

The I3RC Community Monte Carlo Code: Comparisons with SHDOM

Robert Pincus
CIRES-NOAA Climate Diagnostic Center

K. Franklin Evans
University of Colorado

Why an I3RC Community Monte Carlo Code?

- For novices – so 3D RT is easier to do using the correct tool.
 - Stimulate interest interest in 3D RT
- For experts – one way to share algorithm ideas

Agreed Upon Goals for I3RC MC Project

- Easy to use “out of the box”
 - Safe – hard to make clumsy mistakes
 - Flexible
 - Fast enough to be genuinely useful
 - A good example, algorithmically and in coding practice
 - Easily parallelizable
 - Portable
- Some of these goals can be contradictory

The Design: Software Structure Reflects Conceptual Structure

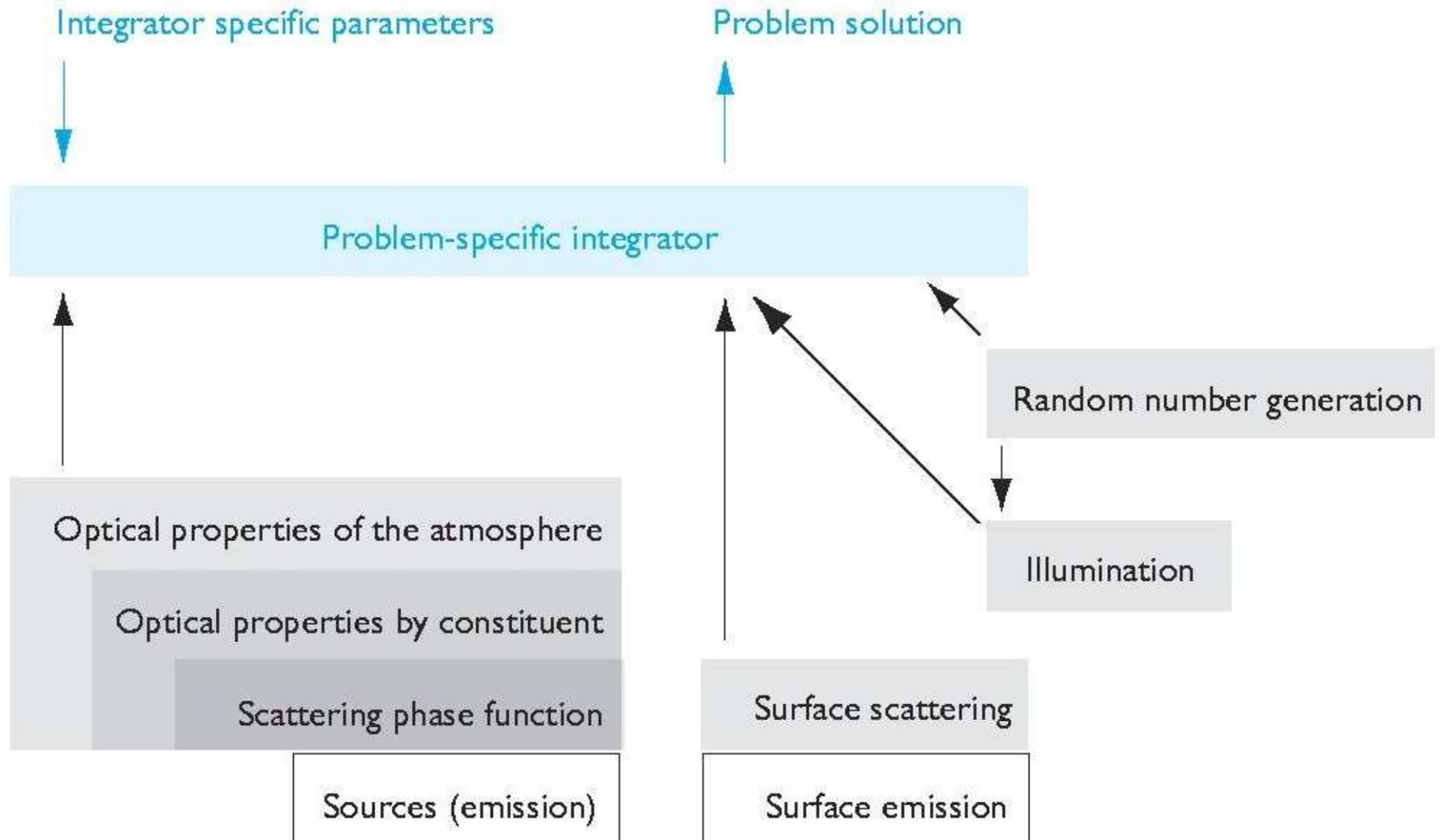
“Its not just about the algorithm”

- The I3RC Community MC code is a software framework, in which each module represents (“encapsulates”) a different conceptual piece of the problem:
 - Single scattering properties of each component in the medium
 - The 3D distribution of optical properties
 - Surface reflection characteristics
 - Illumination
 - Radiative transfer algorithm

Software as Building Blocks

- The I3RC code is a “framework” - a set of interconnecting modules that is the interface to the user.
 - Each module represents an independent piece of the problem.
 - Each module is defined by data structures and procedures (functions and subroutines) that act on the data.
- We are providing a “reference implementation”
 - An implementation you write should have the same interface.
- The framework modules provide the pieces for doing monochromatic radiative transfer
- We are providing “integrators” for flux and radiance.

Loops (spectral, variance-estimating; may be parallel)



On Style

- The full specifications for a 3D RT calculation are complex
 - RT codes typically have one of two interfaces:
 - 1) Arguments in a subroutine call (DISORT)
 - 2) Files to be read by a program (SHDOM)
- These interfaces try to hide the complexity, but do they really...

More On Style

- Doing problems with the I3RC code takes a different approach
 - Calculations require a relatively large number of calls
 - Properties are specified in memory, not in files
 - Like using a library (e.g. netCDF)
- The subroutine calls make the complexity visible.
- The complexity results in much flexibility - the building blocks (subroutine calls) can be put together in different ways.
- Separates pieces of the problem – developers can focus on solving a single piece.
- The I3RC code will work in a wide range of environments.

Current Capabilities

- The framework is for monochromatic RT
- There may be multiple components and phase function tables.
- The example integrator has maximal cross-section and photon-tracing subroutines. It computes
 - Pixel level and domain average fluxes at the boundaries, and column absorbed fluxes and domain average absorption profiles
 - Radiances at boundaries using (slow) local estimation
- To help people get started:
 - Utilities to convert SHDOM files to “internal” netcdf files
 - Simple examples (easily modified for your problem)
 - I3RC phase 1 test cases

```
program i3rc_mc
! I3RC based Monte Carlo solar radiative transfer program that computes
! domain top and bottom pixel level outgoing fluxes and radiances
! for an input SHDOM optical properties file.

! Modules from the I3RC community Monte Carlo model framework
use ErrorMessages
use testingUtils
use RandomNumbers
use scatteringPhaseFunctions
use opticalProperties
use monteCarloIllumination
use monteCarloRadiativeTransfer

! Declare local variables and then the I3RC defined types:
type(phaseFunction), allocatable :: ppVector(:)
type(phaseFunctionTable) :: phaseFuncTable
type(domain) :: thisDomain
type(ErrorMessage) :: status
type(randomNumberSequence) :: randoms
type(photonStream) :: incomingPhotons
type(integrator) :: mcIntegrator
```

```

! Get the input variables with Fortran READs:
!   property file name, solar direction, surface albedo,
!   number of photons, radiance directions, output files.

! Read the SHDOM-style property file
!   including the tabulated phase functions and the extinction, single
!   scattering albedo, and phase function index for each grid point
! Average the optical property level values vertically to get layer means

! Create the I3RC MC "domain" object
thisDomain = new_Domain (deltaX * (/ (i, i = 0, nX) /), &
                        deltaY * (/ (i, i = 0, nY) /), &
                        zPosition, status)

call printStatus(status)

! Give the tabulated phase functions to the scatteringPhaseFunctions module
do i = 1, nPhaseEntries
  ppVector(i) = new_PhaseFunction( &
                                legendreCoefficients(1:n,i) / (/ (2*l+1, l = 1, n) /), &
                                status=status)

  call printStatus(status)
end do

```

```

! Create the phase function table
phaseFuncTable = new_PhaseFunctionTable (ppVector(1:nPhaseEntries), &
                                         key = real( (/ (i, i = 1, nPhaseEntries) /) ), &
                                         status = status)

call printStatus(status)

! Add the optical properties to the domain
call addOpticalComponent (thisDomain, "mixture", &
                          extinctLay(:, :, :), ssaLay(:, :, :), &
                          phaseFuncIndexLay(:, :, :), &
                          phaseFuncTable, status = status)

call printStatus(status)
call finalize_PhaseFunctionTable (phaseFuncTable)

! Set up the integrator object - the integrator makes copies of the
! 3D distribution of optical properties, so we can release the memory
mcIntegrator = new_Integrator (thisDomain, status = status)
call printStatus(status)
call finalize_Domain(thisDomain)

```

```

! Set the surface albedo, table sizes, and maybe the radiance directions
call specifyParameters (mcIntegrator, surfaceAlbedo=sfcalbedo, &
                        minInverseTableSize=nPhaseIntervals, &
                        status=status)

call printStatus(status)
if (intensityFlag) then
  call specifyParameters (mcIntegrator, &
                          minForwardTableSize=nPhaseIntervals, &
                          intensityMus=MuRadiance, intensityPhis=PhiRadiance, &
                          computeIntensity=intensityFlag, status=status)

  call printStatus(status)
endif

! Seed the random number generator.
!   Variable randoms holds the state of the random number generator.
randoms = new_RandomNumberSequence(seed = iseed)

! The following loop is for estimating the uncertainty in the flux and
!   radiance from the variance between numBatches independent calculations.
do batch = 1, numBatches

```

```

! The initial direction and position of the photons are precomputed and
!   stored in an "illumination" object.
incomingPhotons = new_PhotonStream (solarMu, solarAzimuth, &
                                     numberOfPhotons=numPhotonsPerBatch, &
                                     randomNumbers=randoms, status=status)
call printStatus(status)

if (isReady_Integrator (mcIntegrator)) then
    ! Now we compute the radiative transfer for this batch of photons.
    call computeRadiativeTransfer (mcIntegrator, randoms, &
                                    incomingPhotons, status)
    call printStatus(status)
endif

! This particular integrator provides fluxes at the top and bottom
!   of the domain for both the domain mean and pixel level fluxes,
!   and the pixel level radiances at top and/or bottom of domain.
call reportResults (mcIntegrator, &
                    meanFluxUp(batch), meanFluxDown(batch), meanFluxAbsorbed(batch), &
                    fluxUp(:, :, batch), fluxDown(:, :, batch), fluxAbsorbed(:, :, batch), &
                    absorbedProfile(:, batch), status=status)

```

```
if (intensityFlag) then
  call reportResults (mcIntegrator, intensity=Radiance(:, :, :, batch), &
                    status=status)
endif
call printStatus(status)

  ! Release the photon "illumination" object memory (important!)
  call finalize_PhotonStream (incomingPhotons)
end do ! end of batch loop

call finalize_Integrator (mcIntegrator)
call finalize_RandomNumberSequence(randoms)

! Calculate the mean and standard deviation of the mean over the batches

! Output the flux results to a file
! Output the absorption profile results to a file
! Output the radiance results to a file
end program i3rc_mc
```

What is in the future?

- “Shakedown cruise” has discovered bugs, inefficiencies, and documentation errors.
- We expect to add (eventually):
 - more efficient local estimation for radiances
 - k-distribution module for integration over molecular spectrum
 - wider range of utilities and example programs
- Next release (“Bramley”) by year's end to include
 - integrator with radiances
 - many bug fixes
 - wider range of examples

Role for Community Involvement

- Advice is welcome – particularly with respect to needs
- Code is certainly welcome
 - Integrators or modifications to the existing ones
 - There's a learning curve, but you can work from examples
- Use is welcome
 - Don't really expect from this audience, but we can dream
 - Robert & Frank can't hold everyone's hand, but we can develop a community of users and experience

Comparison of I3RC Monte Carlo with SHDOM

- What: Compare models in CPU time – accuracy space
- Why: - Tests I3RC MC code
 - Determine tradeoffs in using the models in various applications.
 - Educate users new to 3D RT how to choose correct public code.
- How:
 - 1) Run models for different 3D fields and wavelengths for range of resolutions/number of photons.
 - 2) Compare output for a variety of output radiative quantities.
 - 3) Compare CPU time as a function of accuracy.
- Caveat: What is the truth?

Simulation Parameters

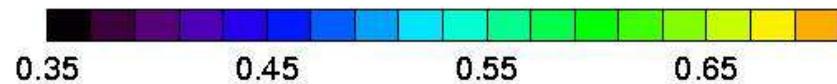
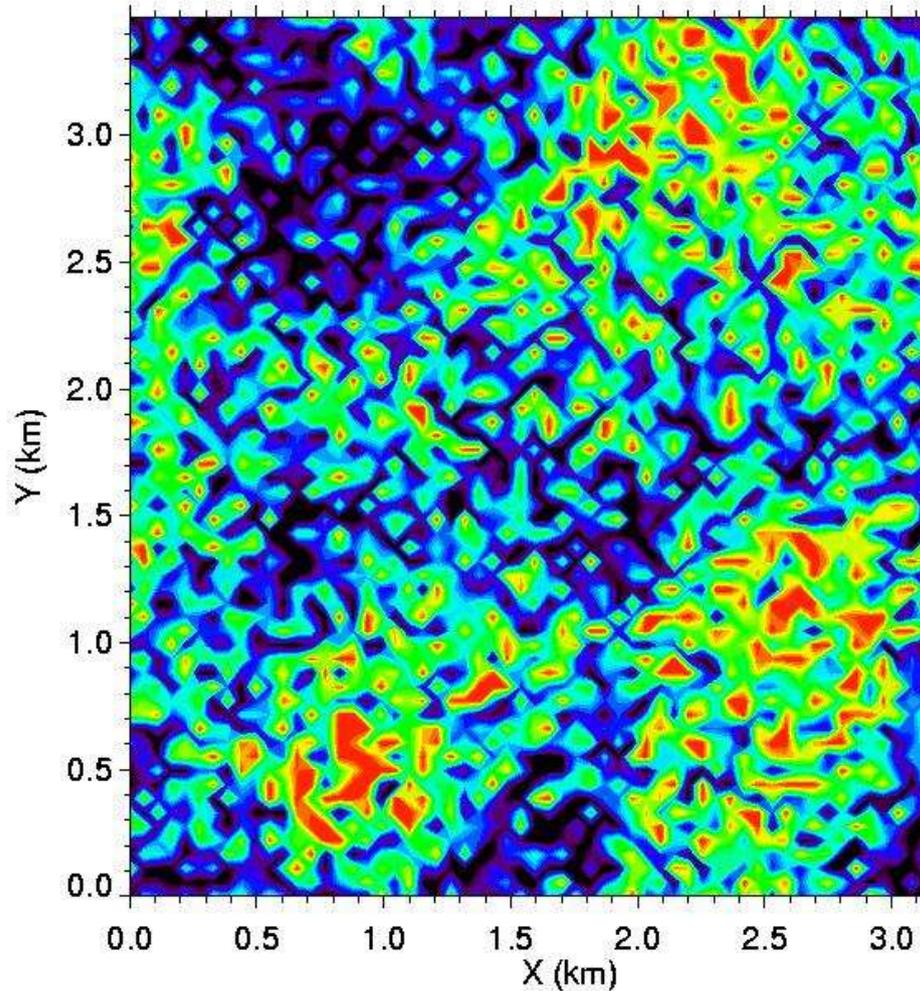
- I3RC Phase II cloud fields:
 - stcu: 64x64x16 stratocumulus field; 55x55x25 m³; 0 to 1 km
 - cu: 96x96x36 cumulus field; 66.7x66.7x40 m³; 0 to 2.5 km
 - Variable r_{eff} ; No aerosols or molecular absorption/scattering.
- Wavelengths: 0.67 and 2.13 μm
- One solar angle: $\mu_0=0.5$
- Lambertian surface reflection: albedo = 0.2
- Output: radiances at $(\mu, \varphi) = (1, 0) (0.5, 0^\circ) (0.5, 90^\circ) (0.5, 180^\circ)$;
pixel fluxes at top and bottom of domain; absorption profile

Model Parameters

- Four SHDOM resolutions:
 - 1) $N_{\mu}=2$ $N_{\phi}=4$ splitacc=0.10
 - 2) $N_{\mu}=4$ $N_{\phi}=8$ splitacc=0.10
 - 3) $N_{\mu}=8$ $N_{\phi}=16$ splitacc=0.05
 - 4) $N_{\mu}=12$ $N_{\phi}=24$ splitacc=0.03
- Monte Carlo: 10^5 , 10^6 , 10^7 , 10^8 , 10^9 photons
- Computers: 3.2 GHz Pentium Xeon & Mac G5

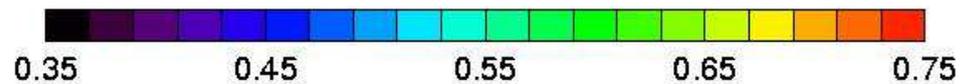
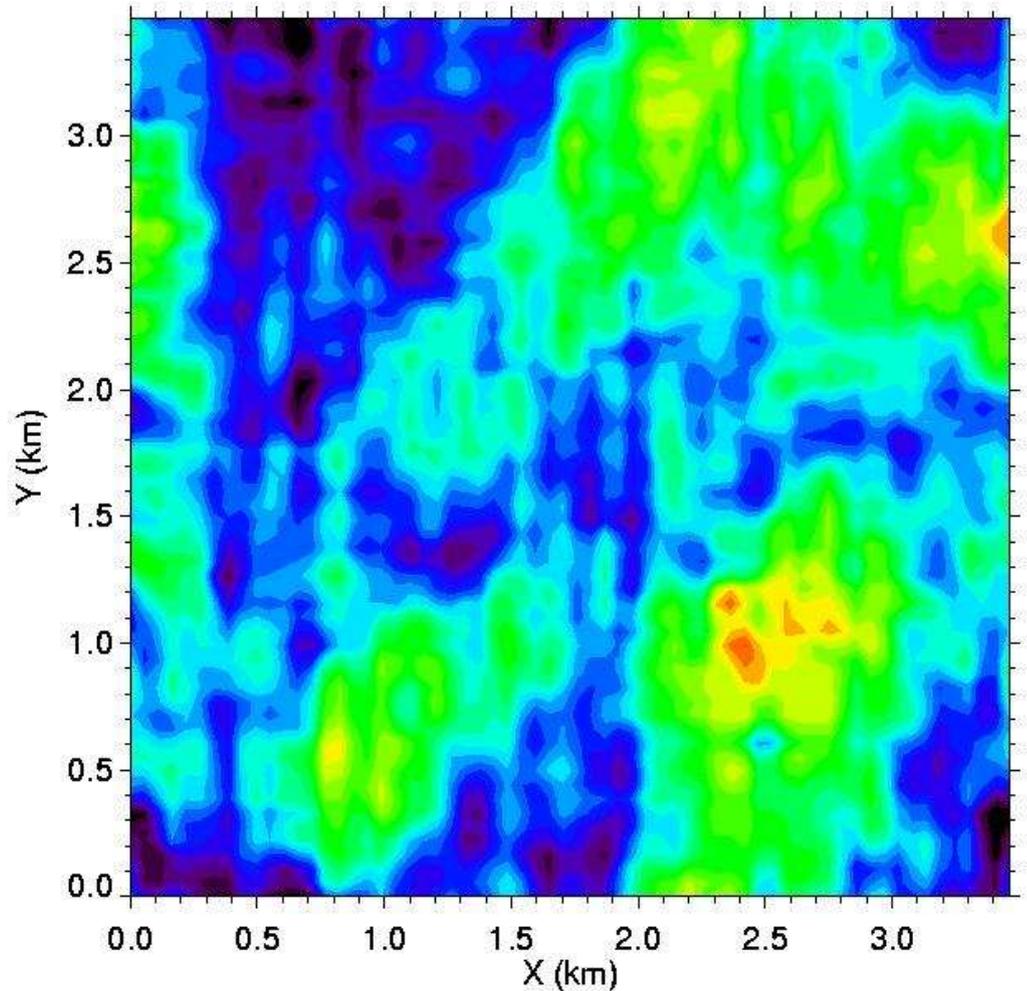
Upwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^5$



Upwelling Flux at Z=1.00 km

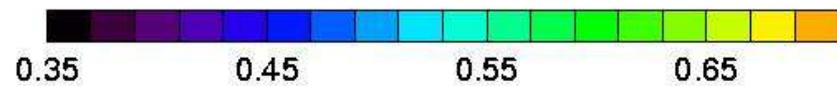
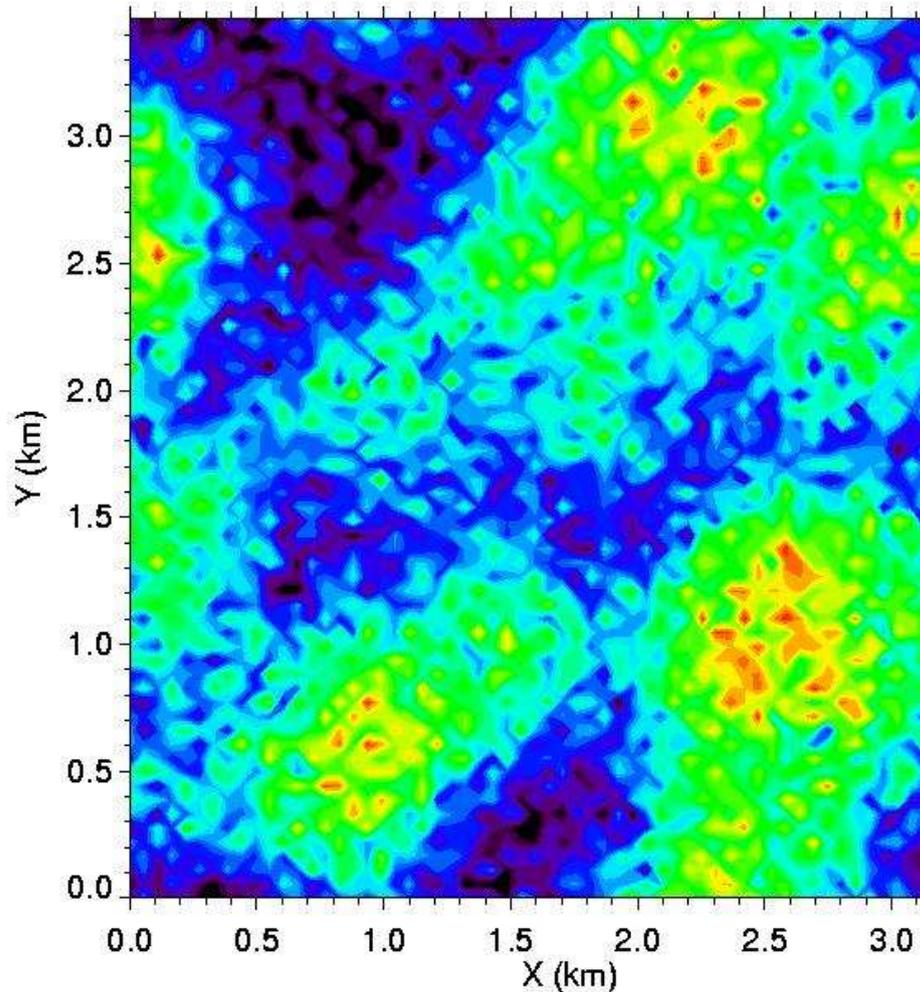
SHDOM: $N_{\mu}=2$ splitacc=0.10



