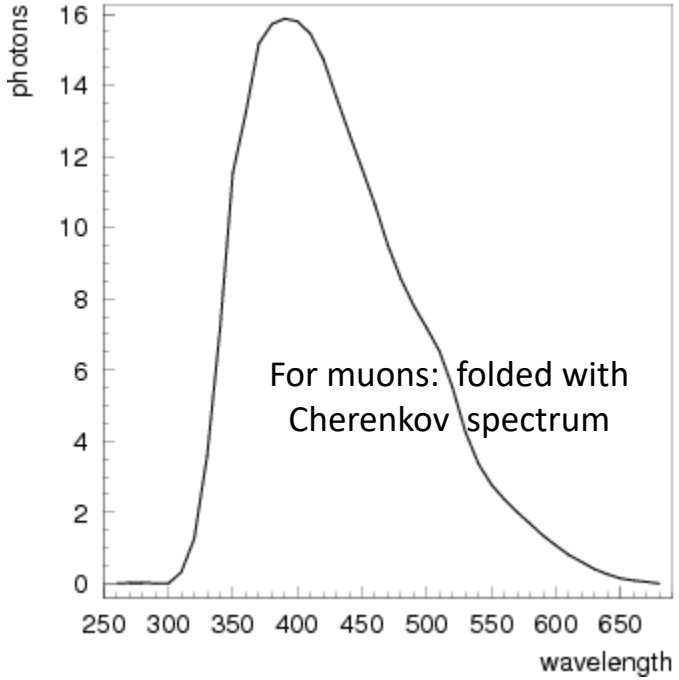
Ice Tilt



tilt.{dat,par}

Photon wavelength sampling



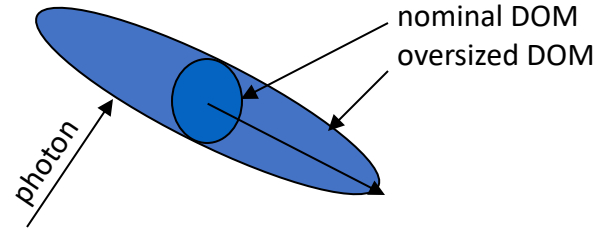


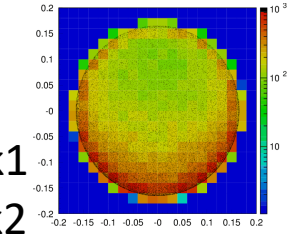
wv.dat

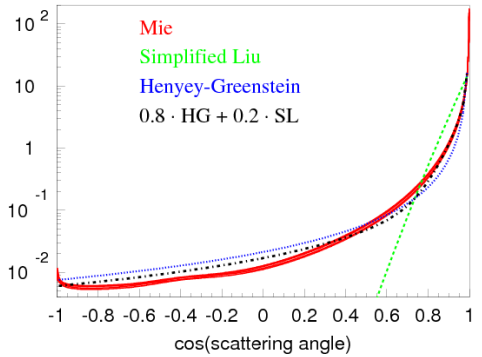
Main configuration file: cfg.txt

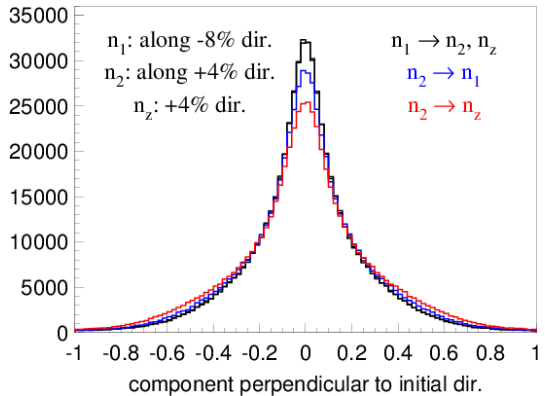
cfg.txt:

```
# ppc configuration file: follow strict order below
5       # over-R: DOM radius "oversize" scaling factor
1.0    # overall DOM efficiency correction
0.35  # 0=HG; 1=SAM
0.9    # g=<cos(theta)>

130   # direction of ice tilt (perp. to flow)
-0.106 # magnitude of major anisotropy coefficient k1
0.053  # magnitude of minor anisotropy coefficient k2

0.5    # hole ice radius in units of [DOM radius]
0.5    # hole ice effective scattering length [m]
100   # hole ice absorption length [m]
0.35  # hole ice 0=HG; 1=SAM
0.9    # hole ice g=<cos(theta)>
```

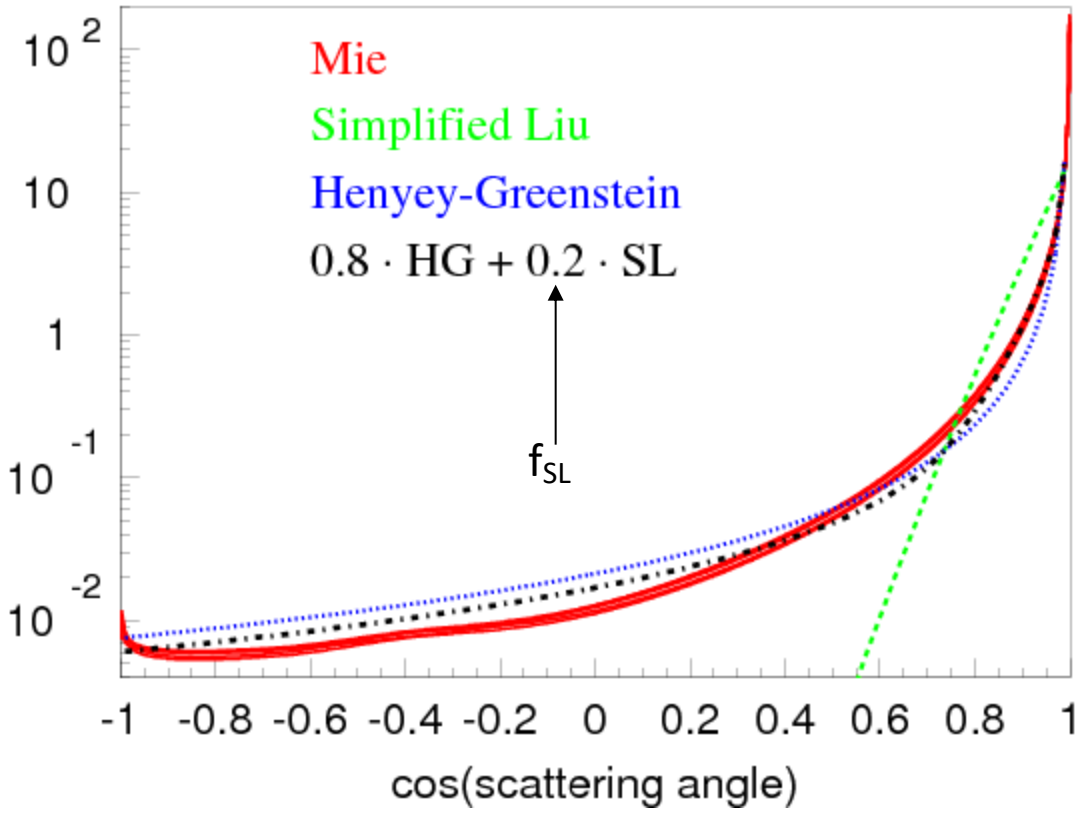








Approximation to Mie scattering



HG/SAM mixing fraction

Simplified Liu:

$$p(\cos\theta) \sim (1+\cos\theta)^{\alpha}, \quad \text{with} \quad \alpha = \frac{2g}{1-g}$$

Henyey-Greenstein:

$$p(\cos\theta) = \frac{1}{2}\frac{1-g^2}{[1+g^2-2g\cdot\cos\theta]^{3/2}}$$

Mie:

Describes scattering on acid, mineral, salt, and soot with concentrations and radii at SP

Anisotropy parameterization

The scattering function we use is f(cos θ), a combination of HG and SL.

One obviously covariant extension that satisfies the symmetry condition of the previous slide is

$$f(\vec{n_i} \cdot \vec{n_o}) \rightarrow f(\vec{k_i} \cdot \vec{k_o}), \quad \vec{k}_{i,o} = \frac{A\vec{n}_{i,o}}{|A\vec{n}_{i,o}|}$$

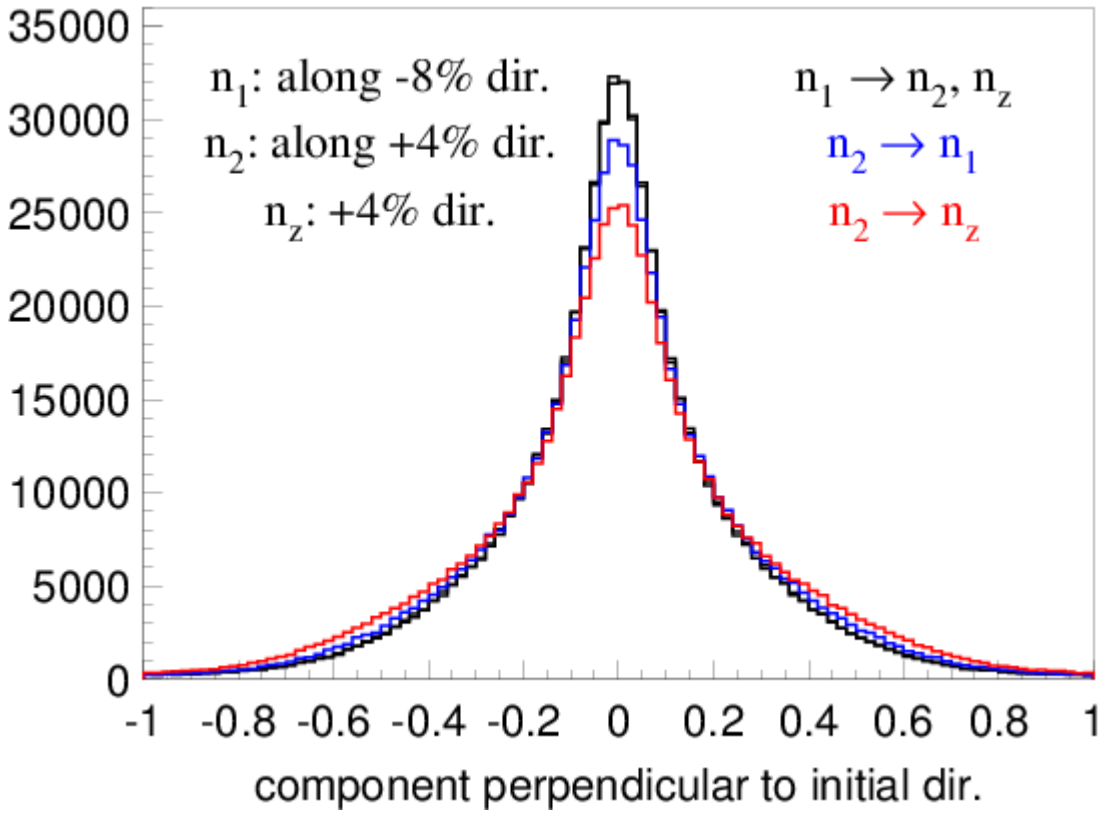
In the basis of the 2 main scattering axes and z (presumably the third one)

$$A = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix}$$

We can impose $\alpha\beta\gamma$=1, thus this parametrization introduces two extra parameters: α,β (in addition to the direction of scattering preference).