Upwelling Flux at Z=1.00 km

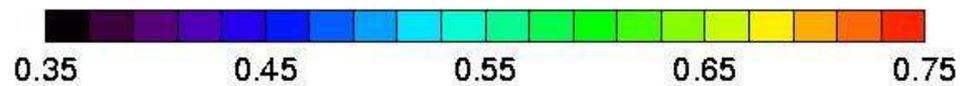
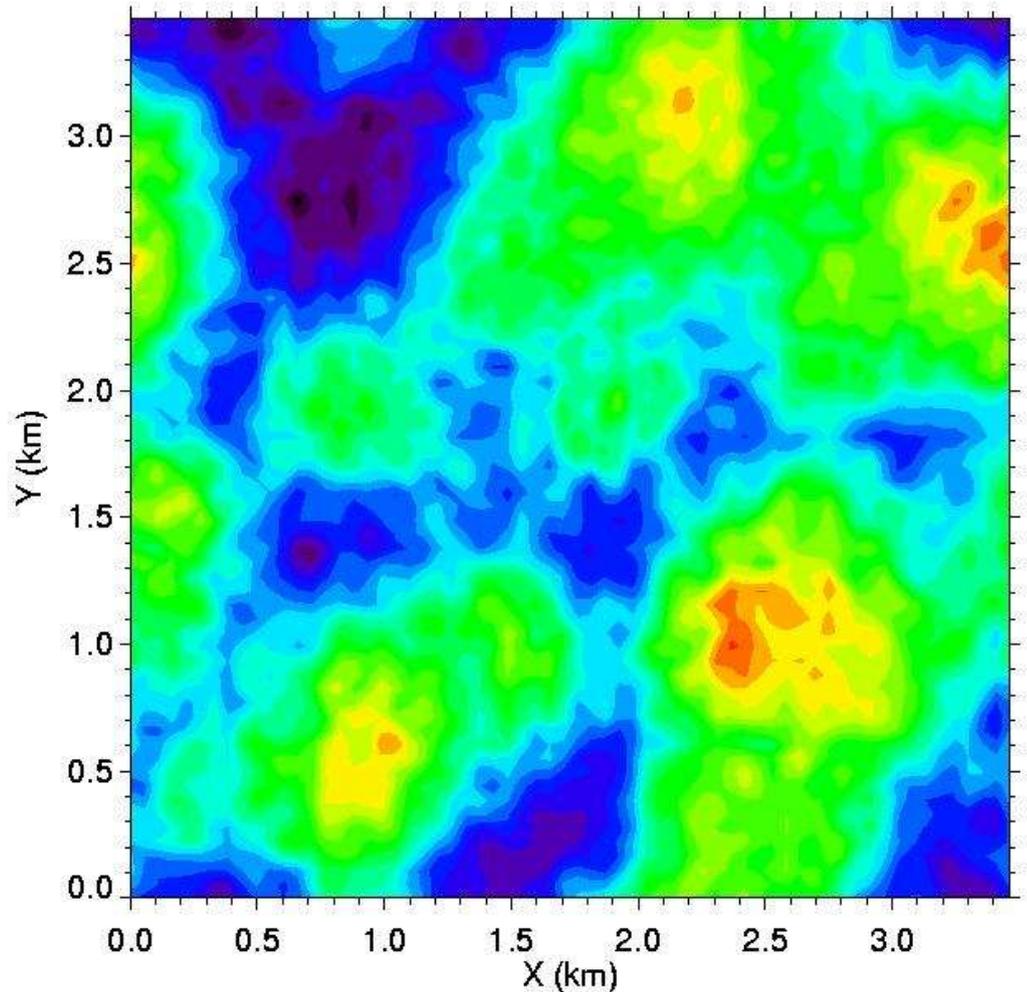
Upwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^6$



Upwelling Flux at Z=1.00 km

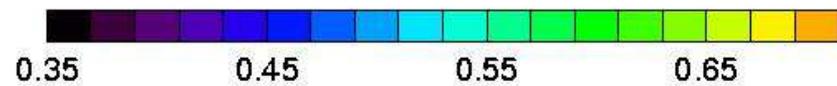
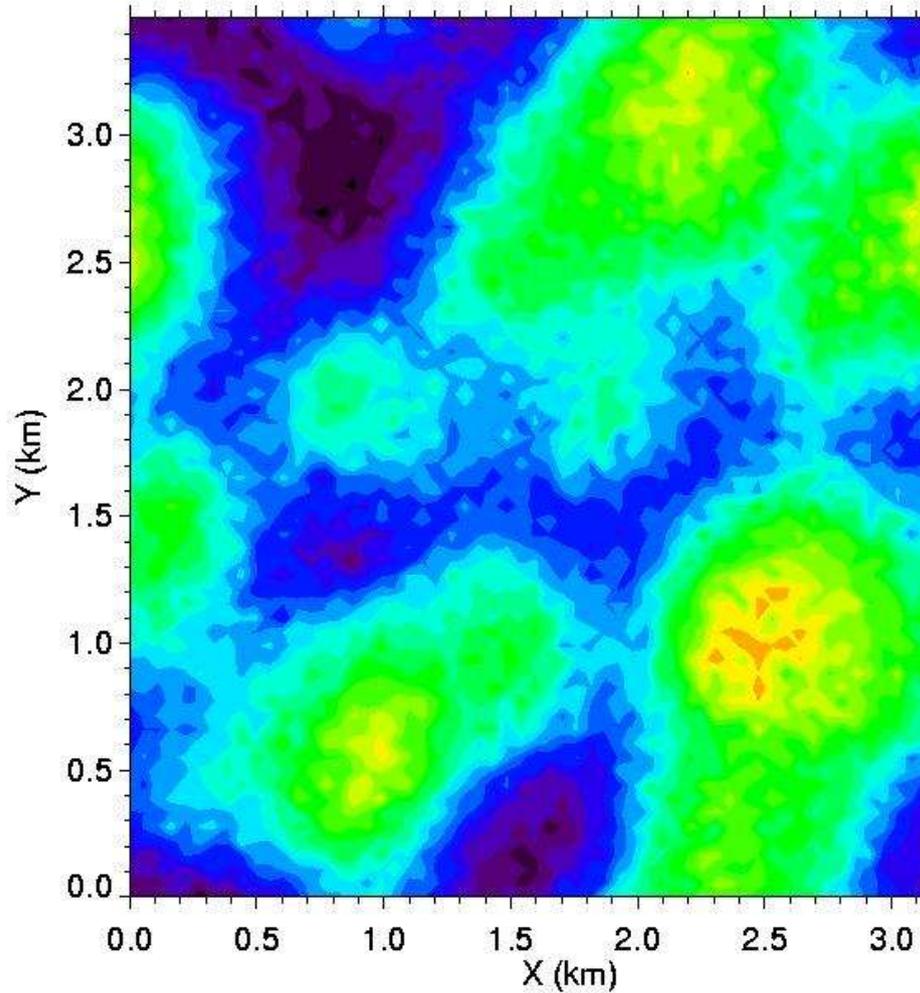
SHDOM: $N_{\mu}=4$ splitacc=0.10



Upwelling Flux at Z=1.00 km

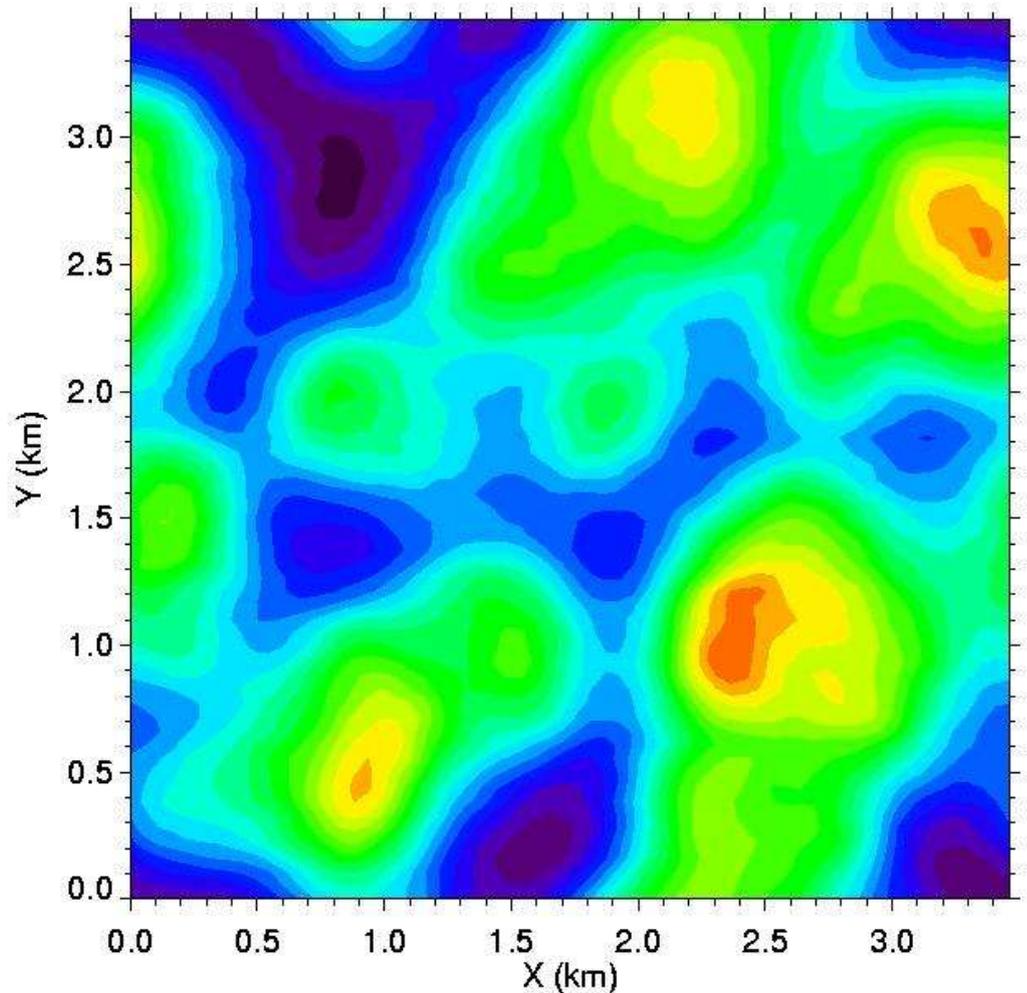
Upwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^7$



Upwelling Flux at Z=1.00 km

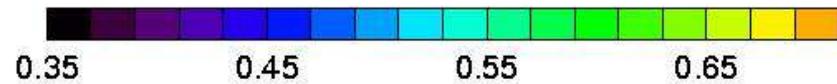
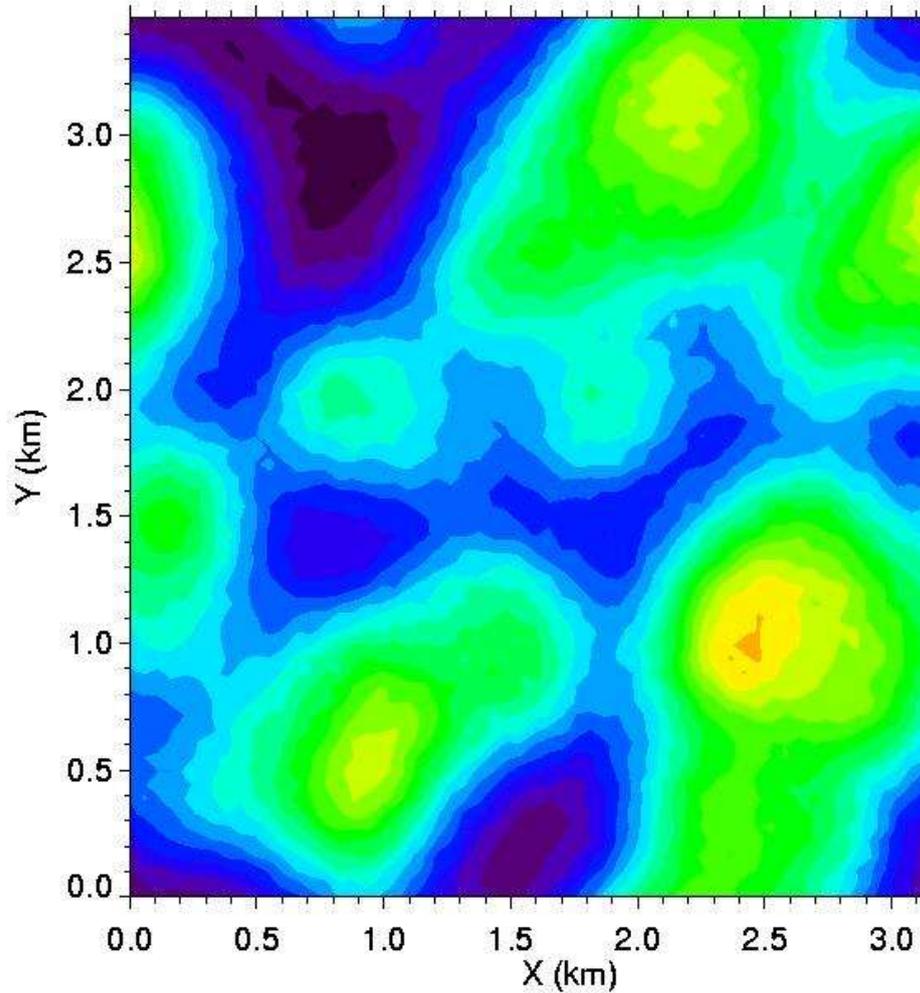
SHDOM: $N_{\mu}=8$ splitacc=0.05



Upwelling Flux at Z=1.00 km

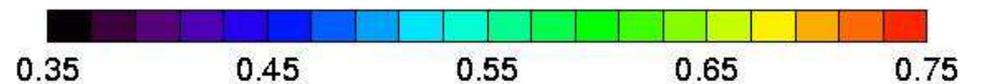
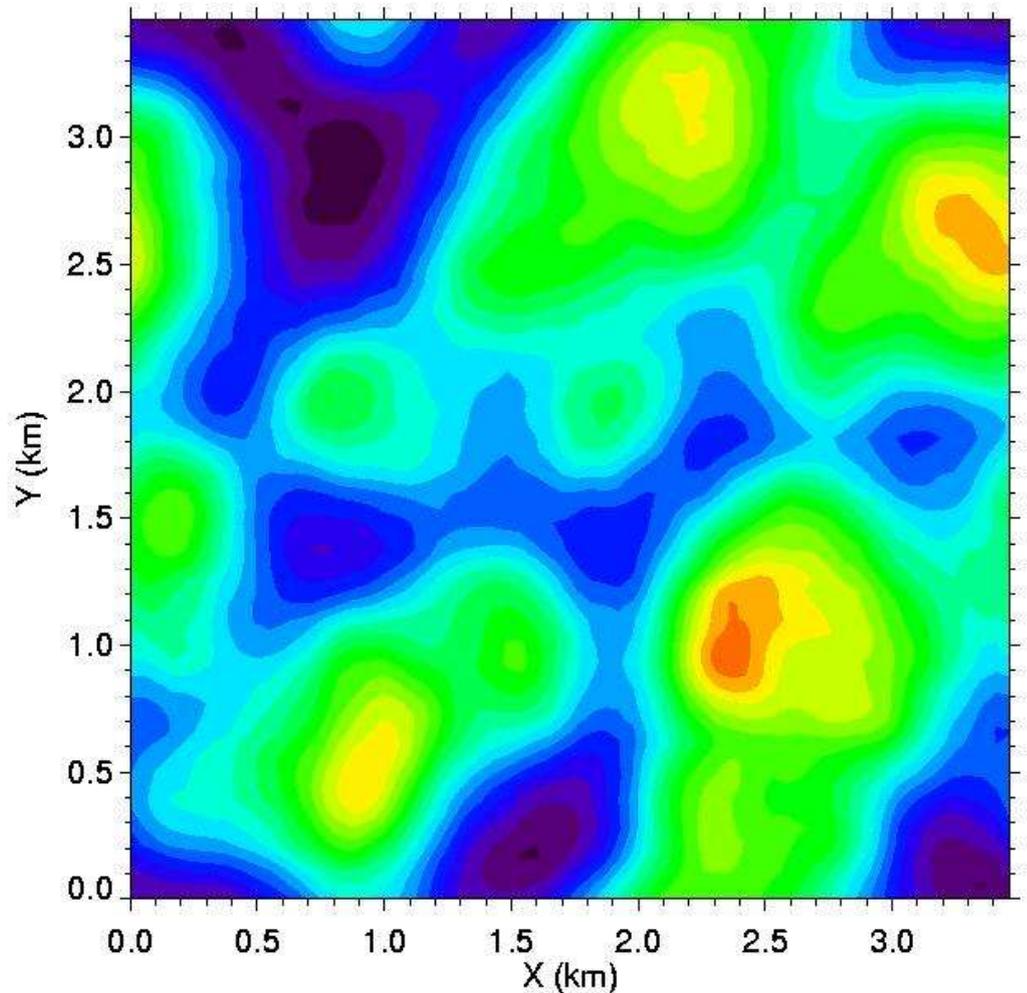
Upwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^8$



Upwelling Flux at Z=1.00 km

SHDOM: $N_{\mu}=12$ splitacc=0.03



Upwelling Flux at Z=1.00 km

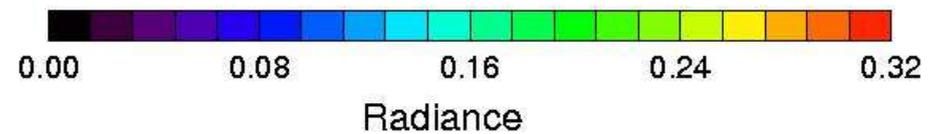
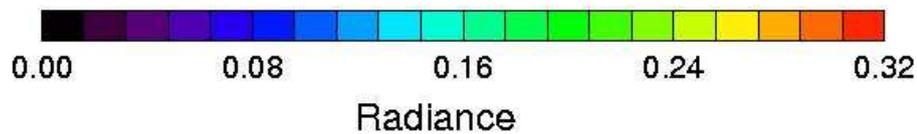
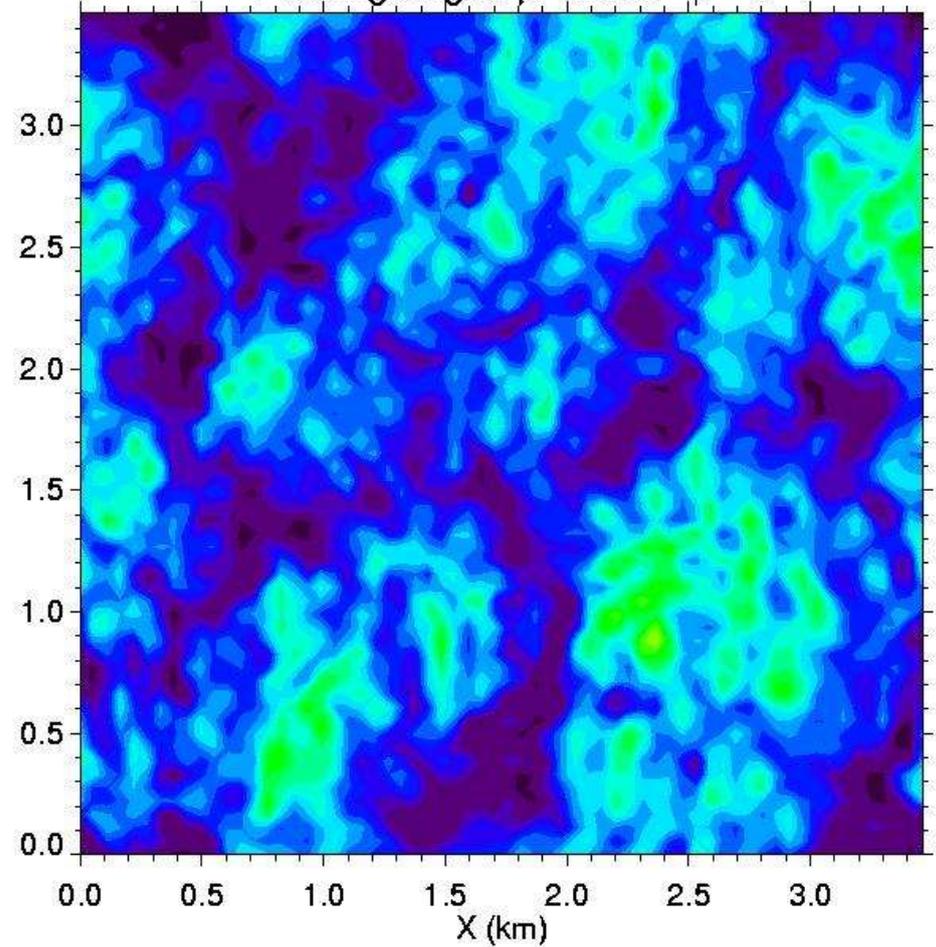
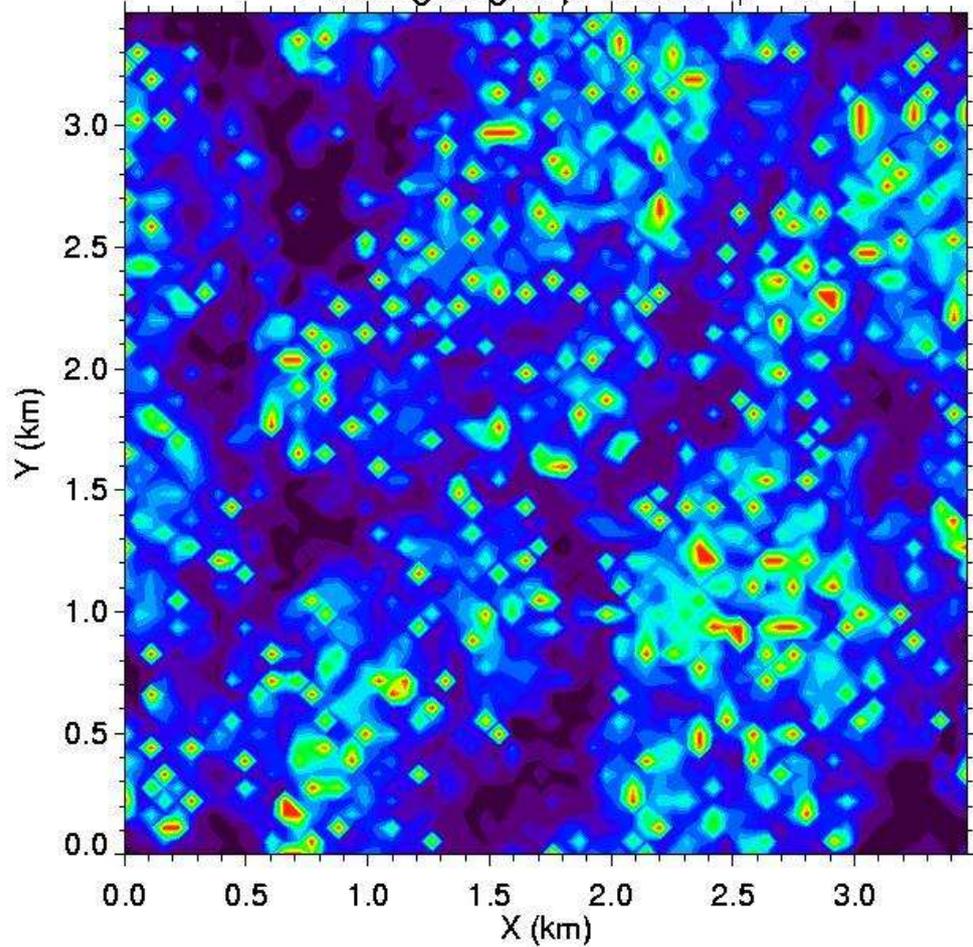
Upwelling Radiance for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^6$