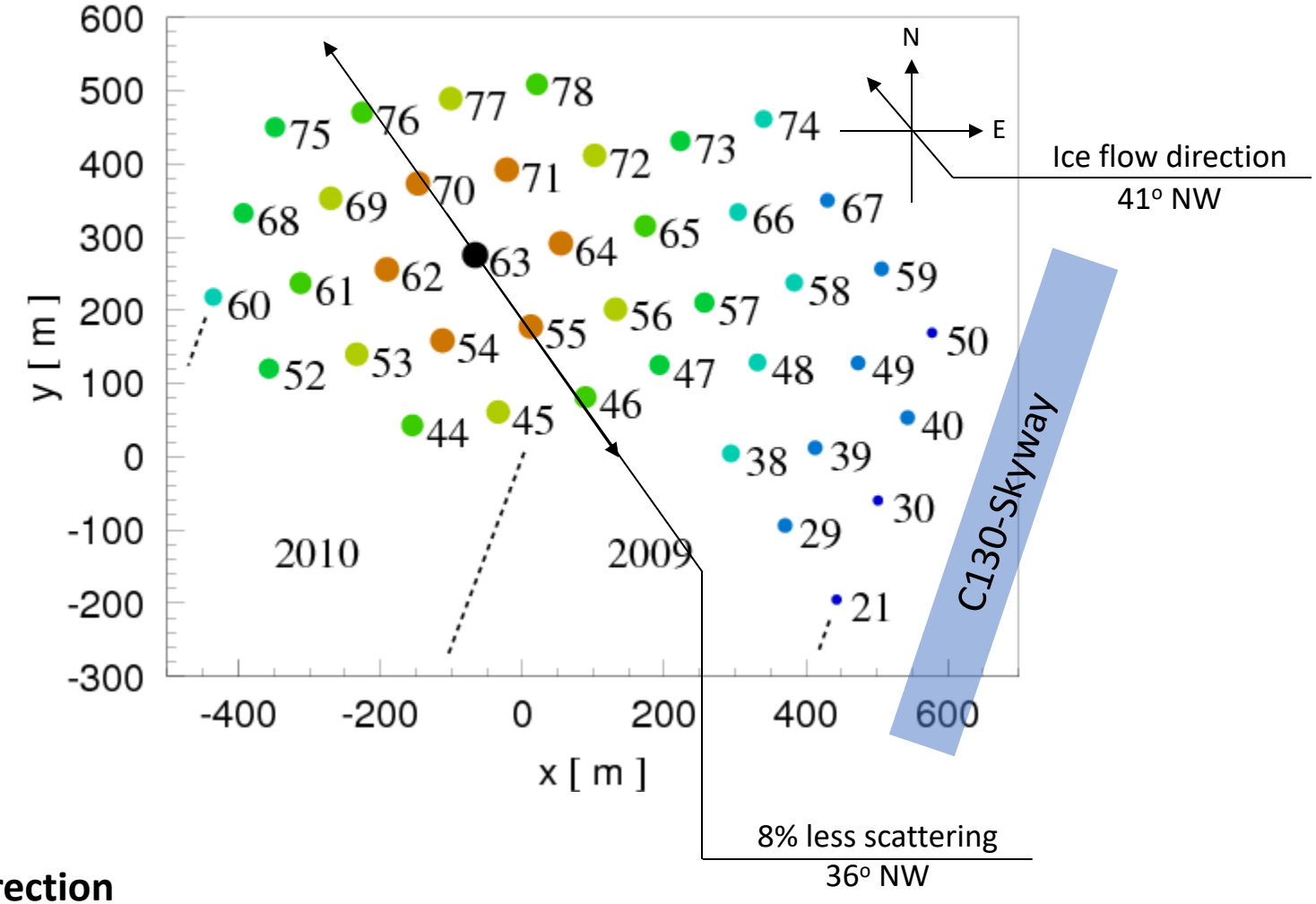
The geometric scattering coefficient is constant with azimuth. However, the effective scattering coefficient receives some azimuthal dependence.