SHDOM: $N_{\mu}=2$ splitacc=0.10

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



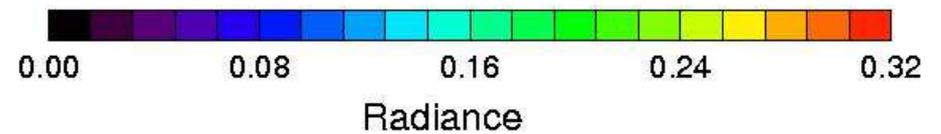
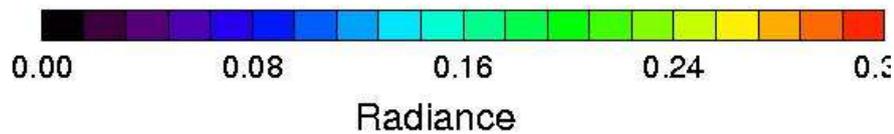
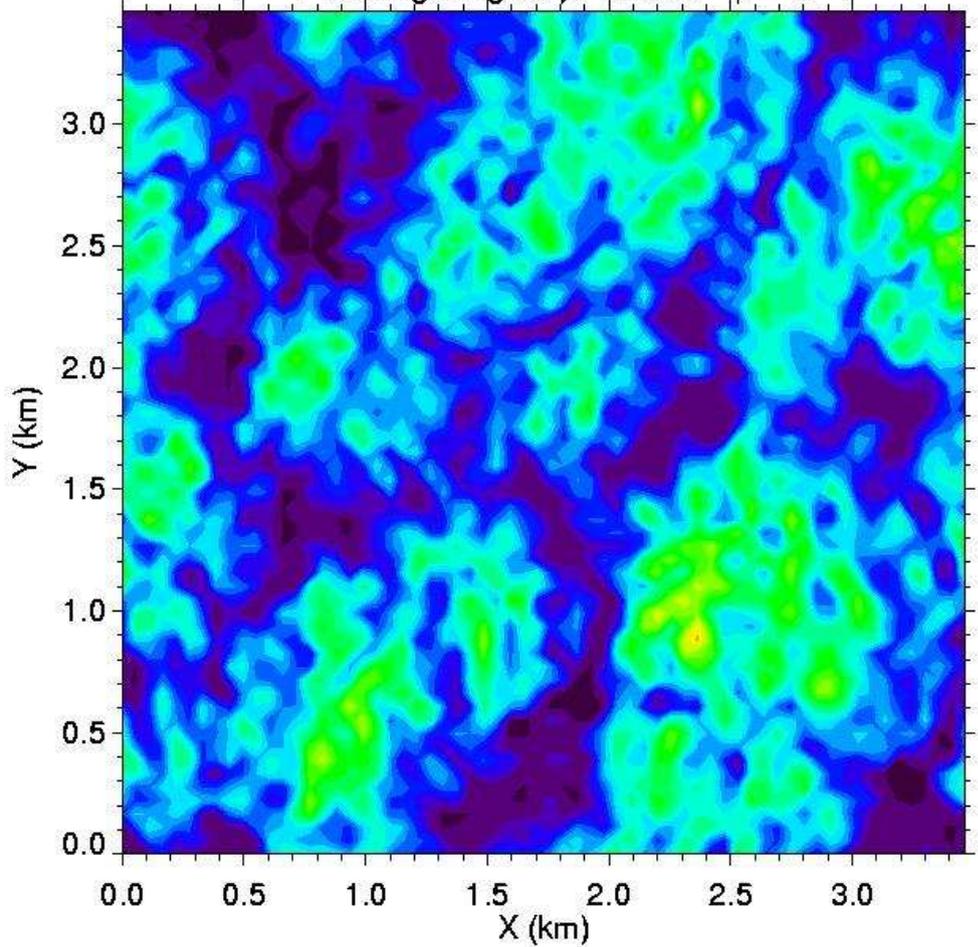
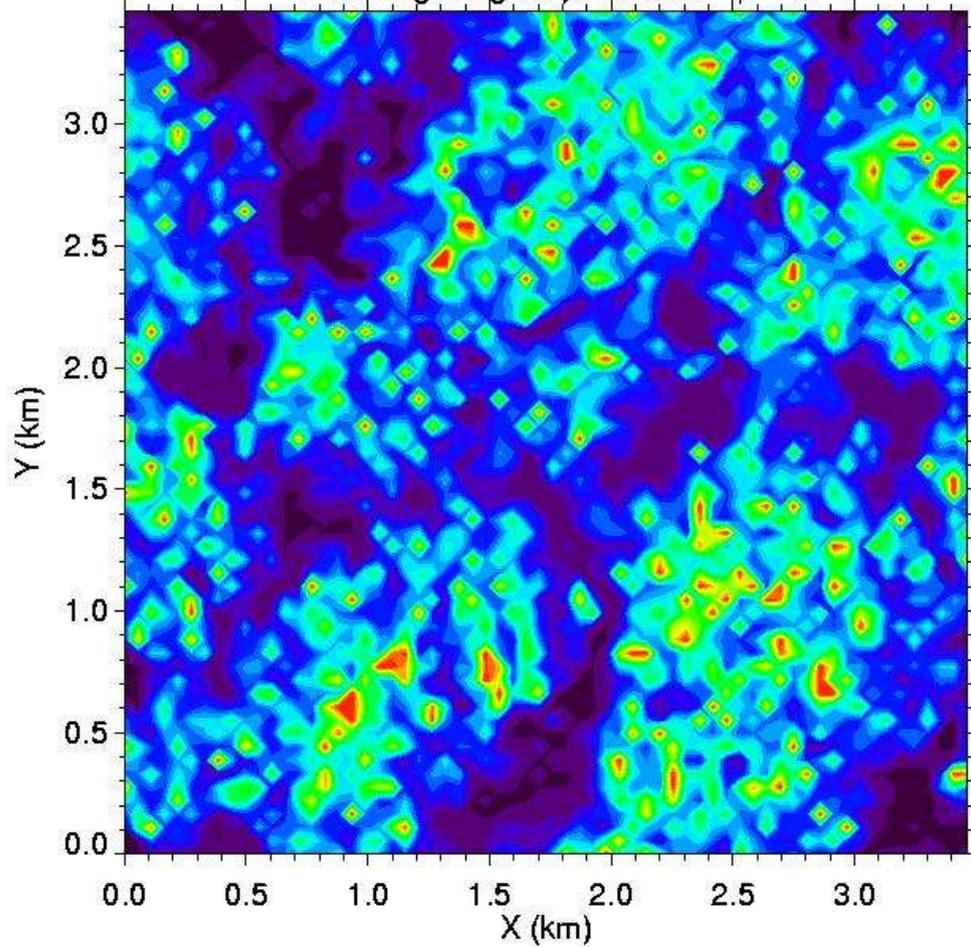
Upwelling Radiance for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^7$

SHDOM: $N_{\mu}=4$ splitacc=0.10

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



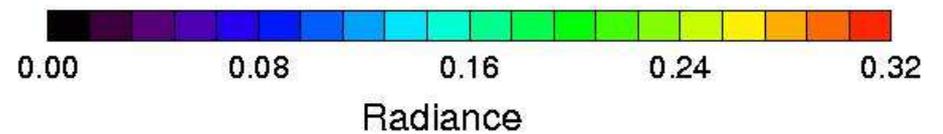
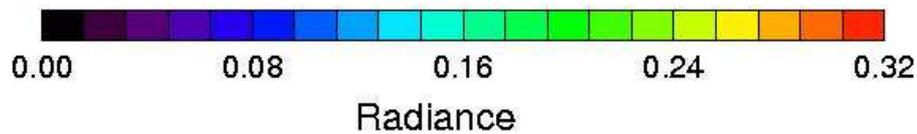
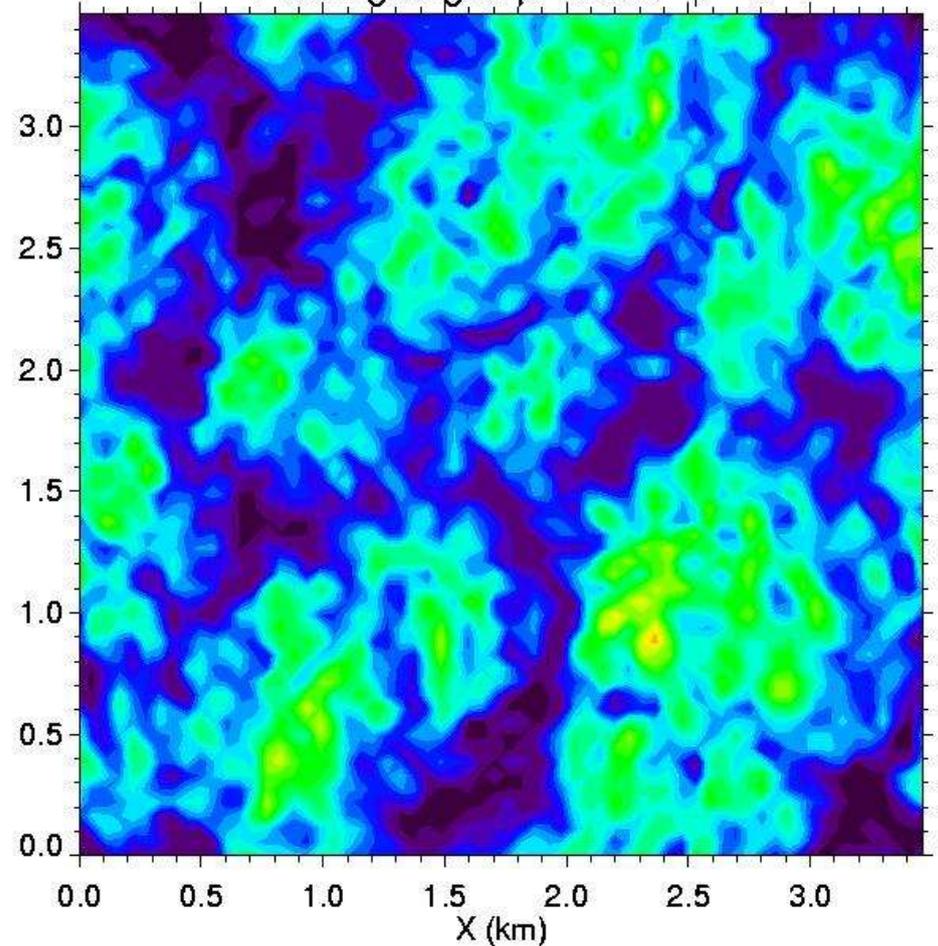
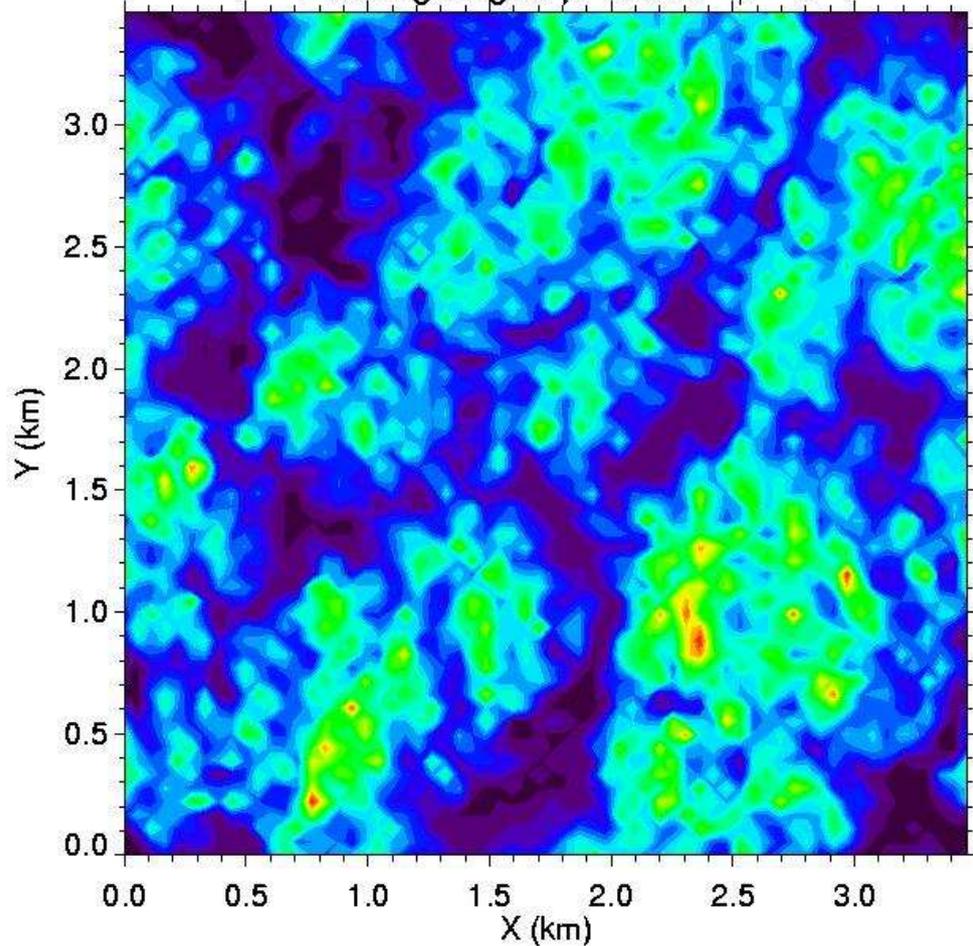
Upwelling Radiance for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^8$

SHDOM: $N_{\mu}=8$ splitacc=0.05

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



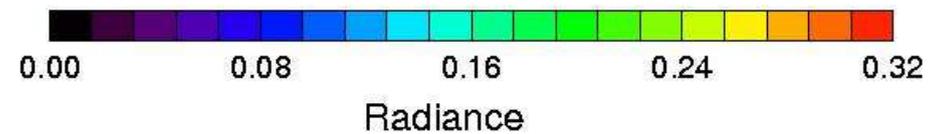
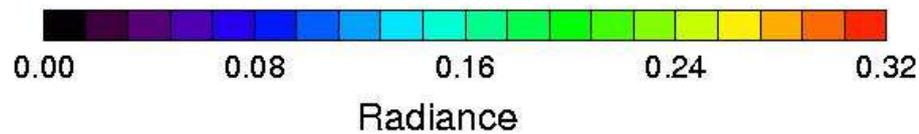
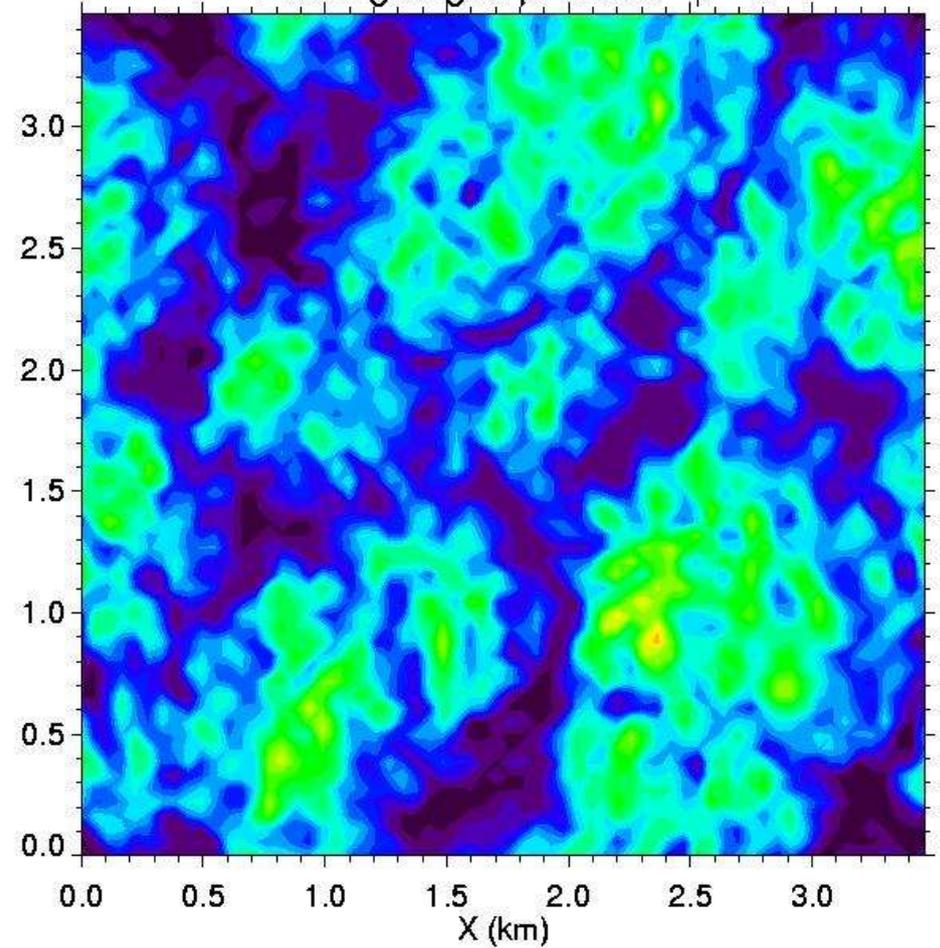
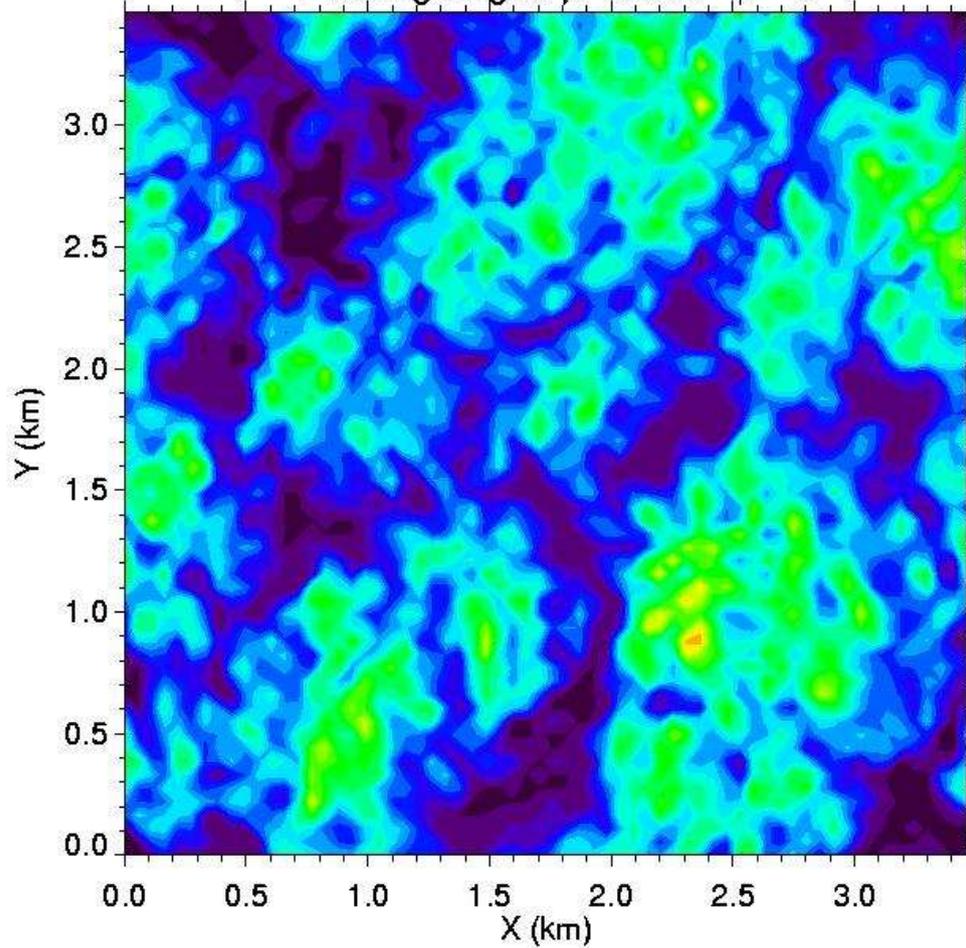
Upwelling Radiance for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^9$