Scattering anisotropy



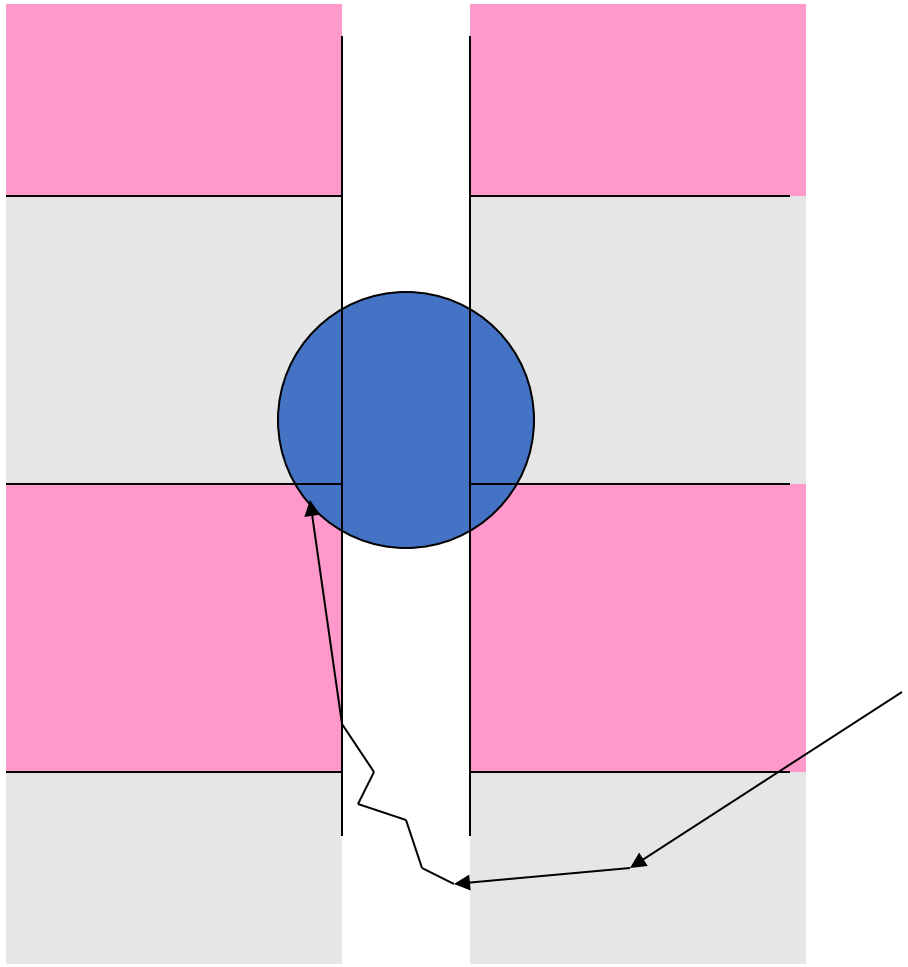
Anisotropy coefficients

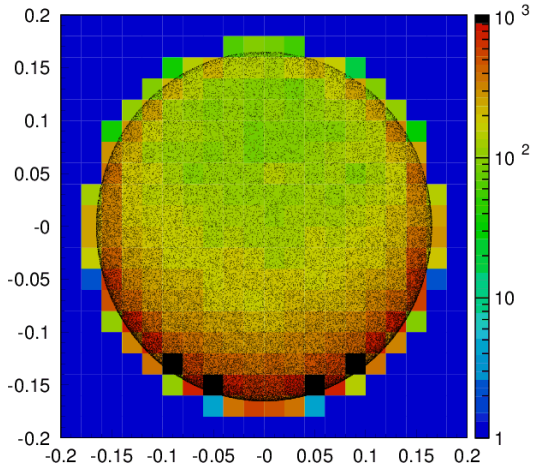
Direction of anisotropy



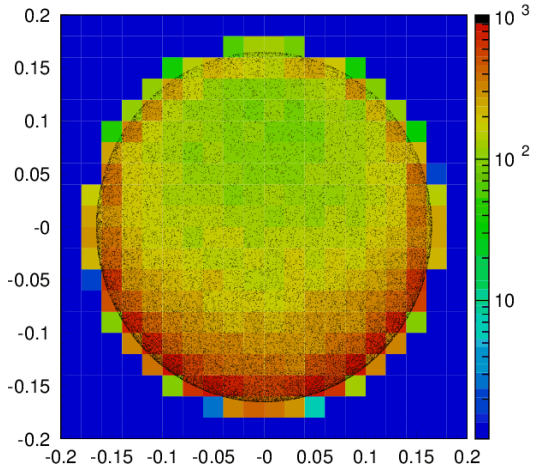
Anisotropy direction

Direct Hole Ice simulation





nominal



hole ice

Flasher/SC configuration

Can simulate:

- horizontal or tilted flashers
- individual, several, or cylindrically-symmetric
- 2d gaussian (von Mises-Fisher) with 9.7 degree spread
- rectangular emission profile (in time)
- single wavelength or specify in wv.dat

- both standard candles, locations and direction hard-coded

Light yield parametrization

Calculated 2450.08 of Cherenkov photons per meter of bare muon track when convolved with transmission properties of glass, gel, and detection efficiency of PMT.

Additional factor: efficiency, or “cable shadowing” is applied to this number.

Via compile-time configuration choose:
Light yield:

- ~~C.W. parametrization of bare muon/em/hadr. cascades~~
- ~~M.K. updated parametrization for em/hadr. cascades~~
- Leif Raedel updated parametrization of muon/em/hadr. cascades

Longitudinal profile and Cherenkov cone:

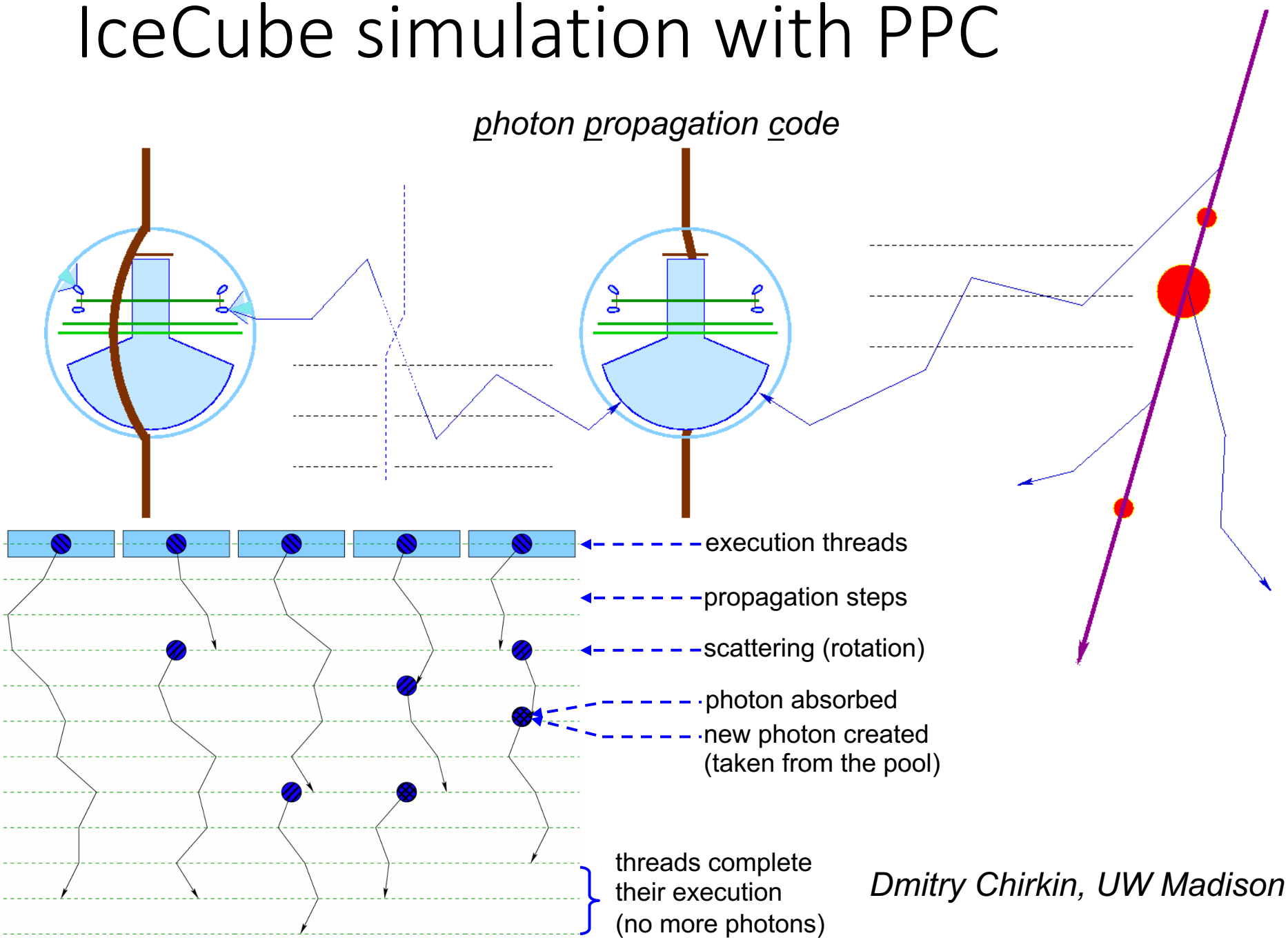
- ~~C.W. parametrization (simplified fit)~~
- Leif Raedel updated parametrization

PPC

Update 2014

IceCube simulation with PPC

photon propagation code



Dmitry Chirkin, UW Madison

Direct photon tracking with PPC

photon propagation code

- simulates flasher/standard candle photons
- same code for muon/cascade simulation

- uses tabulated (in 10 m depth slices) layered ice structure
- employs 6-parameter ice model to extrapolate in wavelength
- tilt in the ice layer structure can be taken into account

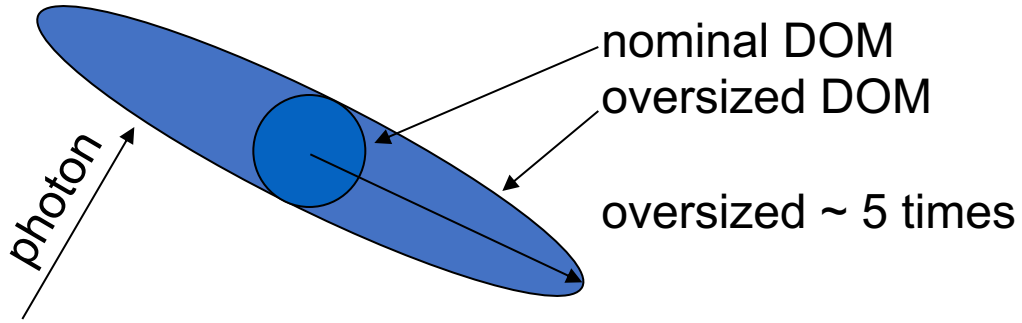
- uses improved scattering function: linear combination of HG+SAM

- precise simulation of the longitudinal development of cascades and
- angular distribution of particles emitting Cherenkov photons

Oversized DOM treatment

This is a crucial optimization.

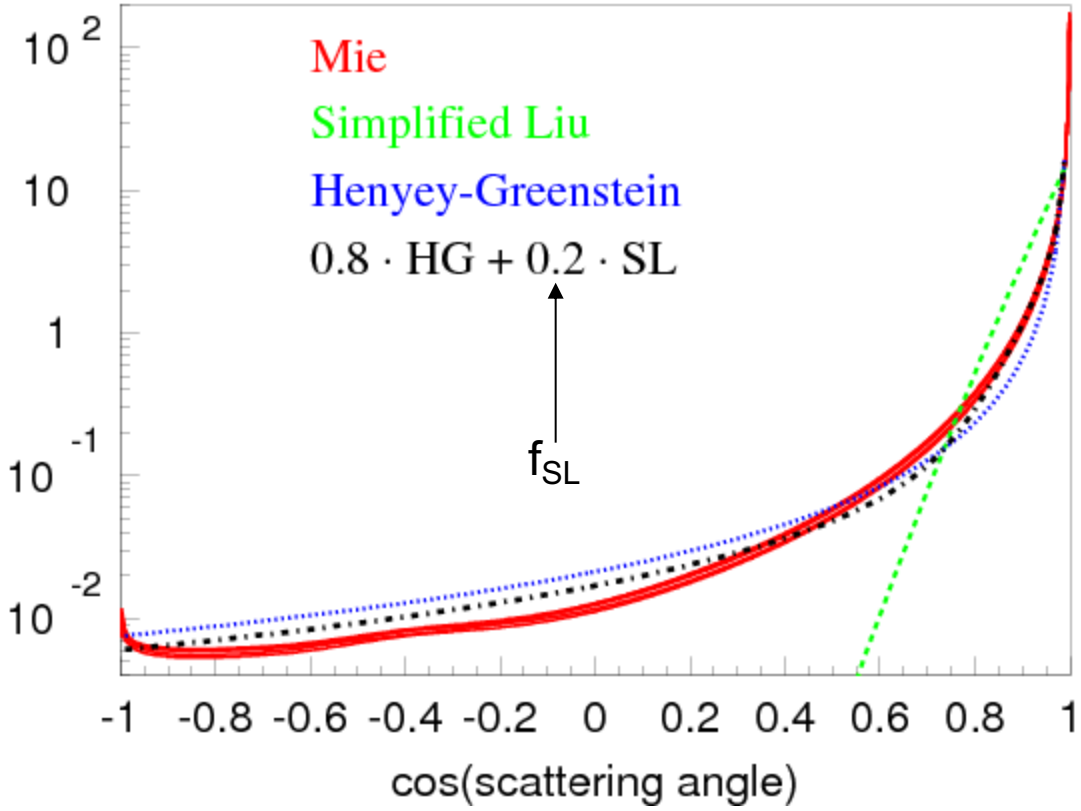
The oversize model was chosen carefully to produce the best possible agreement with the nominal x1 case.



Some bias is unavoidable since DOMs occupy larger space:

- x1: diameter of 33 cm
- x5: 1.65 m
- x16: 5.3 m

Approximation to Mie scattering



Simplified Liu:

$$p(\cos\theta) \sim (1+\cos\theta)^{\alpha}, \quad \text{with} \quad \alpha = \frac{2g}{1-g}$$

Henyey-Greenstein:

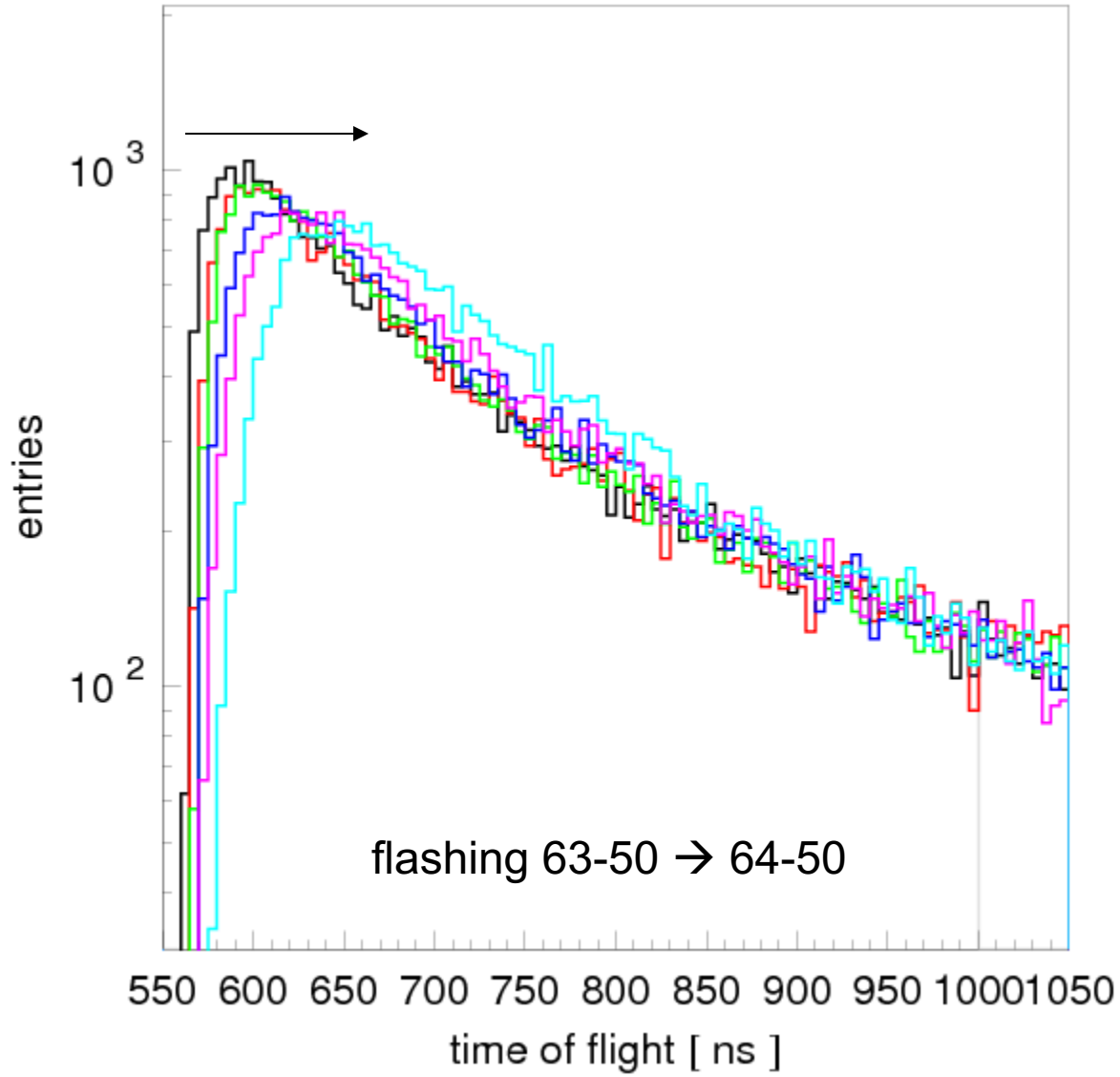
$$p(\cos\theta) = \frac{1}{2}\frac{1-g^2}{[1+g^2-2g\cdot\cos\theta]^{3/2}}$$

Mie:

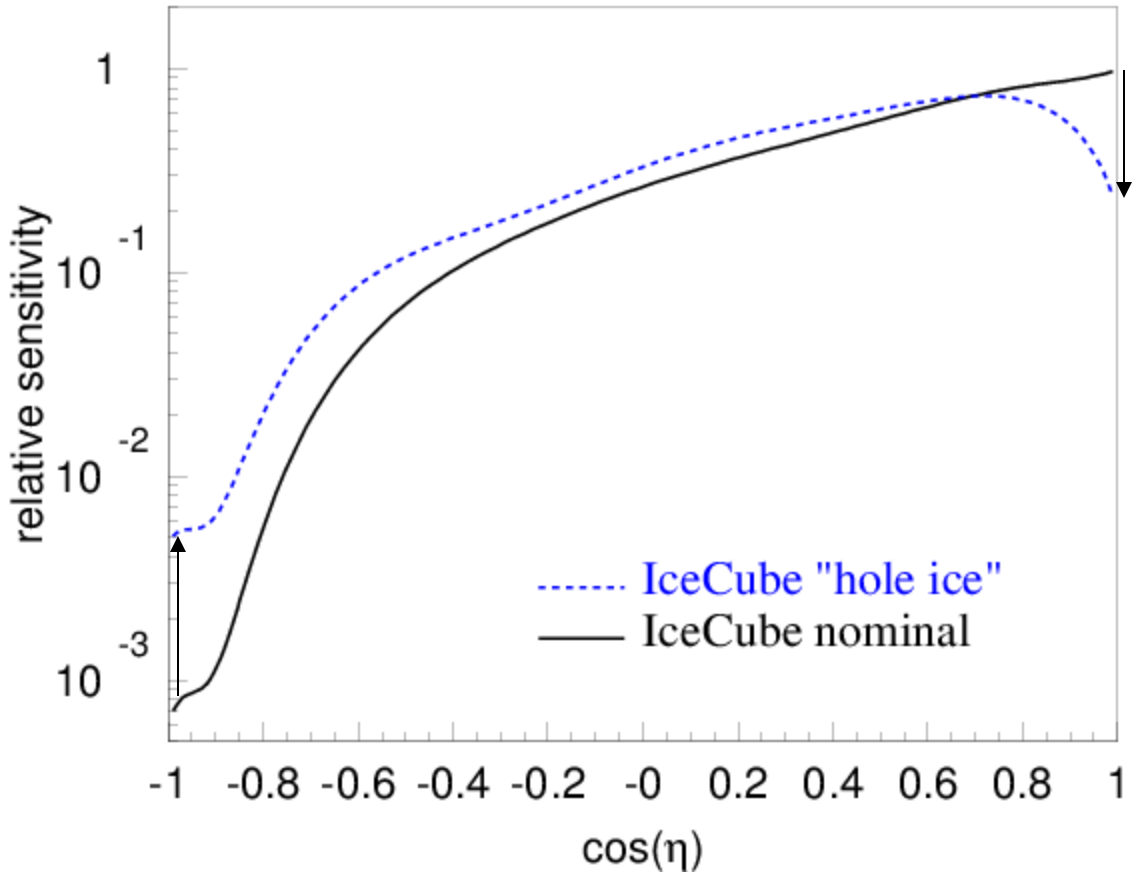
Describes scattering on acid, mineral, salt, and soot with concentrations and radii at SP