SHDOM: $N_{\mu}=12$ splitacc=0.03

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



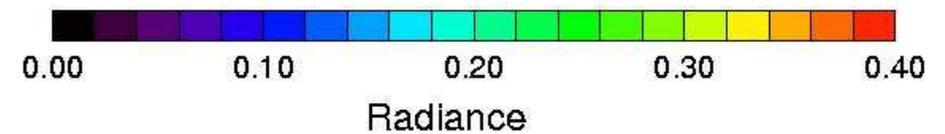
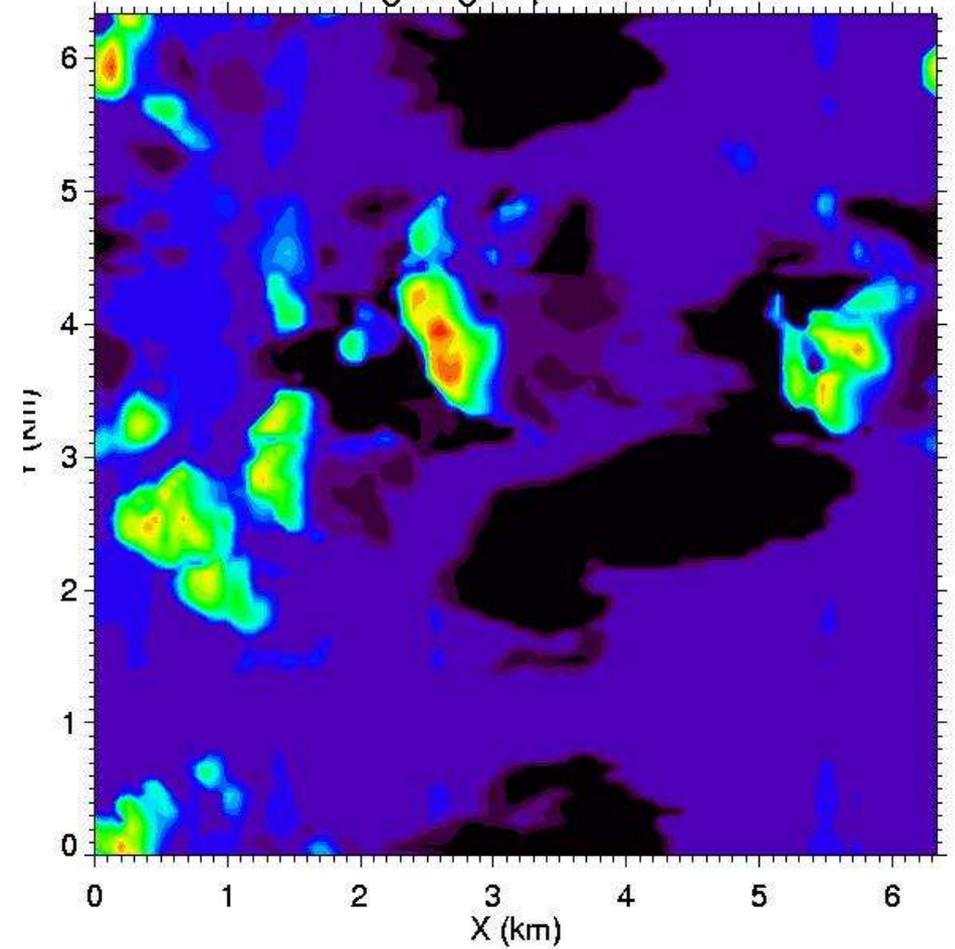
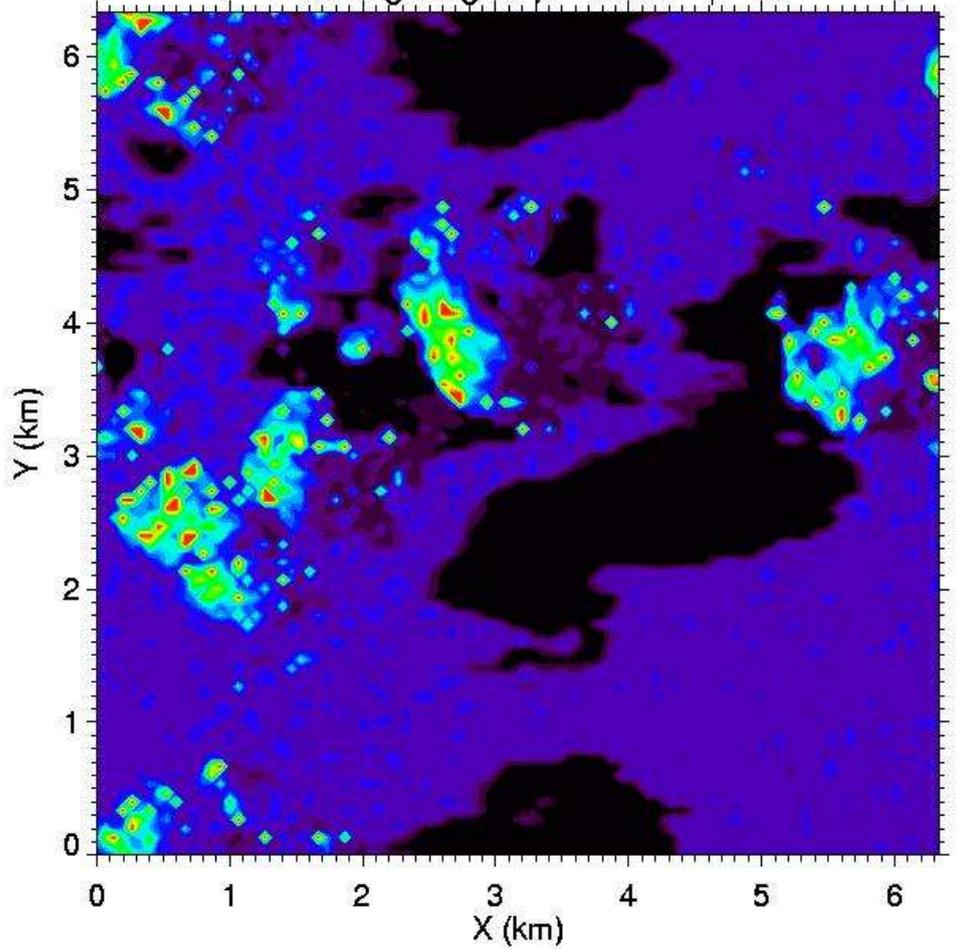
Upwelling Radiance for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^6$

SHDOM: $N_{\mu}=2$ splitacc=0.10

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



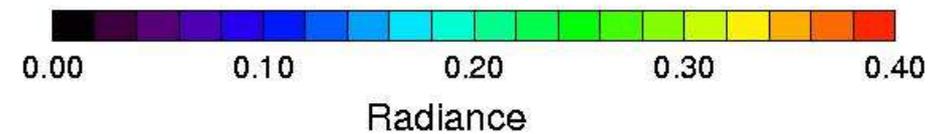
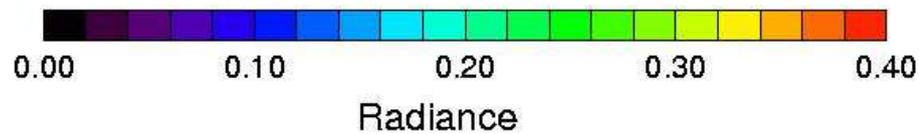
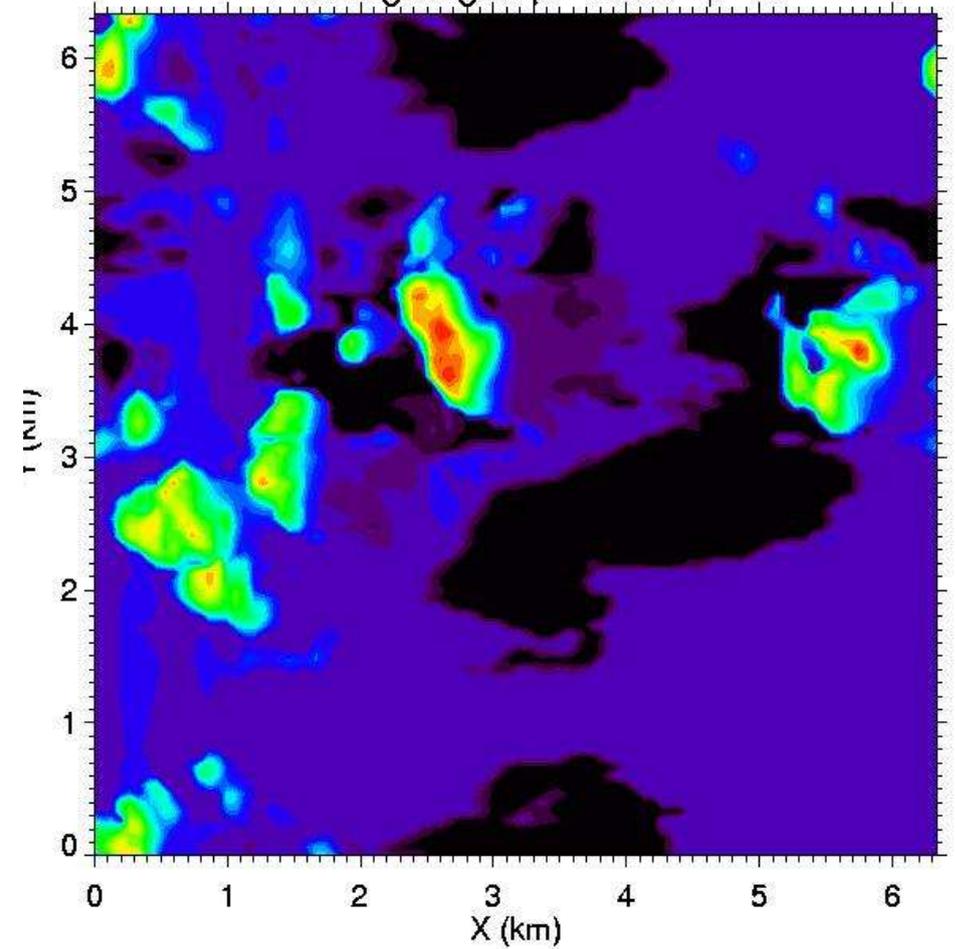
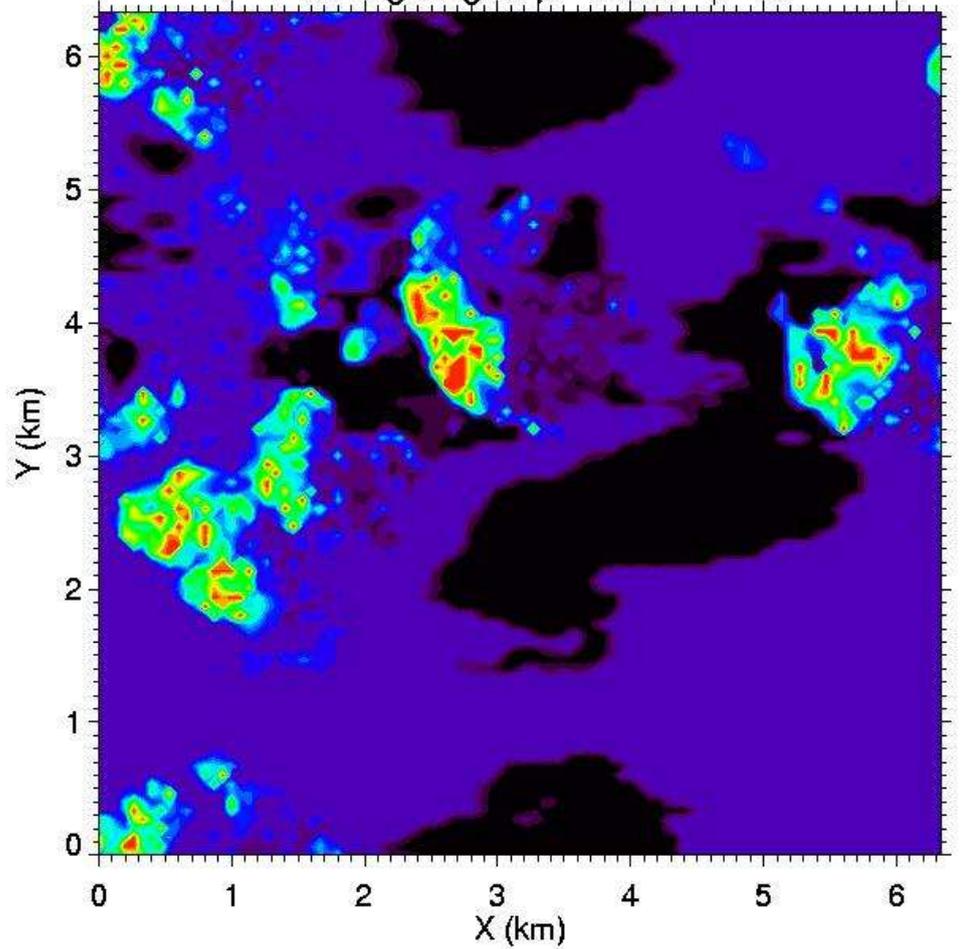
Upwelling Radiance for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^7$

SHDOM: $N_{\mu}=4$ splitacc=0.10

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



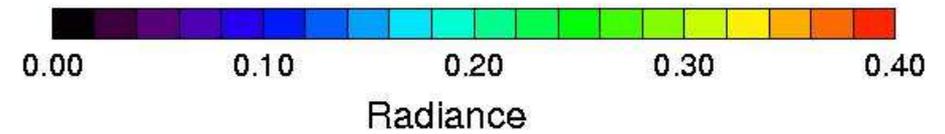
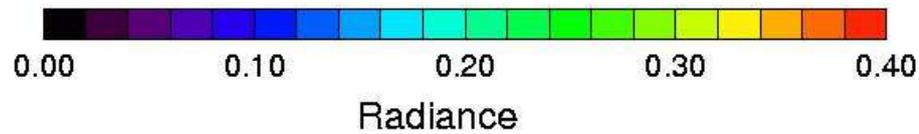
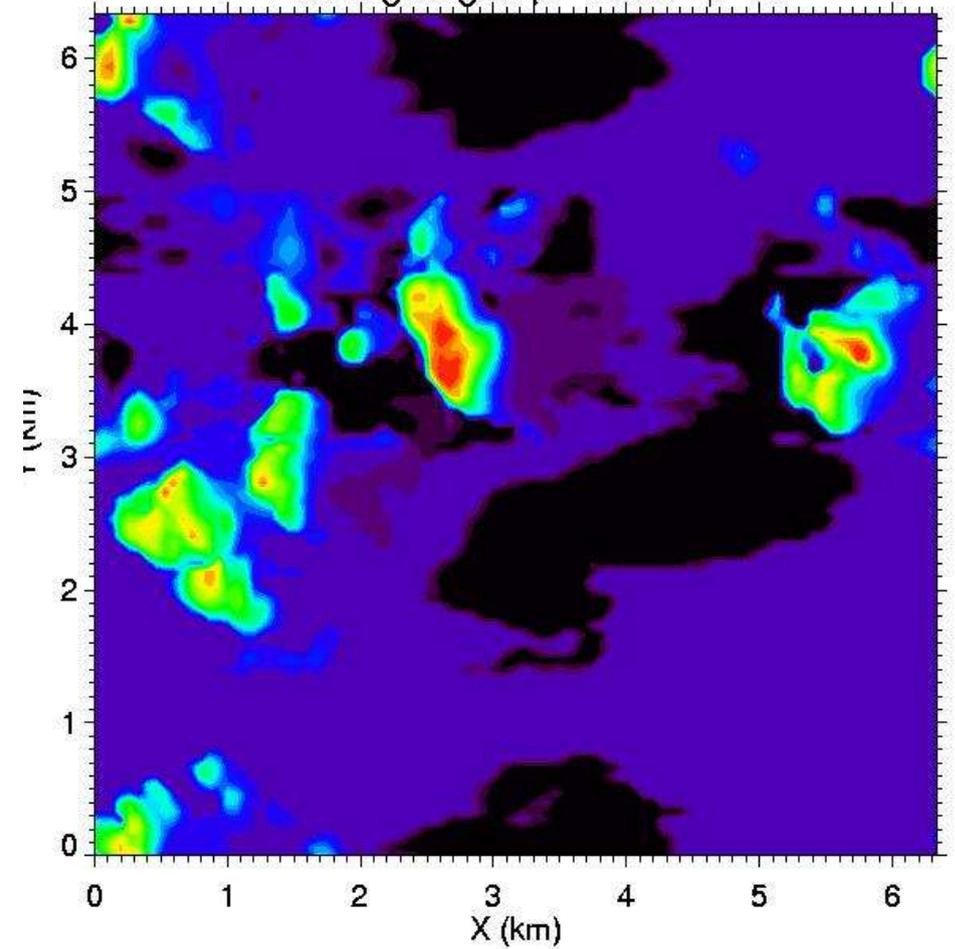
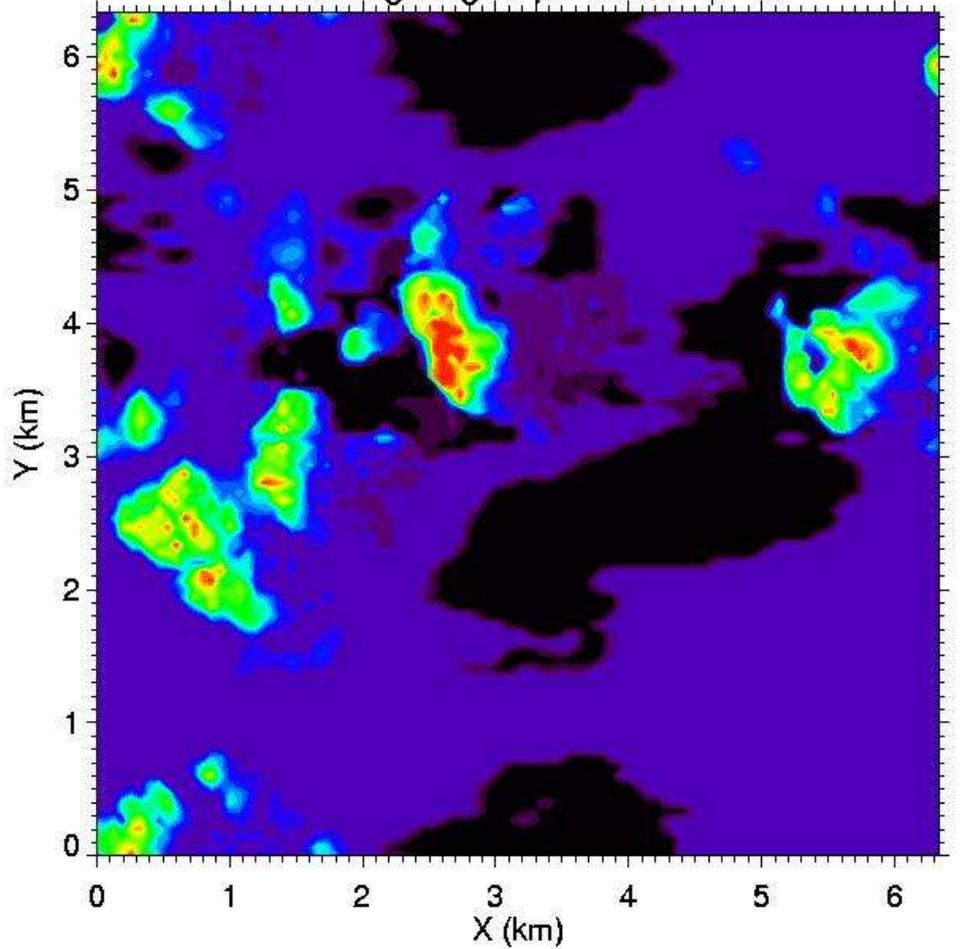
Upwelling Radiance for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^8$

SHDOM: $N_{\mu}=8$ splitacc=0.05

Viewing angle: $\mu=1.000$ $\phi=0$

Viewing angle: $\mu=1.000$ $\phi=0$



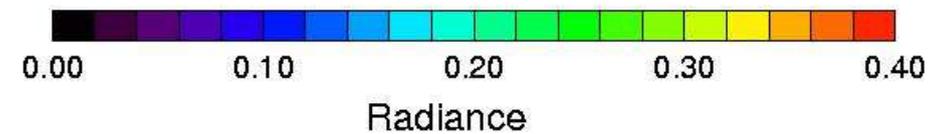
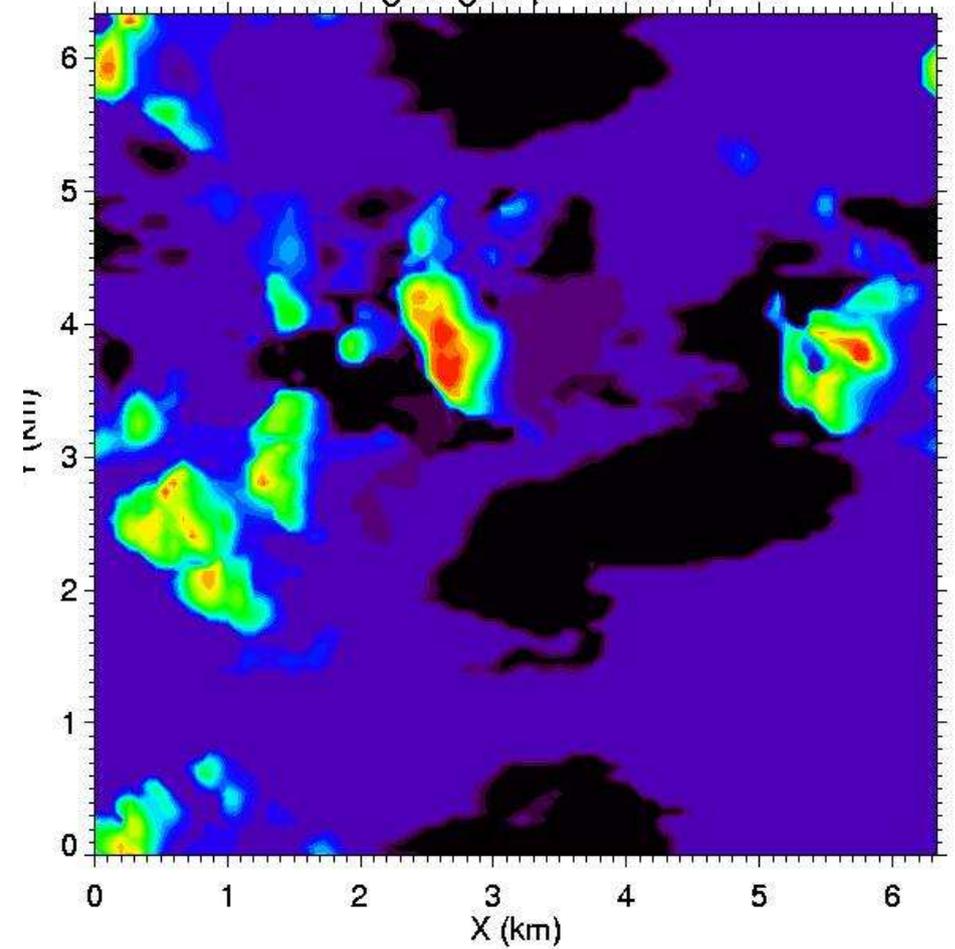
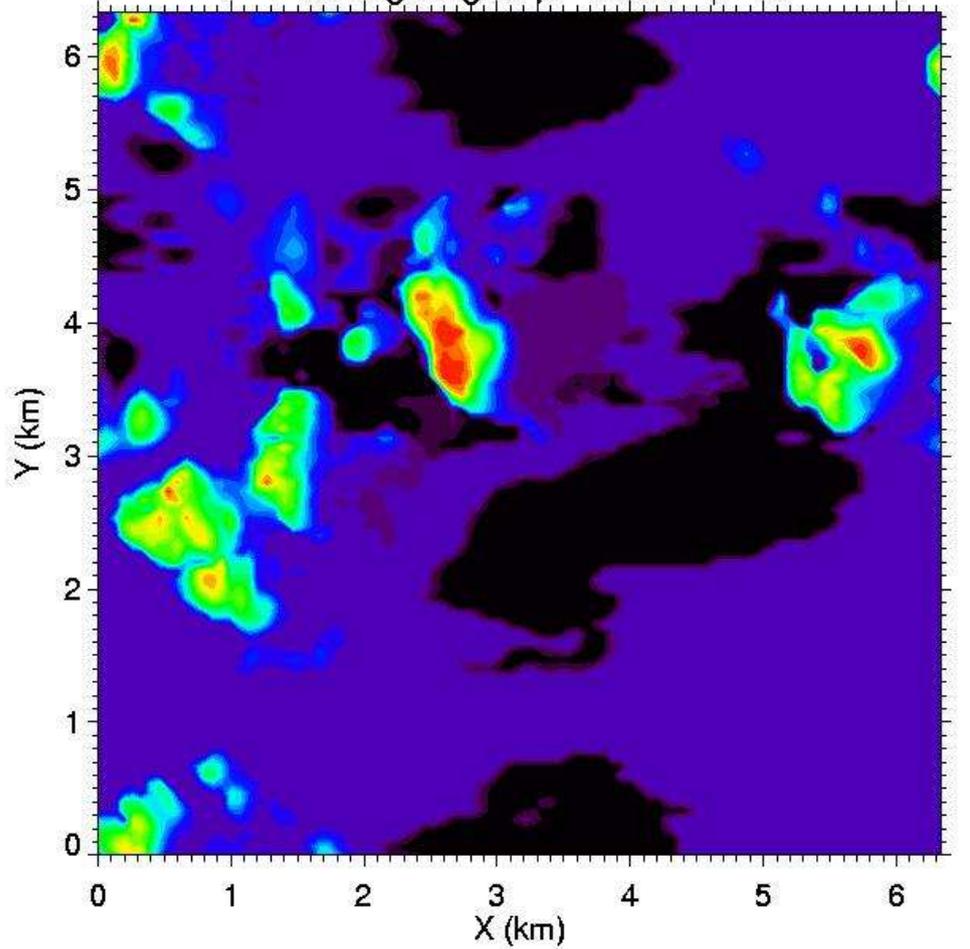
Upwelling Radiance for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^9$

SHDOM: $N_{\mu}=12$ splitacc=0.03

Viewing angle: $\mu=1.000$ $\phi=0$

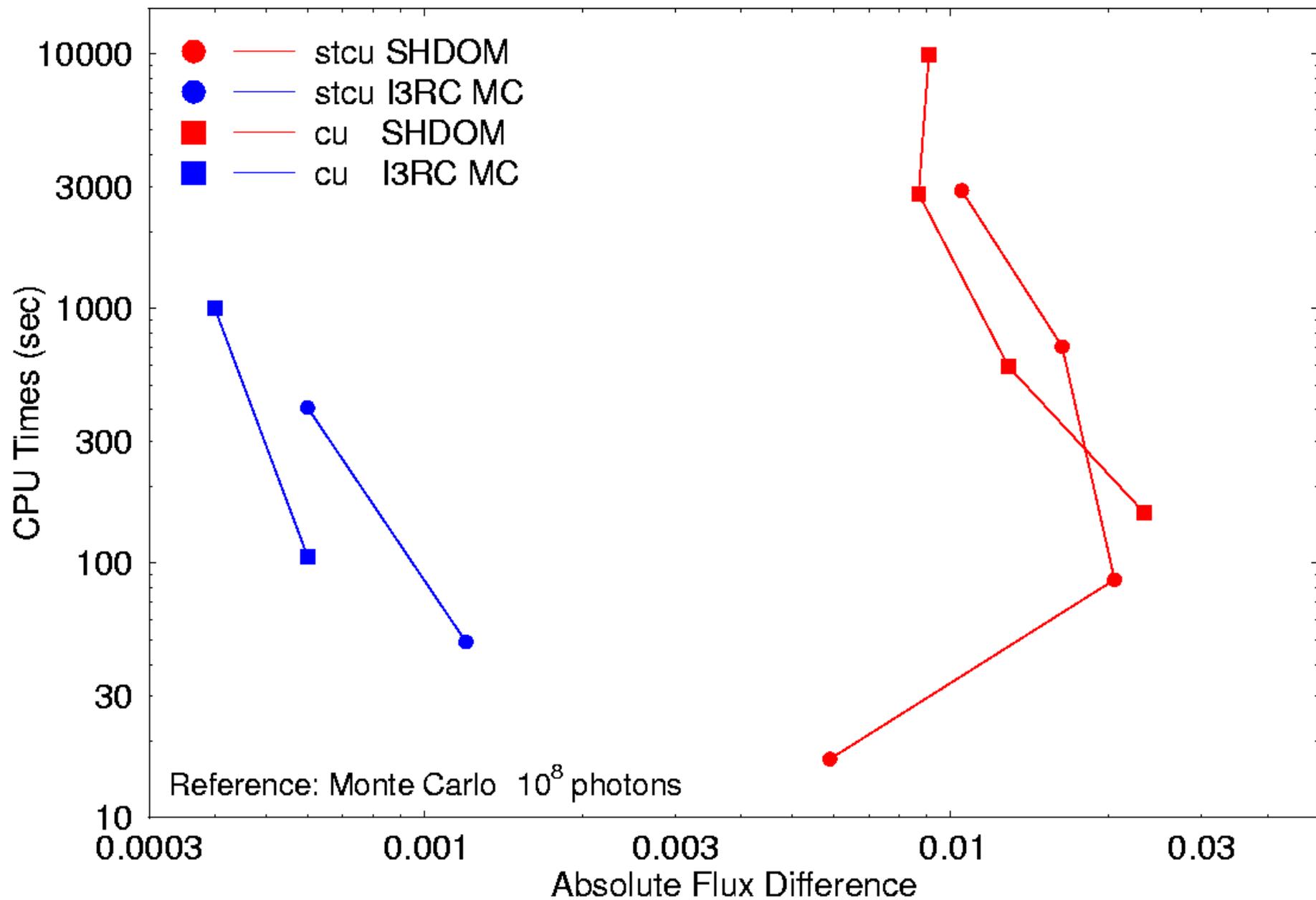
Viewing angle: $\mu=1.000$ $\phi=0$



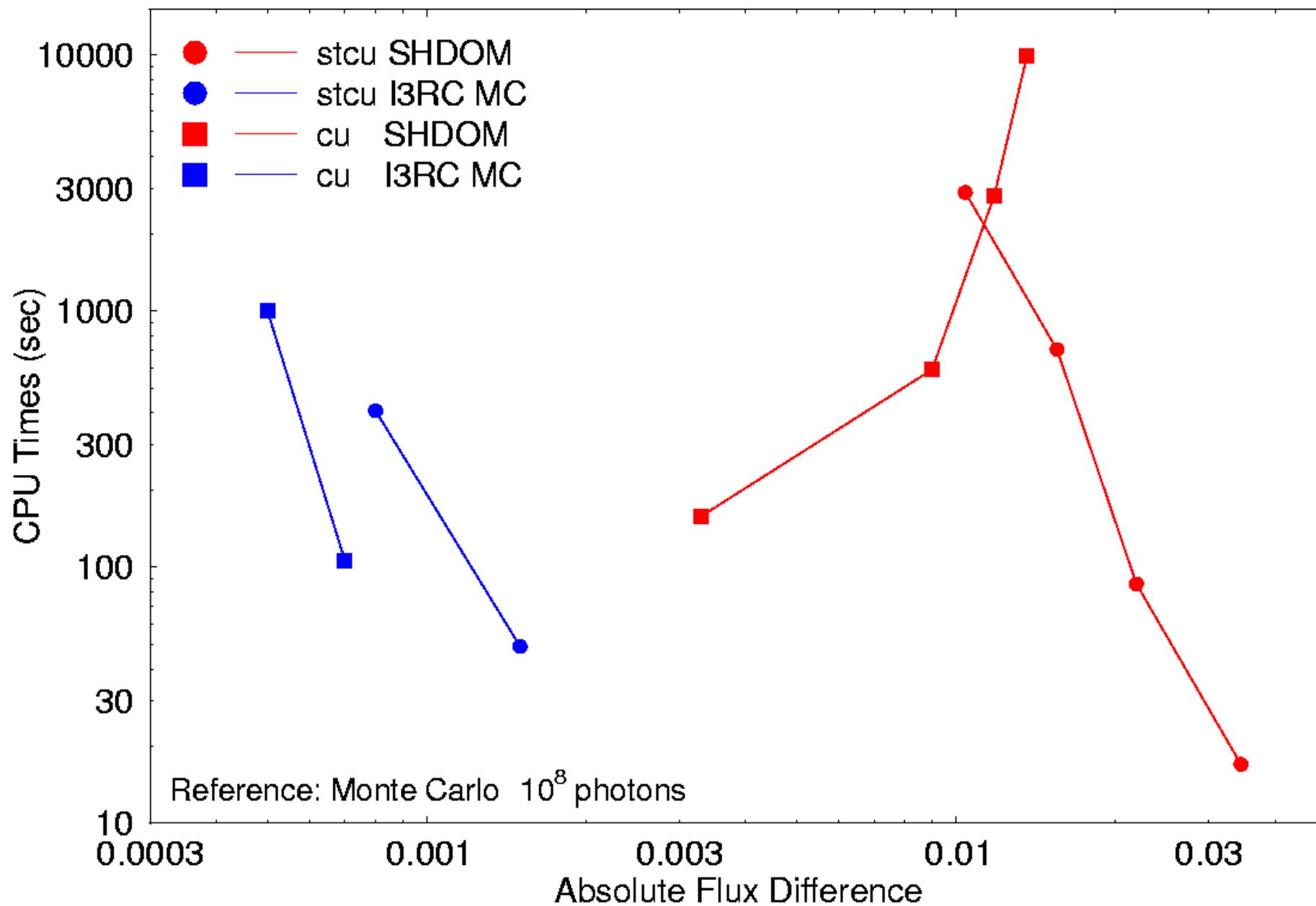
Comparing Results

- SHDOM optical property files input to I3RC MC driver program
- **Problem:** SHDOM interpolates properties between grid points, while I3RC MC assumes uniform grid cells.
 - Attempted solution: MC driver program interpolates properties vertically between levels to make grid cell values.
- Compare: domain average boundary fluxes, domain average absorption profile, pixel fluxes, domain average radiances, and pixel radiances.
- Use difference values or rms differences in radiative quantities

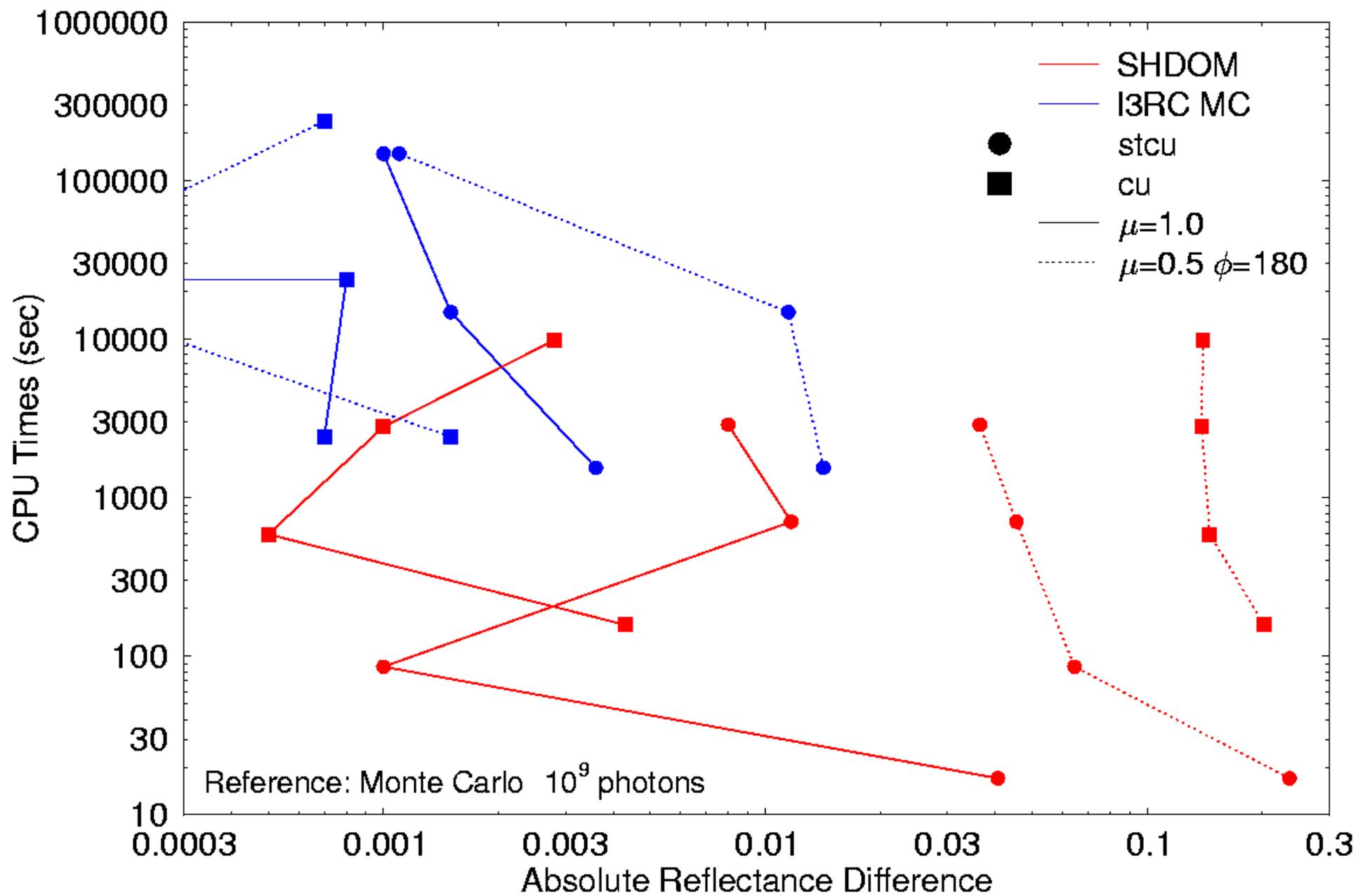
0.67 μm Domain Average Upwelling Flux Comparison