Dependence on g=<cos(θ)> and f_{SL}

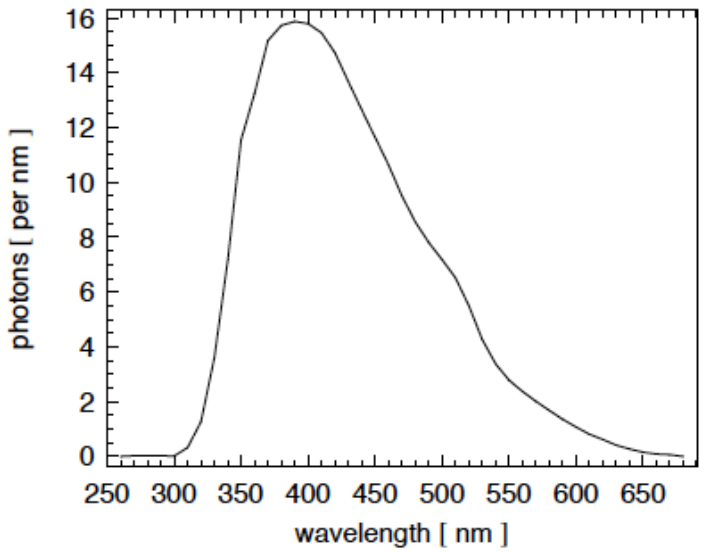
g=<cos(θ)>	f_{SL}
0.8	0
0.9	0
0.95	0
0.9	0.3
0.9	0.5
0.9	1.0



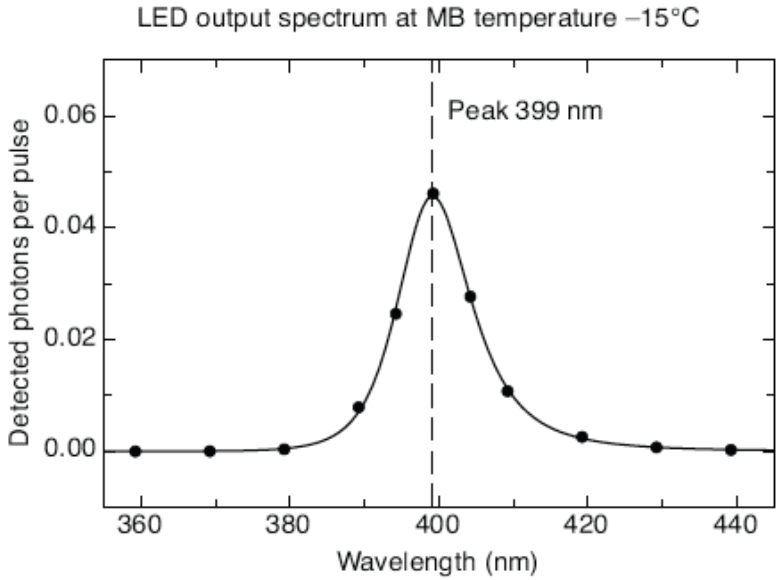
“hole ice” angular sensitivity



Wavelength distributions



Cherenkov emission



405 nm flashers

ppc icetray module

• at http://code.icecube.wisc.edu/svn/projects/ppc/trunk/

• uses a wrapper: private/ppc/i3ppc.cxx, which compiles by cmake into the libppc.so.

• an additional library libxppc.so is compiled by default for the OpenCL version or by running make in private/ppc/gpu:

→ “make glib” compiles gpu-accelerated version (needs cuda tools)
→ “make clib” compiles cpu version (from the same sources)

→ “make olib” in private/ppc/ocl to compile the OpenCL version by hand

• link to libxppc.so and libcudart.so (if CUDA version) from build/lib directory

• this library file must be loaded before the libppc.so wrapper library

Ice configuration

as.dat	DOM angular sensitivity parametrization
cfg.txt	configuration file
icemodel.dat	ice table: depth, $b_e(400)$, $a_{dust}(400)$, δT
icemodel.par	wavelength dep. parameters: α, κ, A, B
rnd.txt	random number multipliers
tilt.dat	ice tilt table (depth corrections)
tilt.par	positions of tabulated points
wv.dat	source wavelength dependence
wv.rde	correction to high-QE DOM wavelength dep.

$$b_e(\lambda) = b_e(400) \cdot \left(\frac{\lambda}{400}\right)^{-\alpha}$$

$$a(\lambda) = a_{\text{dust}}(\lambda) + Ae^{-B/\lambda} \cdot (1 + 0.01 \cdot \delta\tau), \quad \textbf{with} \quad a_{\text{dust}}(\lambda) = a_{\text{dust}}(400) \cdot \left(\frac{\lambda}{400}\right)^{-\kappa}$$

cfg.txt

```
# ppc configuration file: follow strict order below
5      # over-R: DOM radius "oversize" scaling factor
1.0    # overall DOM efficiency correction
0.4    # 0=HG; 1=SAM
0.9    # g=<cos(theta)>

225     # direction of ice tilt (perp. to flow)
0.047   # magnitude of ice anisotropy along tilt
-0.075 # magnitude of ice anisotropy along flow

0.0    # hole ice radius in units of [DOM radius]
0.5    # hole ice effective scattering length [m]
100   # hole ice absorption length [m]
0.45 # hole ice 0=HG; 1=SAM
0.9    # hole ice g=<cos(theta)>
```

ppc example script run.py

```
if(len(sys.argv)!=6):
    print "Use: run.py [corsika/nugen/flasher] [gpu] [seed] [infile/num of flasher events] [outfile]"
    sys.exit()
…
det = "ic86"
detector = False
…
os.putenv("PPCTABLESDIR", expandvars("$I3_BUILD/ppc/resources/ice/mie"))
…
if(mode == "flasher"):
    …
    str=63
    dom=20
    nph=8.e9

    tray.AddModule("I3PhotoFlash", "photoflash")(…)

    os.putenv("WFLA", "405")  # flasher wavelength; set to 337 for standard candles
    os.putenv("FLDR", "-1")   # direction of the first flasher LED
    …
    # Set FLDR=x+(n-1)*360, where 0<=x<360 and n>0 to simulate n LEDs in a
    # symmetrical n-fold pattern, with first LED centered in the direction x.
    # Negative or unset FLDR simulates a symmetric in azimuth pattern of light.

    tray.AddModule("i3ppc", "ppc")(
        ("gpu", gpu),
        ("bad", bad),
        ("nph", nph*0.1315/25),   # corrected for efficiency and DOM oversize factor; eff(337)=0.0354
        ("fla", OMKey(str, dom)), # set str=-str for tilted flashers, str=0 and dom=1,2 for SC1 and 2
        )

else:
```

ppc example script run.py (cont.)

```
os.putenv("PPCTABLESDIR",
expandvars("$I3_BUILD/ppc/resources/ice/lea"))


    load("libxppc")
    load("libppc")

    tray.AddModule("i3ppc", "ppc")(
        ("gpu", gpu),
    )
```