0.67 μm Domain Average Downwelling Flux Comparison

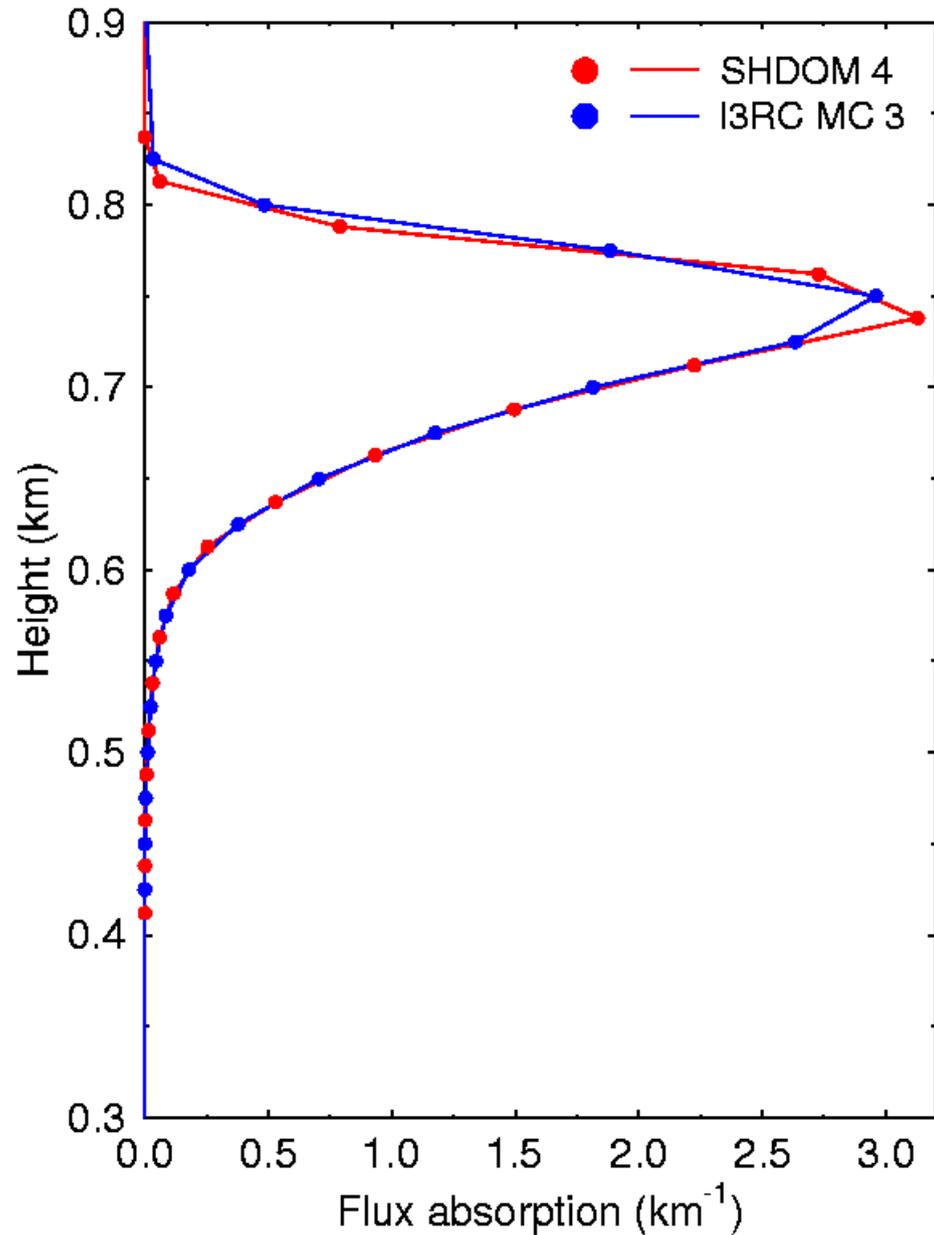


0.67 μm Domain Average Upwelling Reflectance Comparison

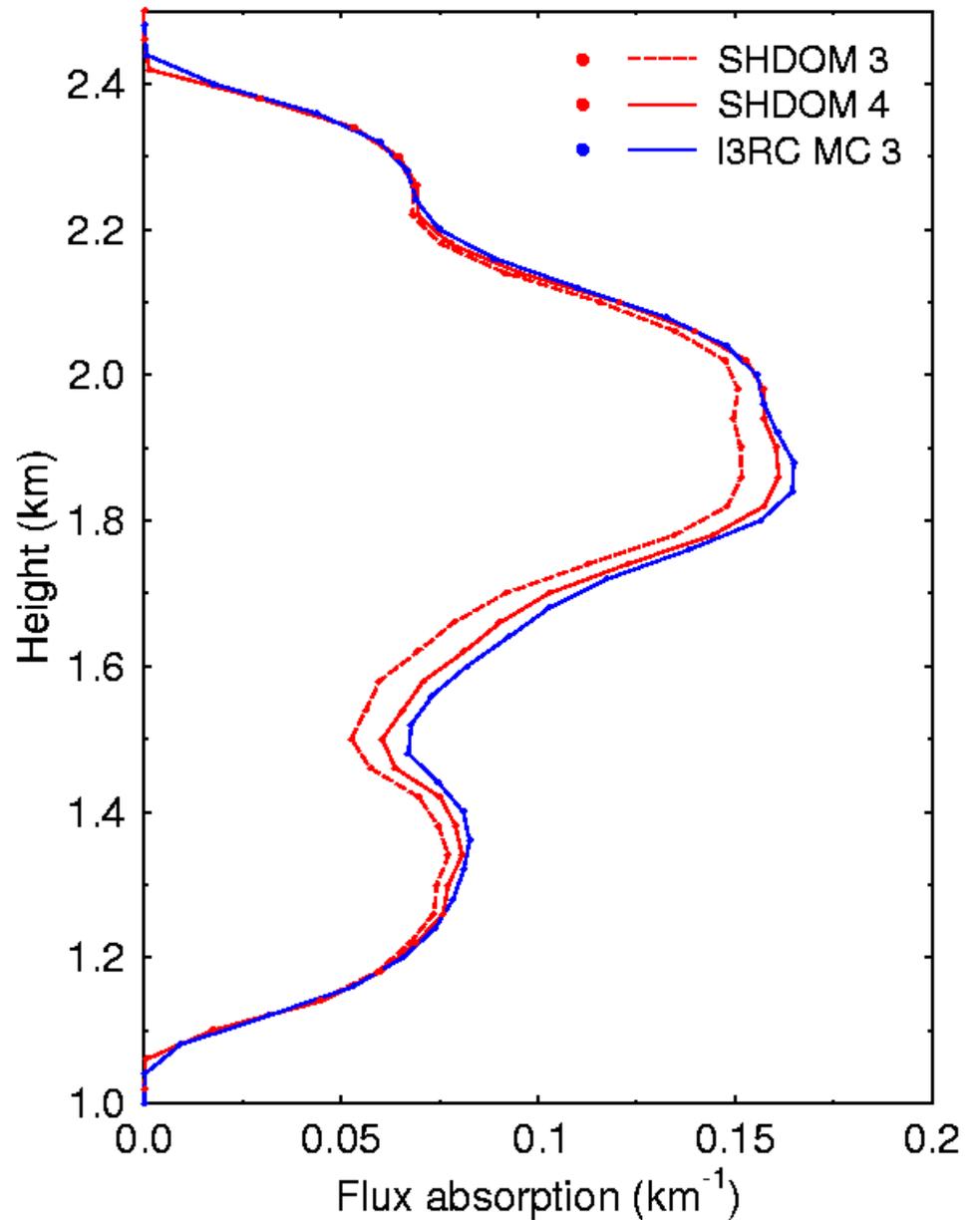


2.13 μm Absorption Profiles

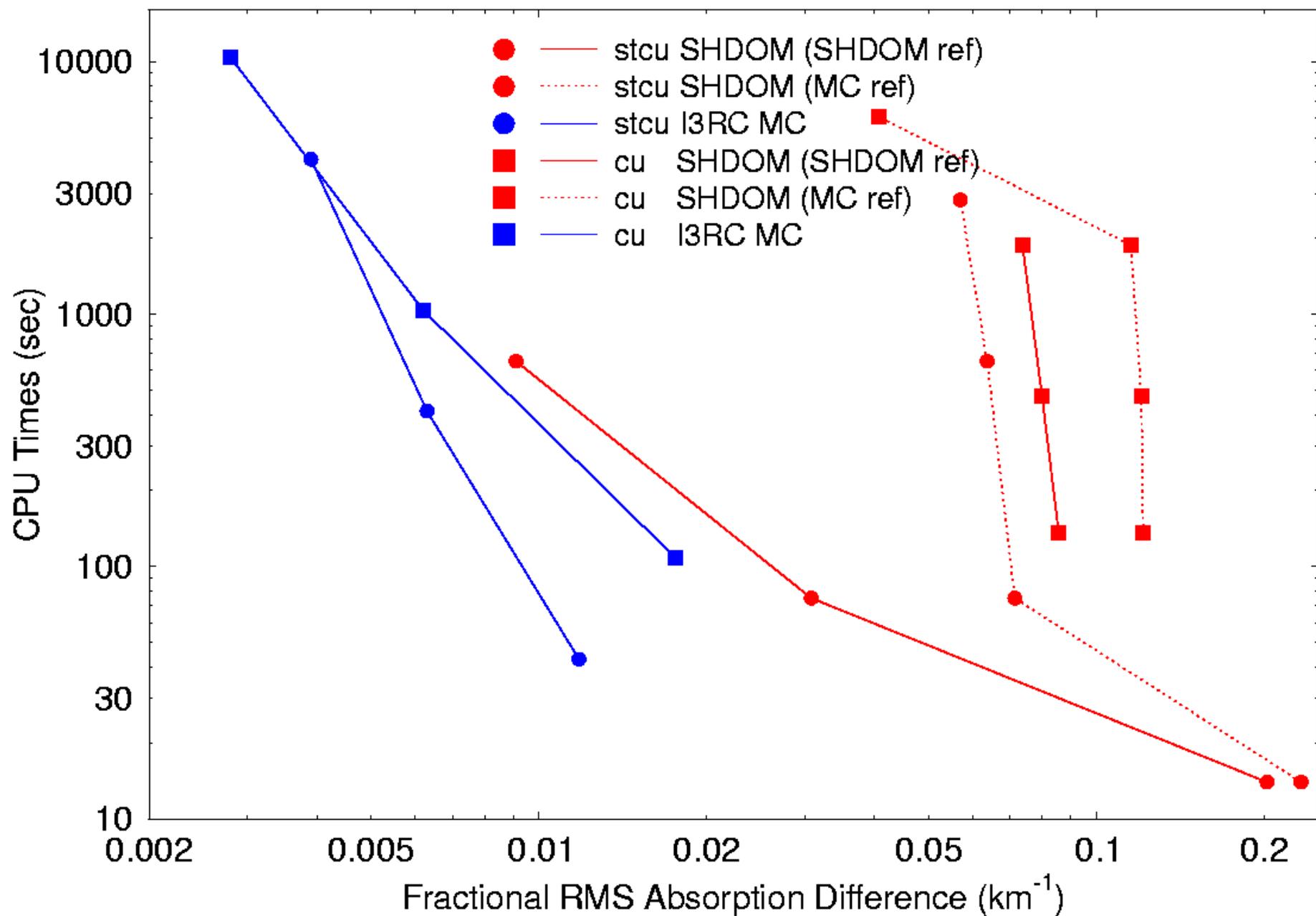
Stratocumulus



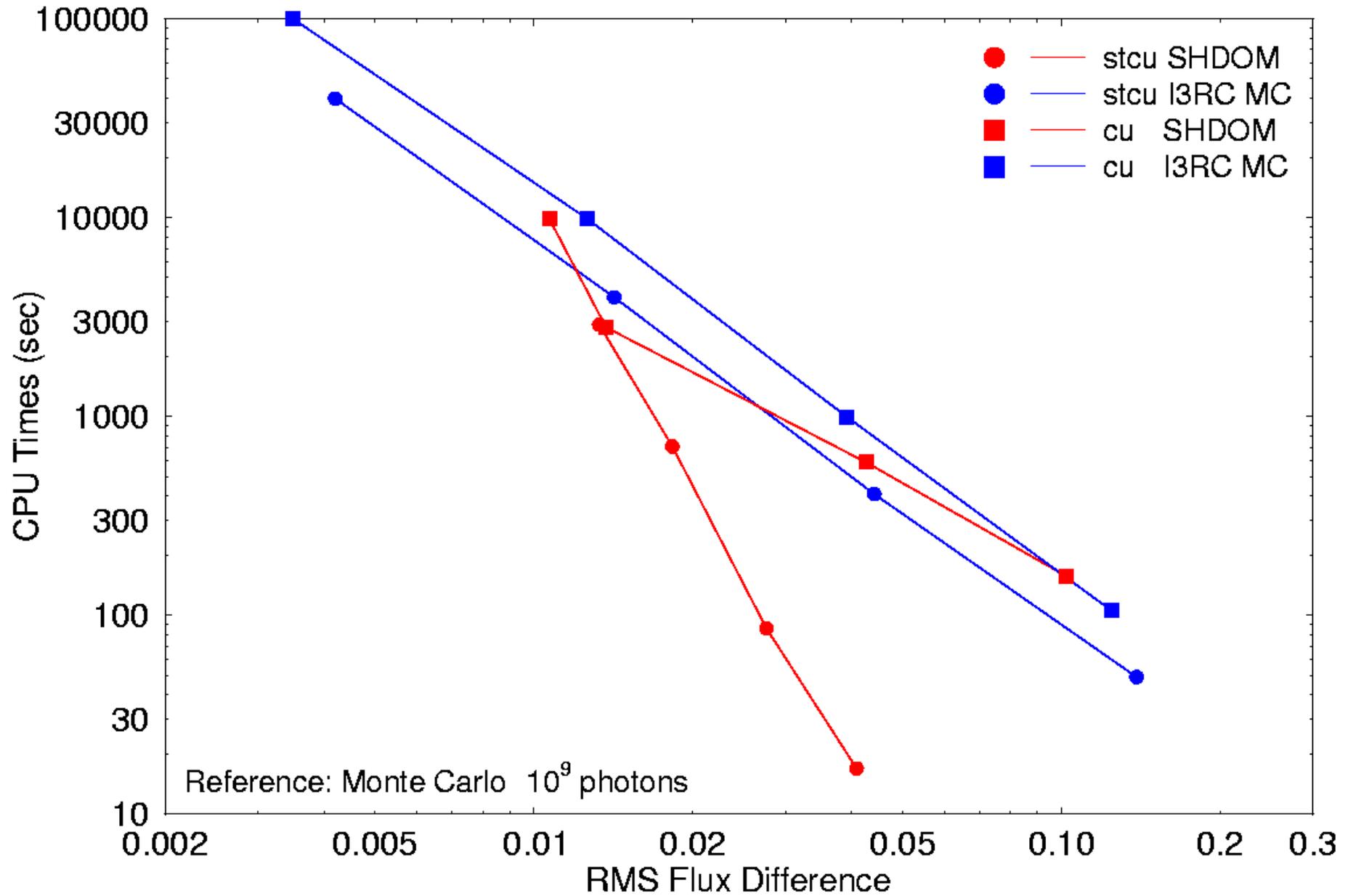
Cumulus



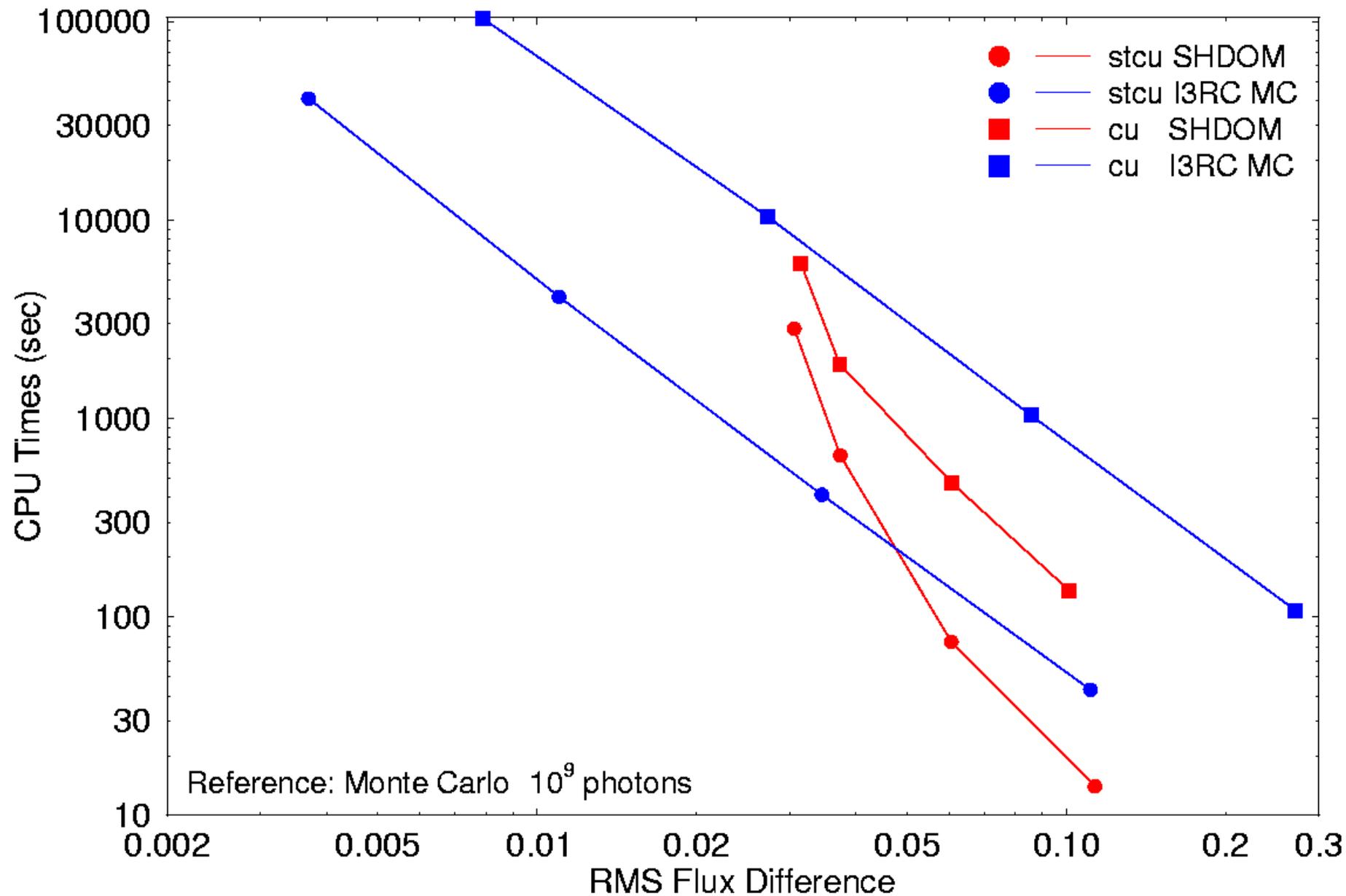
2.13 μm Absorption Profile Accuracy Comparison



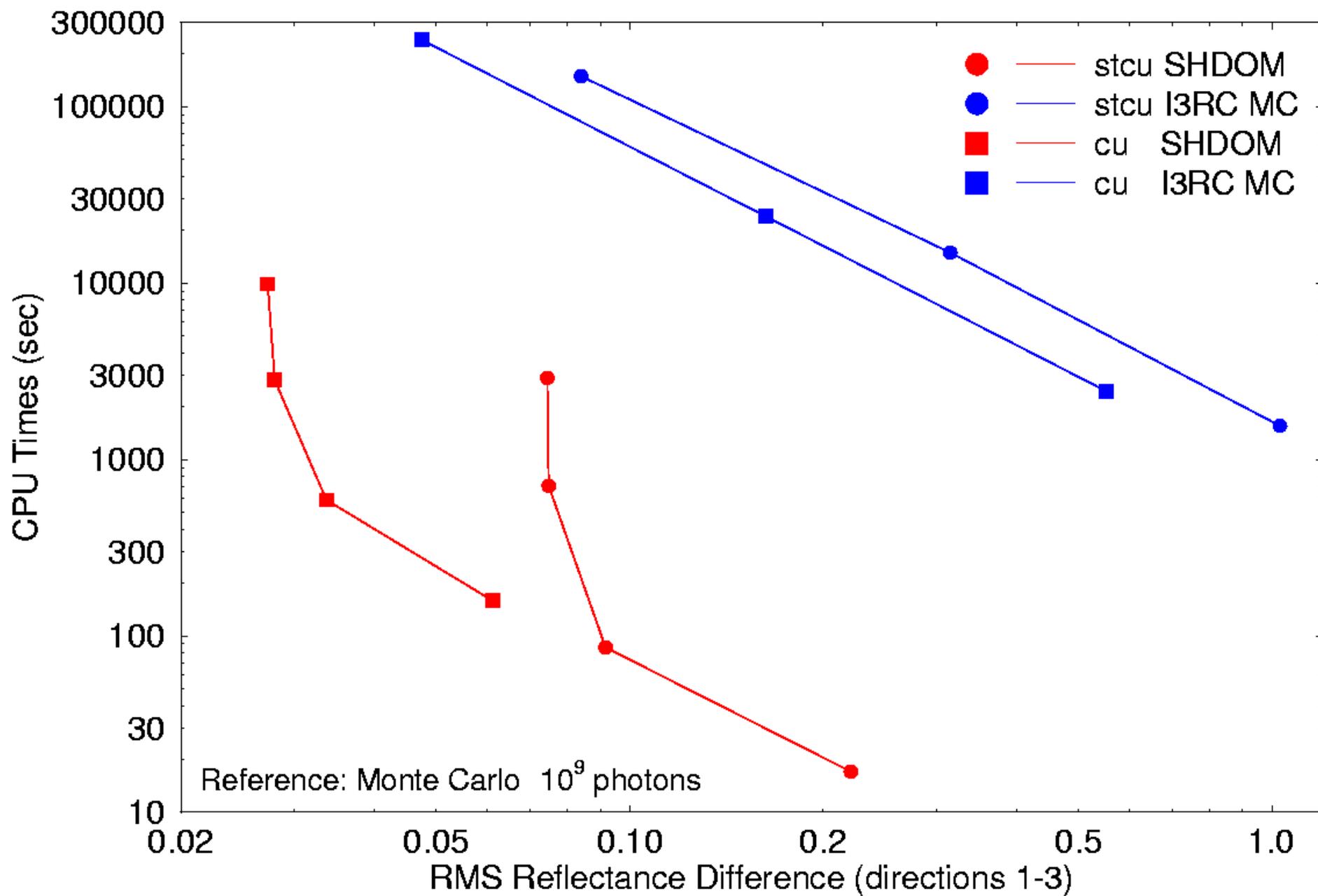
0.67 μm Pixel Level Upwelling Flux Comparison



2.13 μm Pixel Level Downwelling Flux Comparison



0.67 μm Pixel Level Reflectance Comparison



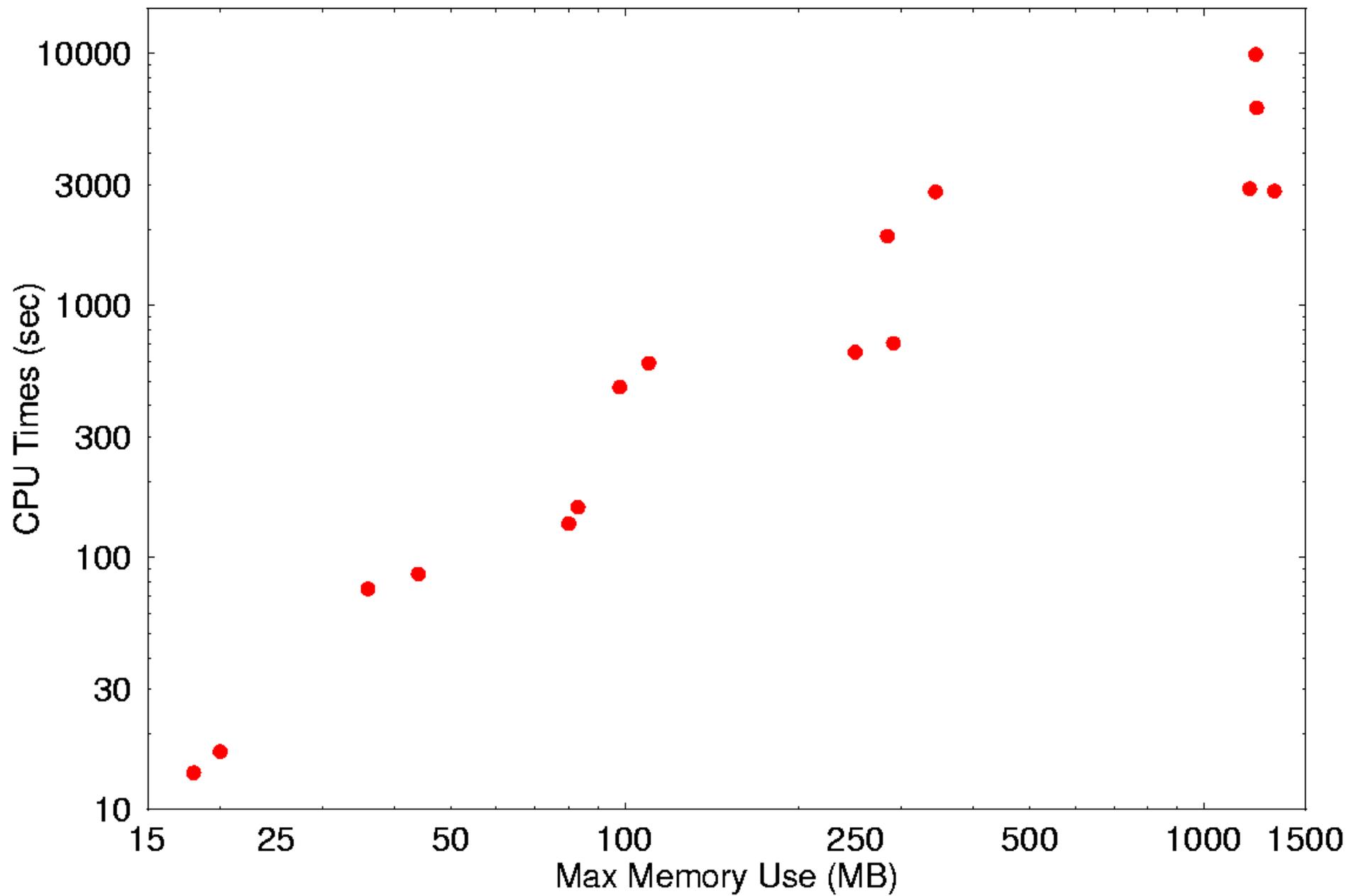
Monte Carlo Computational Speed

- Compare Evans and I3RC Monte Carlo for flux calculations with 10^7 photons.
- Both use maximum cross section method, but Evans model can have separate slabs with different max σ .

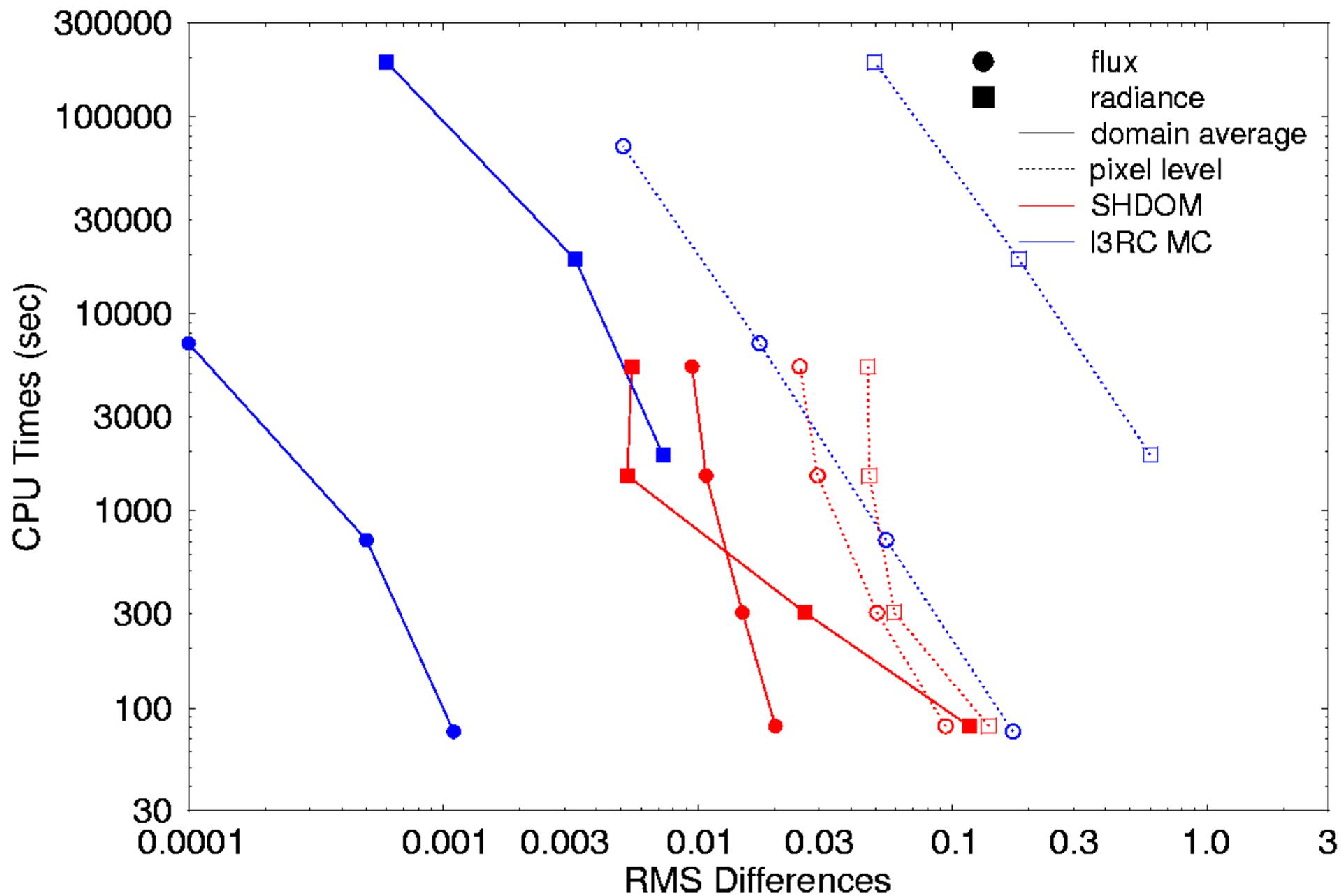
	CPU Time Ratio (I3RC/Evans)	
Cloud	$N_{\text{slab}}=1$	$N_{\text{slab}}=2$
stcu	1.17	1.82
cu	1.46	2.32

- I3RC MC only slightly slower for fair comparison ($N_{\text{slab}}=1$).

SHDOM Memory Use



Summary of Comparisons (average over wavelengths and clouds)



Summary of Comparison

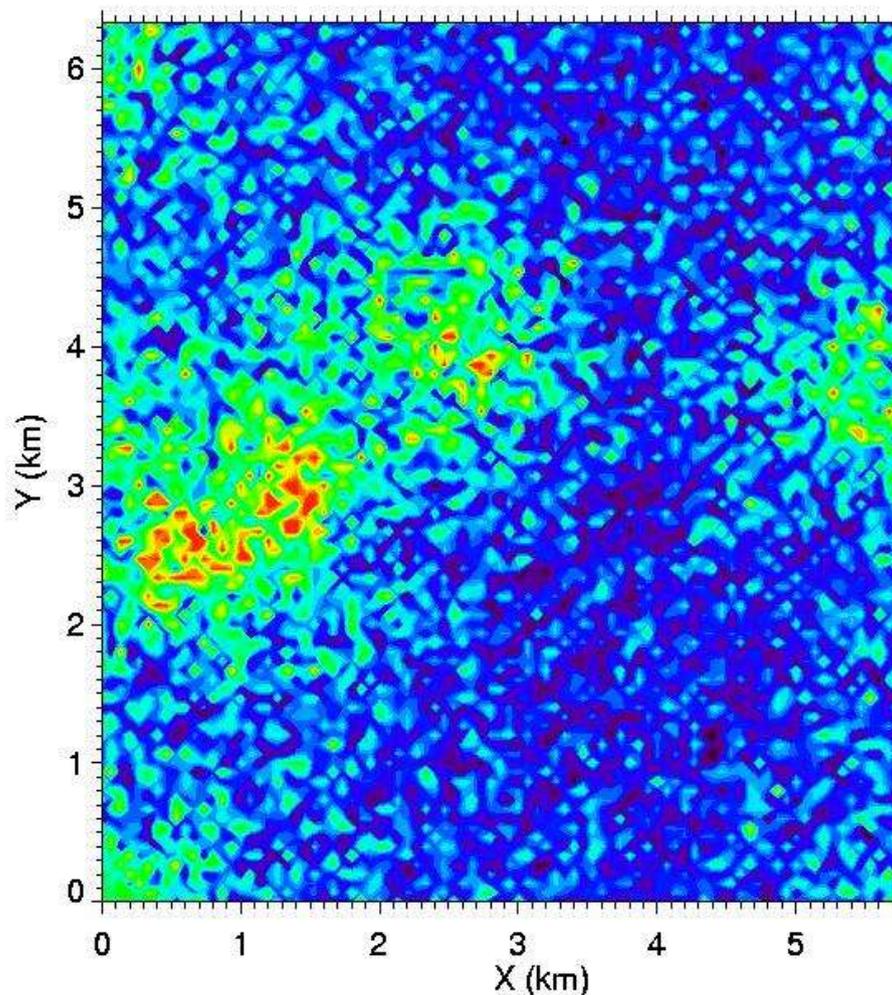
- Preliminary comparison of tradeoff between I3RC MC and SHDOM
 - First, SHDOM has limited zone of applicability: grid cells must have optical depth ~ 1 or less; less than 10^6 grid points.
 - SHDOM runs into memory limits long before CPU time limits.
 - Difficult to achieve very high accuracy with SHDOM in 3D
 - SHDOM is much faster for pixel radiances, only somewhat faster for fluxes compared to MC.
 - MC is superior for any horizontally domain averaged quantity.
 - SHDOM – MC tradeoff depends strongly on medium (not on λ)

Future Comparison Work

- Include other atmospheric components?
- Use version of MC model that interpolates between grid points like SHDOM does?
- Compare photon-tracing with maximal cross section for clouds
- Improve I3RC MC radiance computation speed
- Optimize SHDOM parameters for comparisons
- Add spectrally broadband grid point heating rates to comparison

Upwelling Flux for Cumulus Field

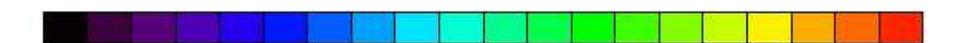
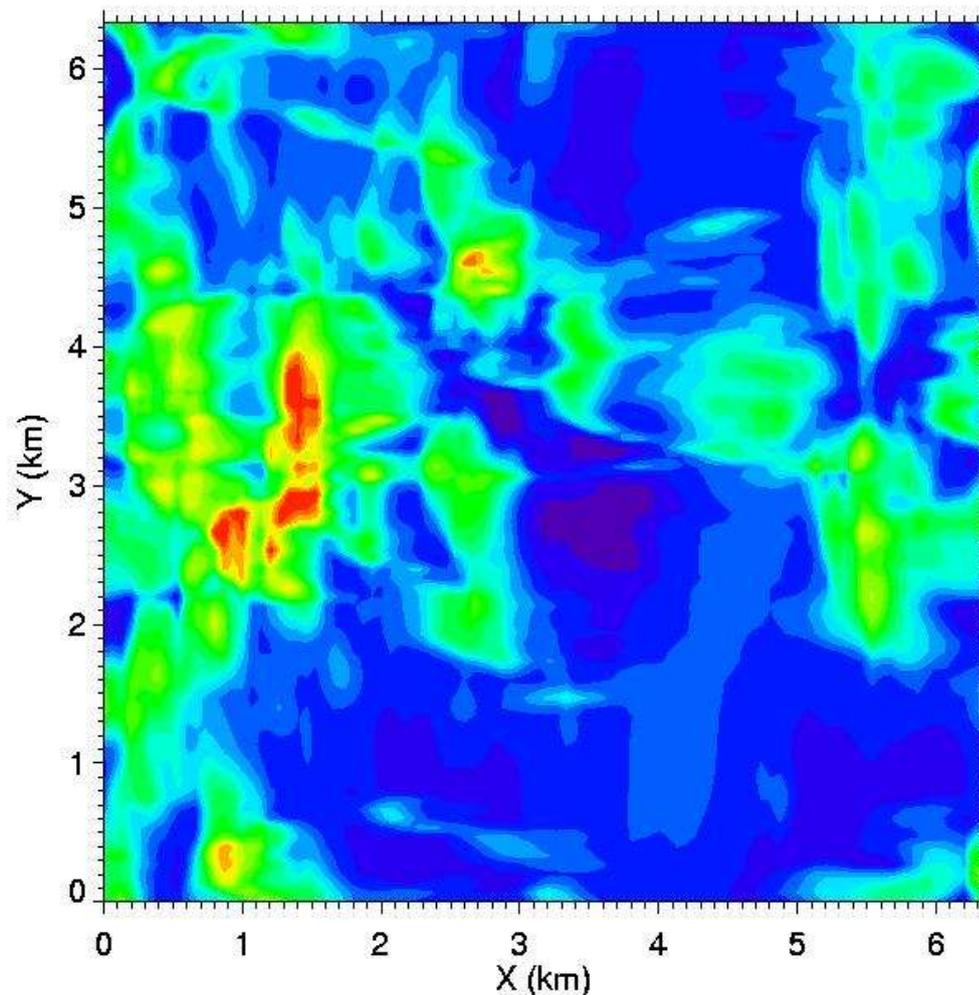
I3RC MC: $N_{\text{phot}}=10^5$



0.00 0.20 0.40 0.60

Upwelling Flux at $Z=2.50$ km

SHDOM: $N_{\mu}=2$ splitacc=0.10

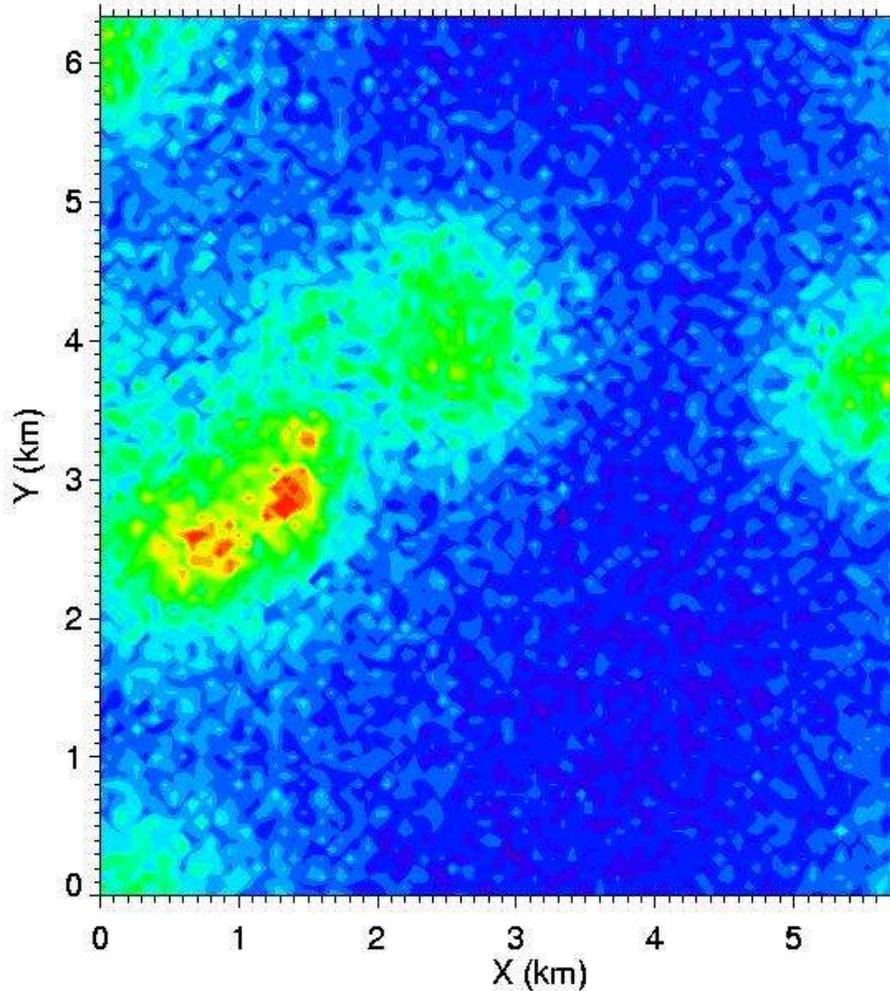


0.00 0.20 0.40 0.60 0.80

Upwelling Flux at $Z=2.50$ km

Upwelling Flux for Cumulus Field

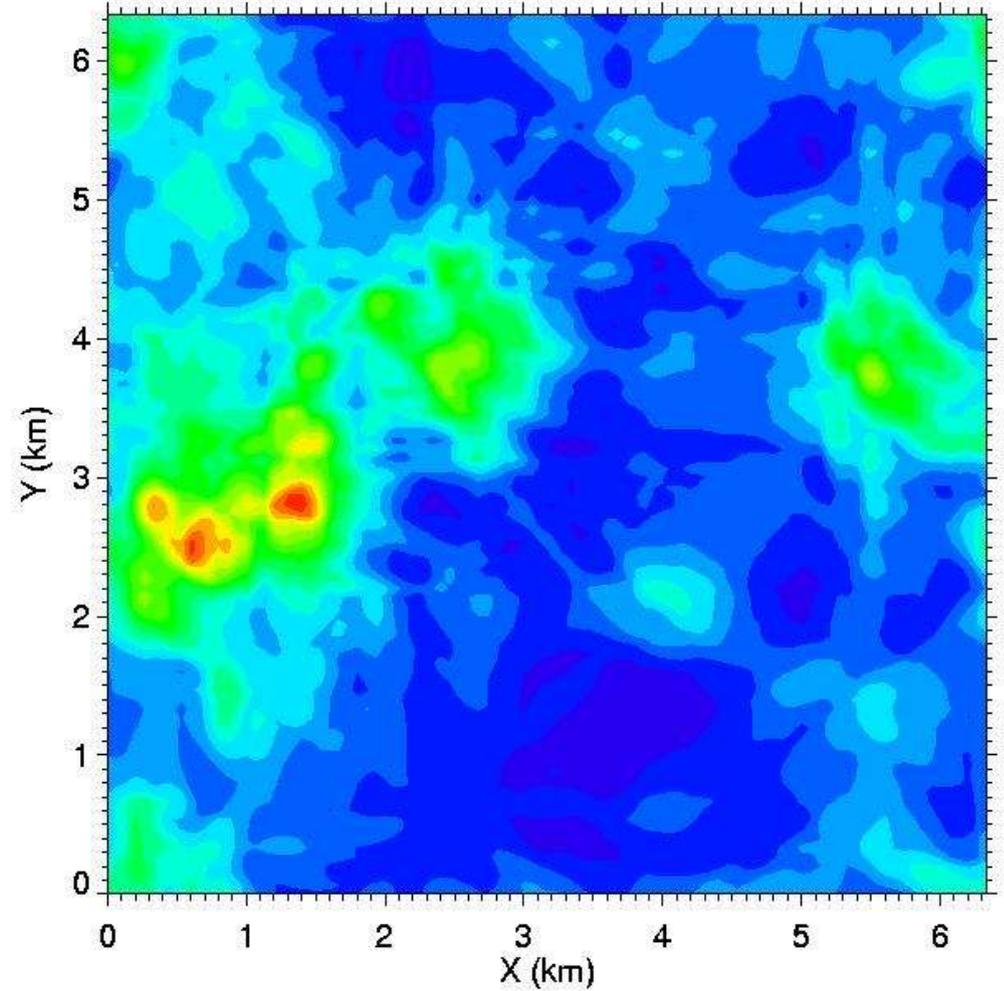
I3RC MC: $N_{\text{phot}}=10^6$



0.00 0.20 0.40 0.60

Upwelling Flux at $Z=2.50$ km

SHDOM: $N_{\mu}=4$ splitacc=0.10

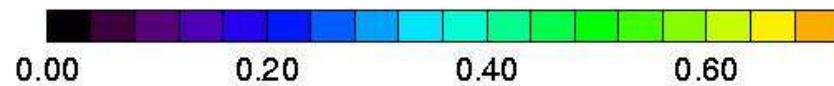
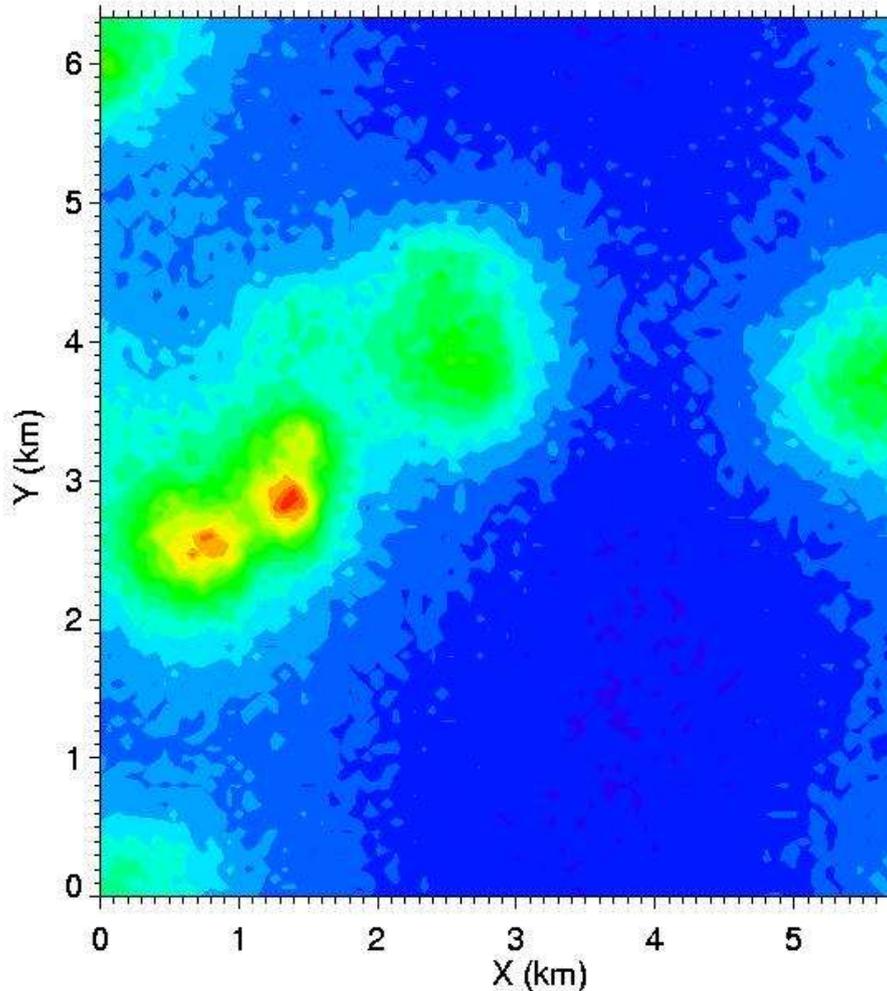


0.00 0.20 0.40 0.60 0.80

Upwelling Flux at $Z=2.50$ km

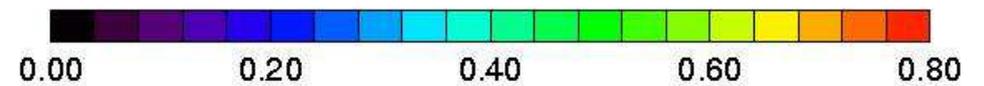
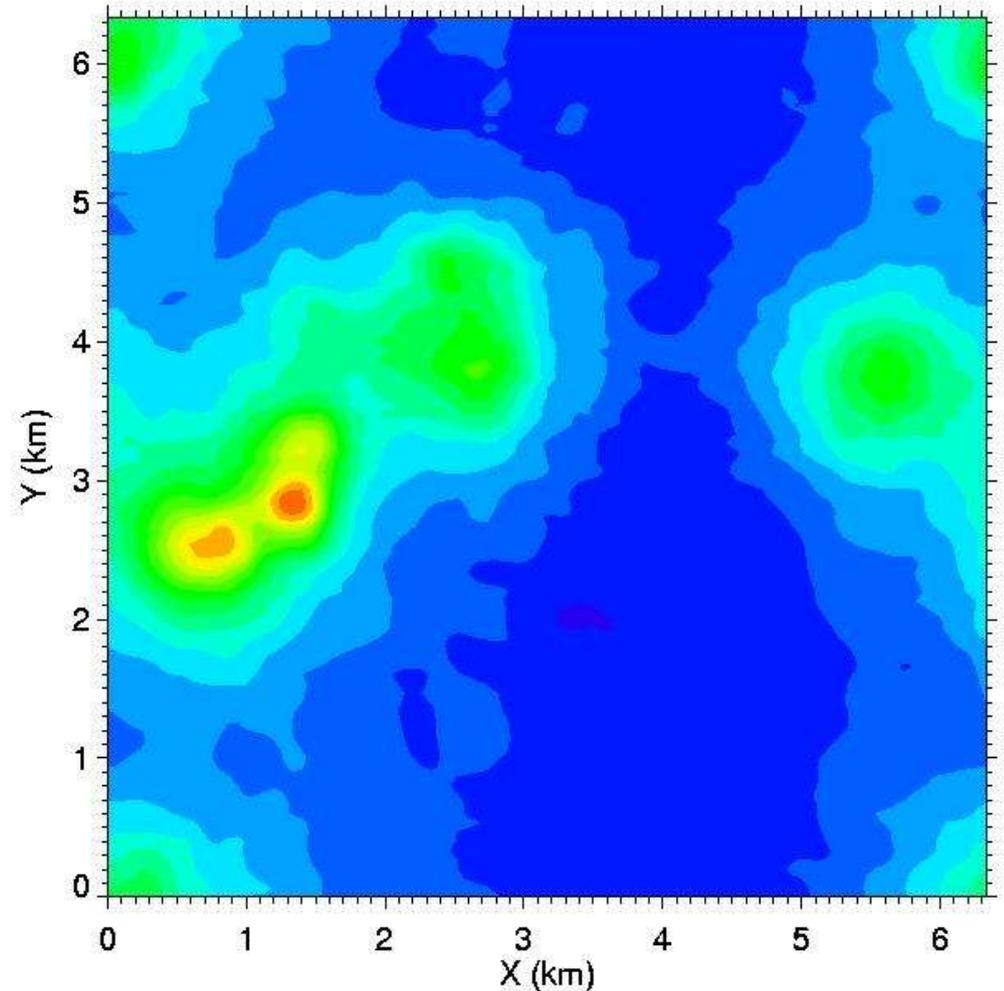
Upwelling Flux for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^7$



Upwelling Flux at Z=2.50 km

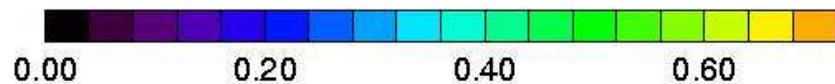
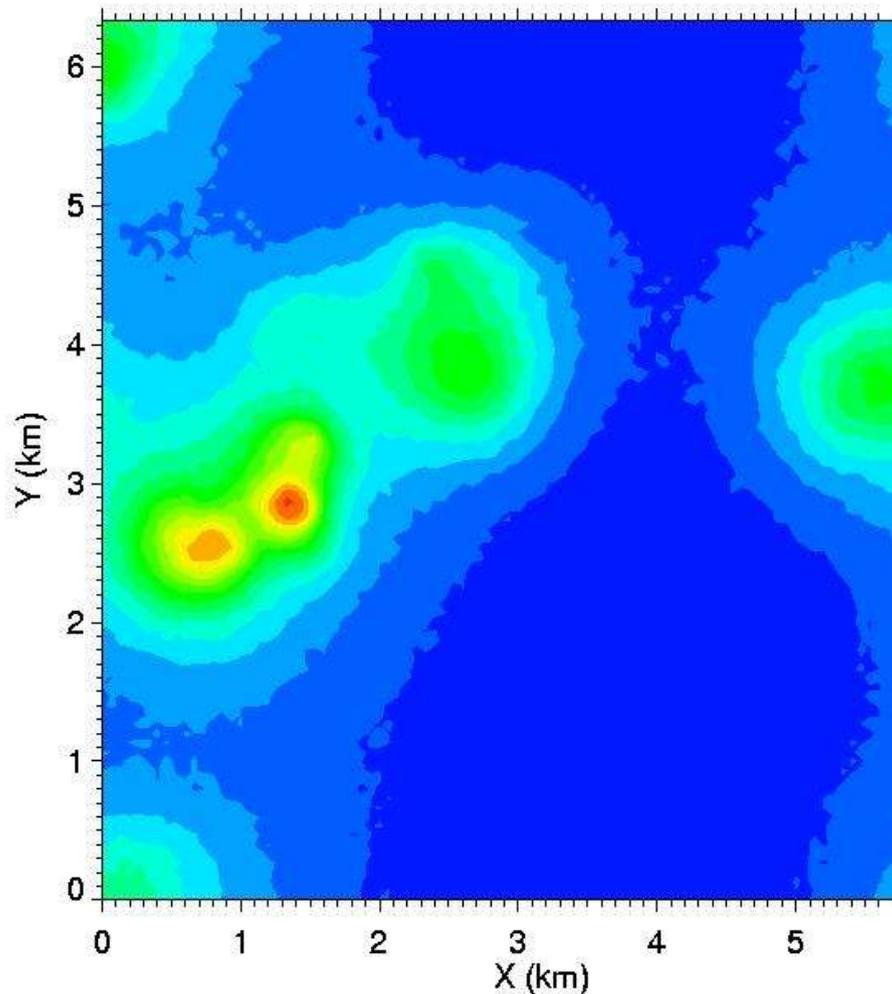
SHDOM: $N_{\mu}=8$ splitacc=0.05