PPC module parameters

```
  AddParameter ("gpu", "GPU to use", gpu);
  AddParameter ("bad", "DEPRECATED : DOMs not to use", bad);
  AddParameter ("fla", "Flasher position", fla);
  AddParameter ("nph", "Number of photons", nph);
  AddParameter ("wid", "Flasher pulse width", wid);
  AddParameter ("MCTree", "MCTree to use", mct);
  AddParameter ("JPALpulses", "Simulate Jitter and
Pre,After,Late pulses", false);
  AddParameter ("cyl", "use cylinder (1) or strict +300 m
(0) detector volume", cyl);
```

Environment settings

PPCTABLESDIR	configuration directory
WFLA	sets single wavelength (disables wv.dat)
FLDR	configuration of flasher LEDs
FWID	flasher LED beam width (9.7 default)
NPHO/NPHO_0	number of photons per 1 thread
CUDA:	
BADMP/BADMP_0	disables given MP (multiprocessor)
OpenCL:	
XMLT/XMLT_0	simultaneously load multiple job grids
OCPU/OGPU/OACC	use only CPU/GPU/accelerator cards

ppc-pick and ppc-eff

ppc-pick: restrict to primaries below *MaxEpri*

```
load("libppc-pick")

tray.AddModule("I3IcePickModule<I3EpriFilt>","emax")(
        ("DiscardEvents", True),
        ("MaxEpri", 1.e9*I3Units.GeV)
        )
```

ppc-eff: reduce efficiency from 1.0 to *eff*

```
load("libppc-eff")

tray.AddModule("AdjEff", "eff")(
    ("eff", eff)
    )
```

Additional resources

README files:

resources/README

http://icecube.wisc.edu/~dima/work/WISC/ppc/readme.html

SPICE paper:

http://arxiv.org/abs/1301.5361

Anisotropy paper:

http://arxiv.org/pdf/1309.7010.pdf

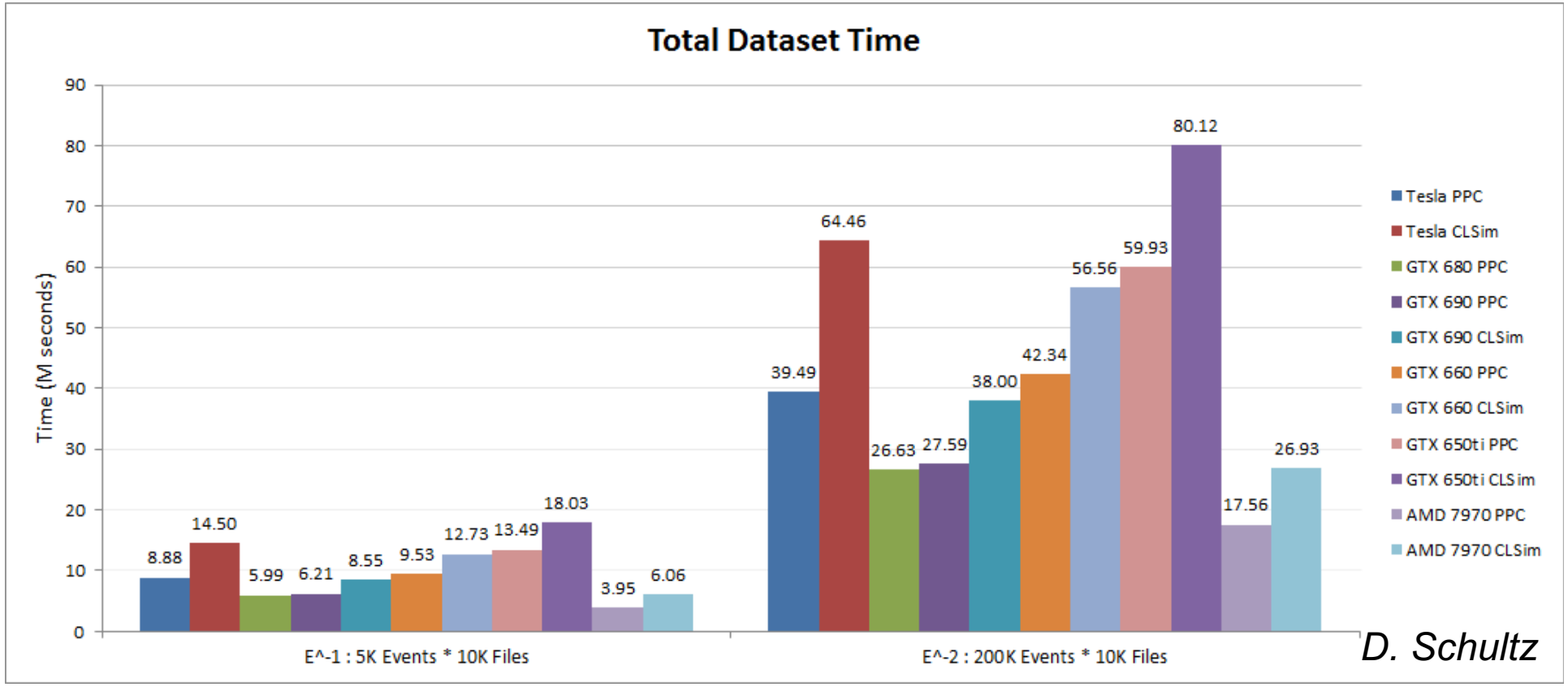
Code forks:

http://code.icecube.wisc.edu/svn/sandbox/aobertacke/ppc/

http://code.icecube.wisc.edu/svn/sandbox/schatto/ppc-tables/

Run Time Benchmarks

From https://wiki.icecube.wisc.edu/index.php/GPU_Benchmarks



Ratio run time ppc/clsim = 0.62