Upwelling Flux at Z=2.50 km

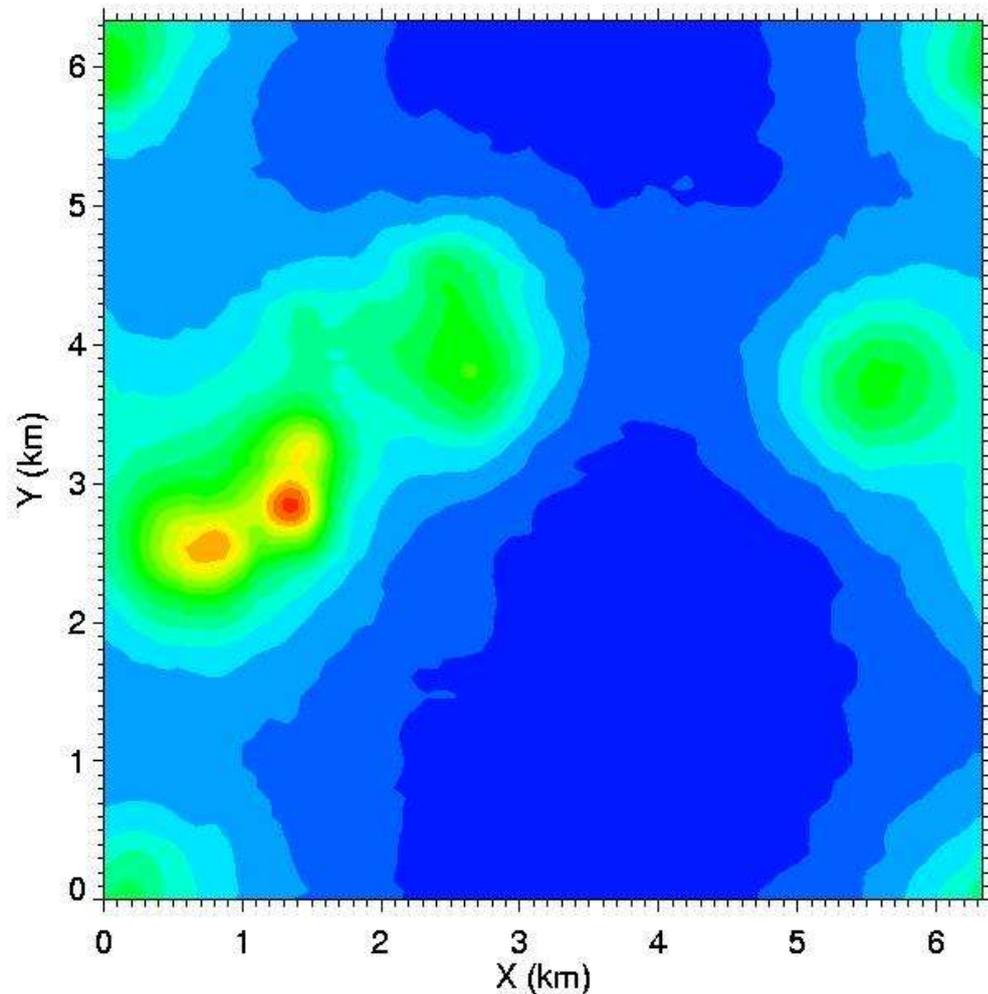
Upwelling Flux for Cumulus Field

I3RC MC: $N_{\text{phot}}=10^8$



Upwelling Flux at $Z=2.50$ km

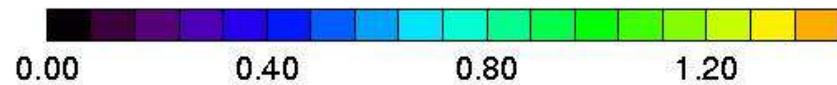
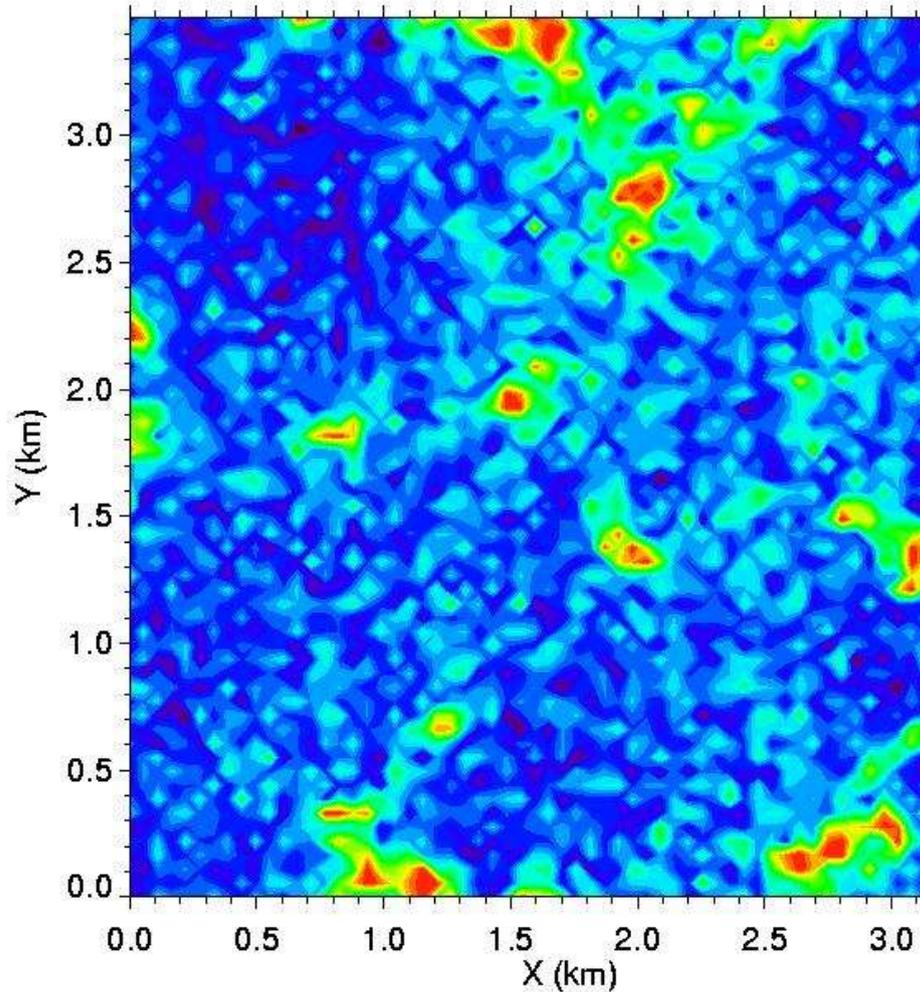
SHDOM: $N_{\mu}=12$ splitacc=0.03



Upwelling Flux at $Z=2.50$ km

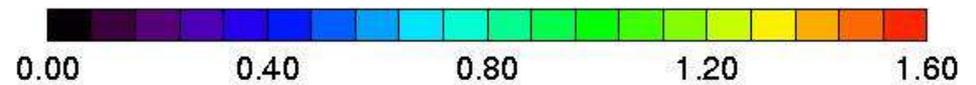
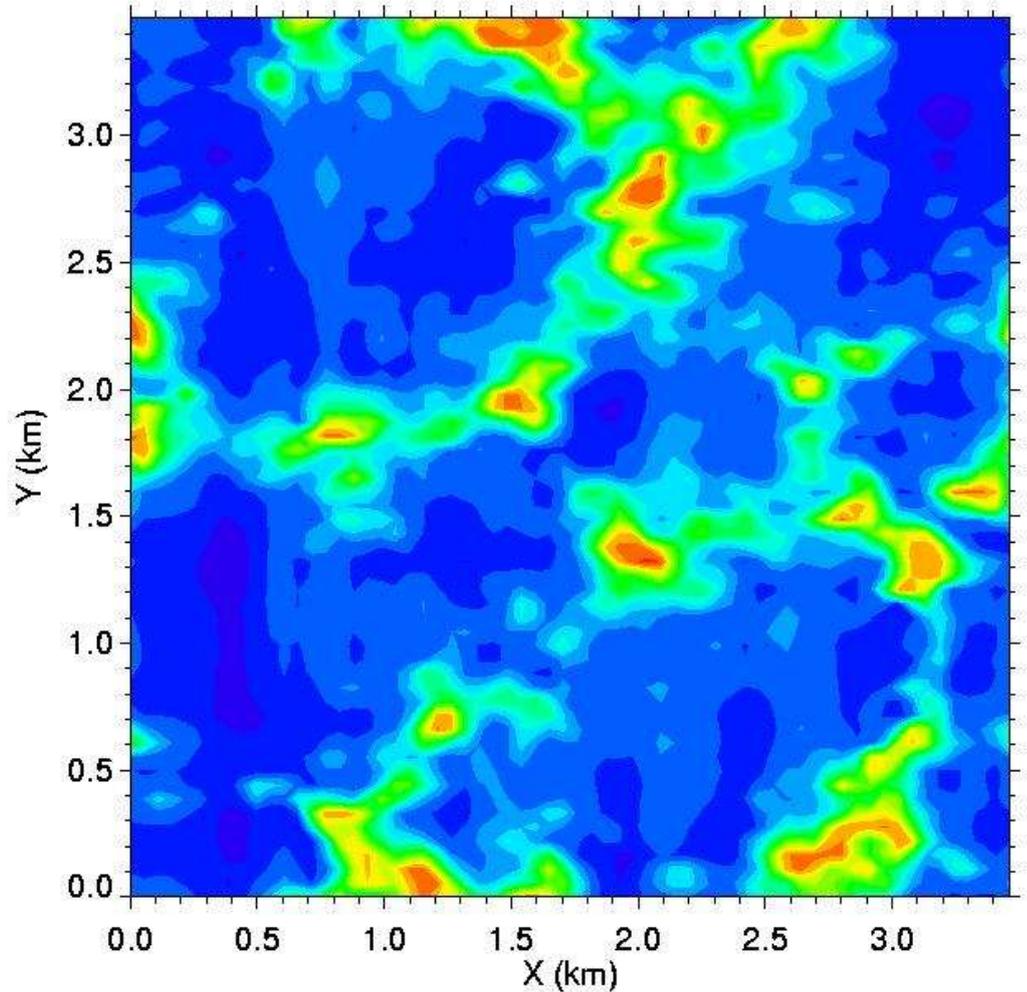
Downwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^5$



Downwelling Flux at Z=0.00 km

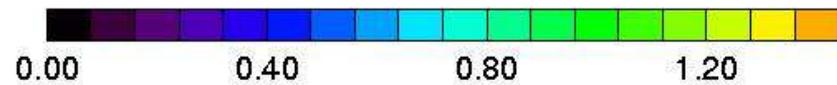
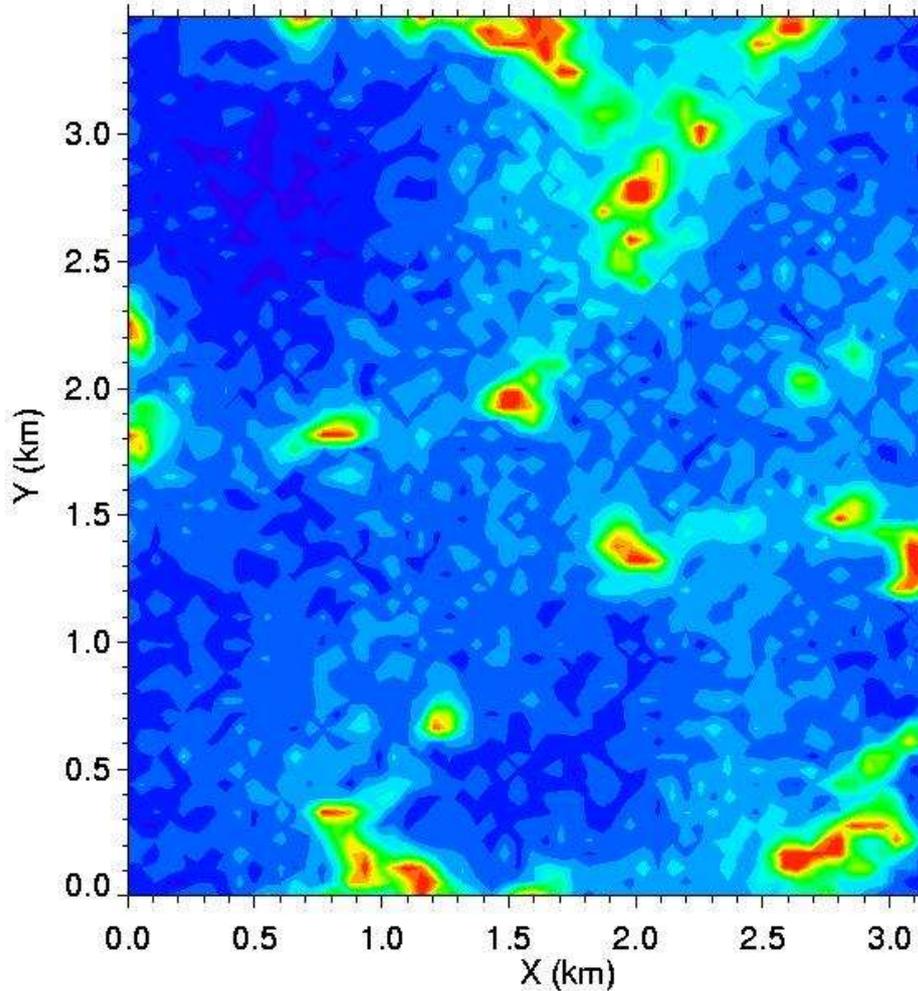
SHDOM: $N_{\mu}=2$ splitacc=0.10



Downwelling Flux at Z=0.00 km

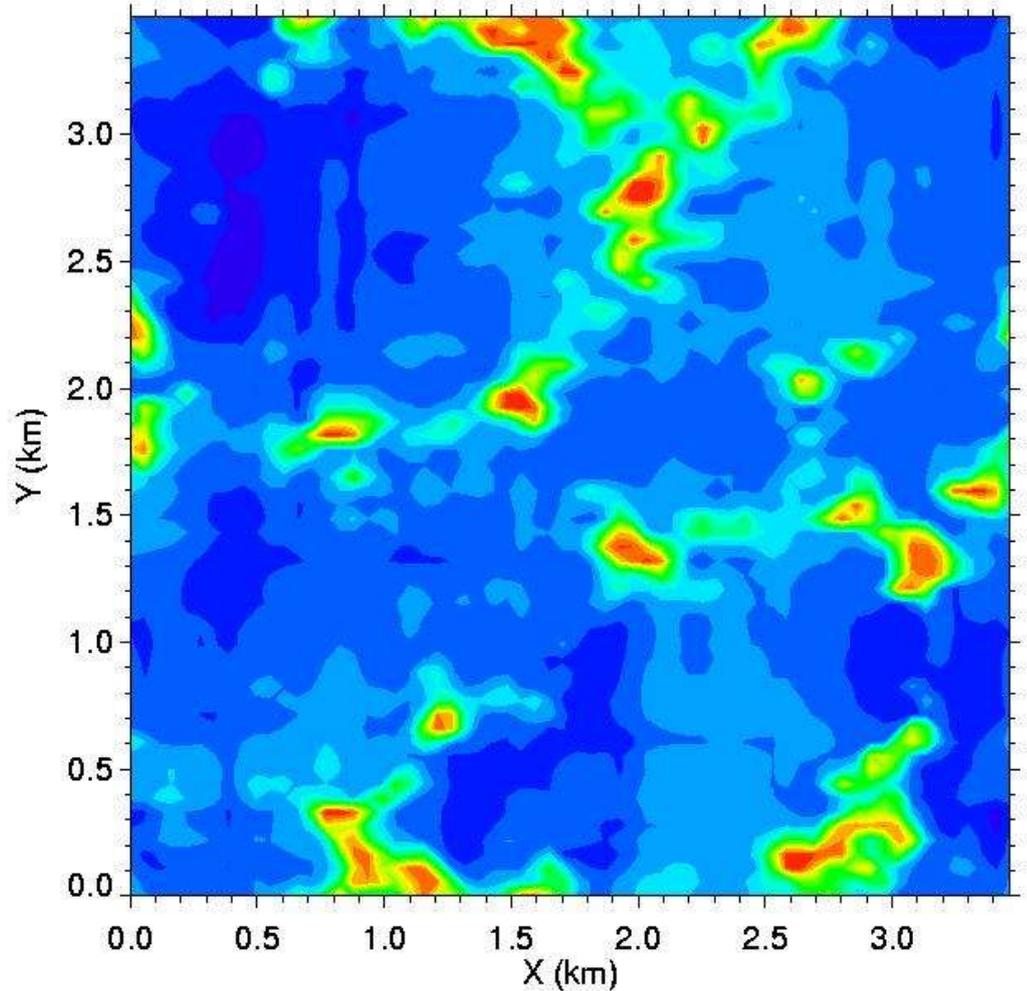
Downwelling Flux for Stratocumulus Field

I3RC MC: $N_{\text{phot}}=10^6$



Downwelling Flux at Z=0.00 km

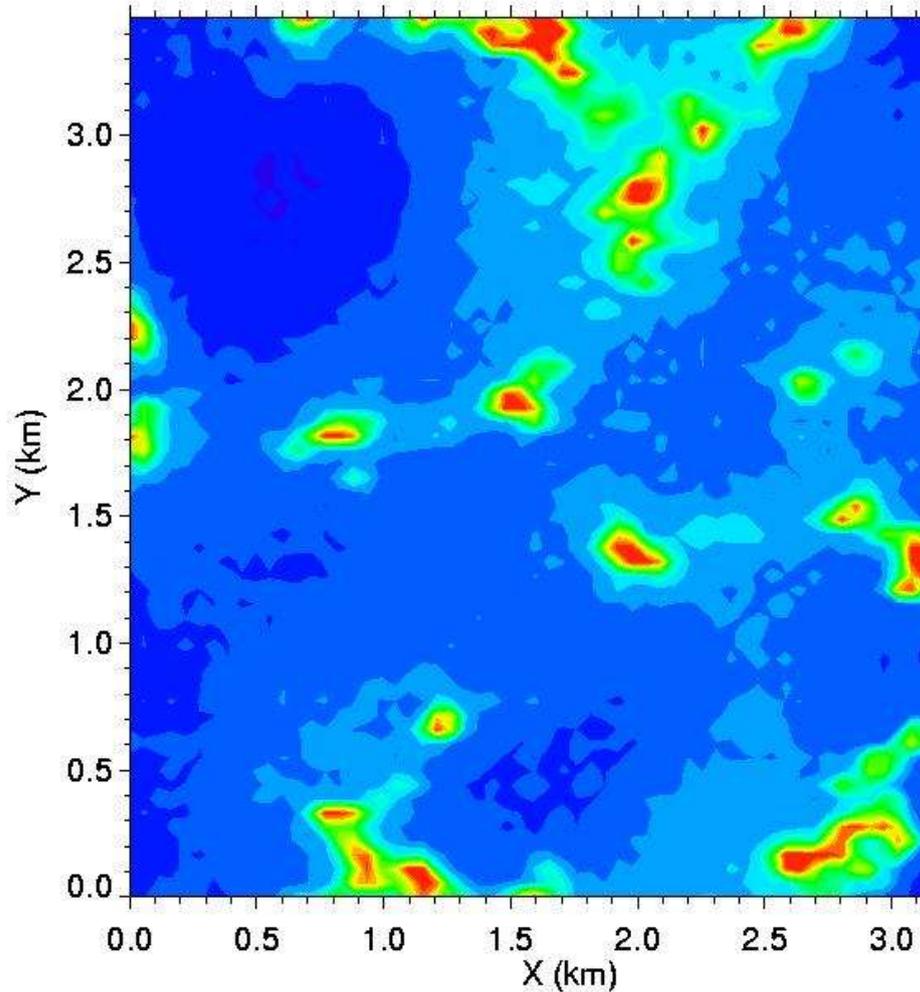
SHDOM: $N_{\mu}=4$ splitacc=0.10



Downwelling Flux at Z=0.00 km

Downwelling Flux for Stratocumulus Field

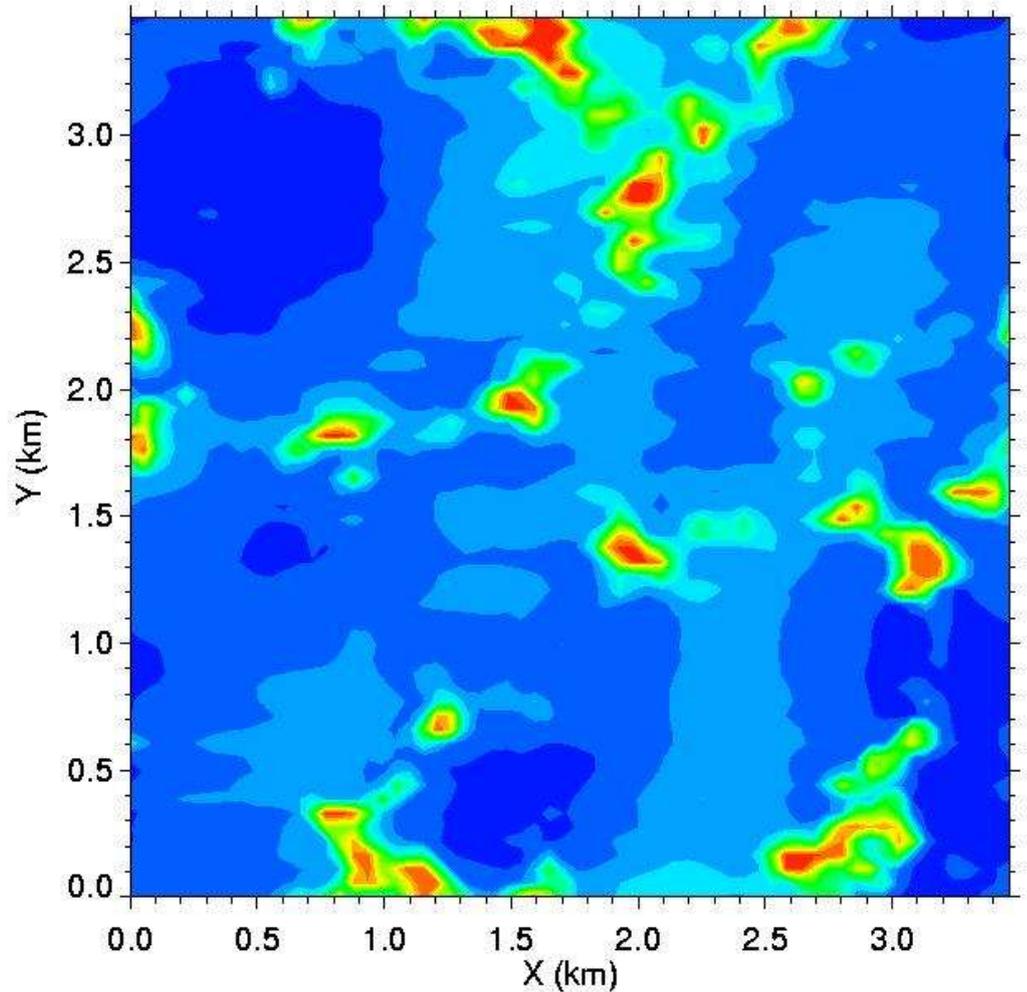
I3RC MC: $N_{\text{phot}}=10^7$



0.00 0.40 0.80 1.20

Downwelling Flux at Z=0.00 km

SHDOM: $N_{\mu}=8$ splitacc=0.05

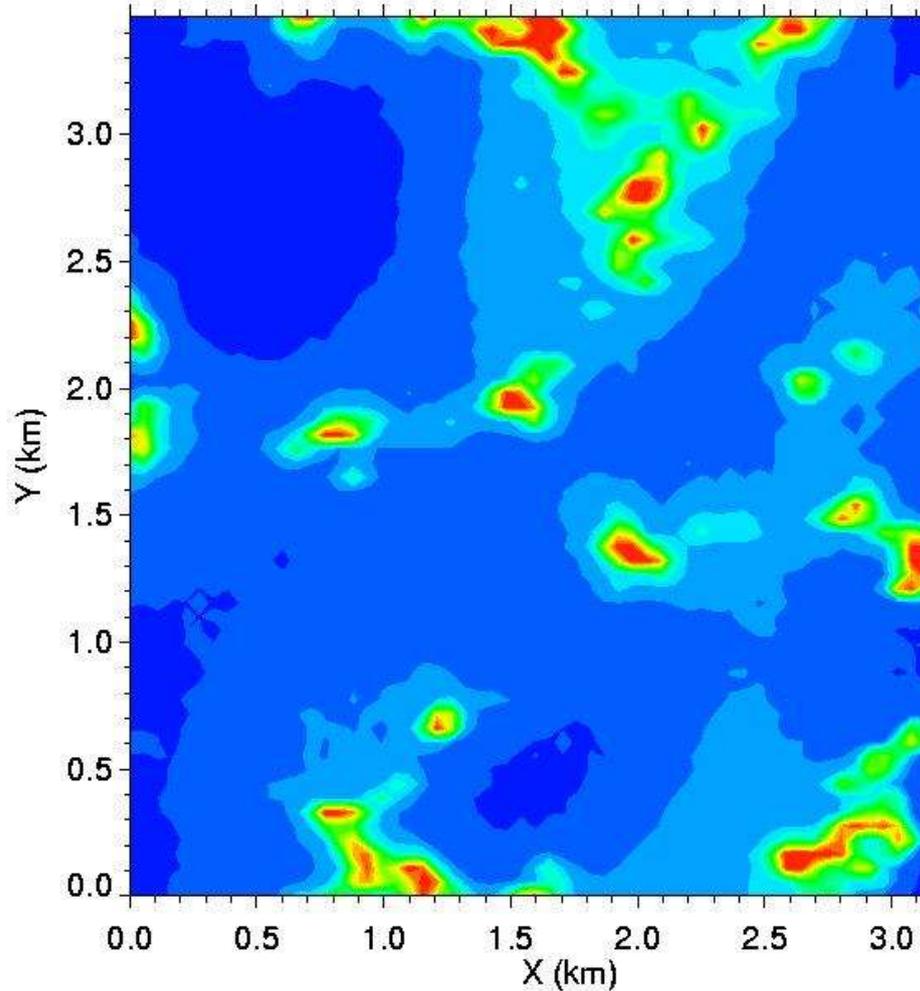


0.00 0.40 0.80 1.20 1.60

Downwelling Flux at Z=0.00 km

Downwelling Flux for Stratocumulus Field

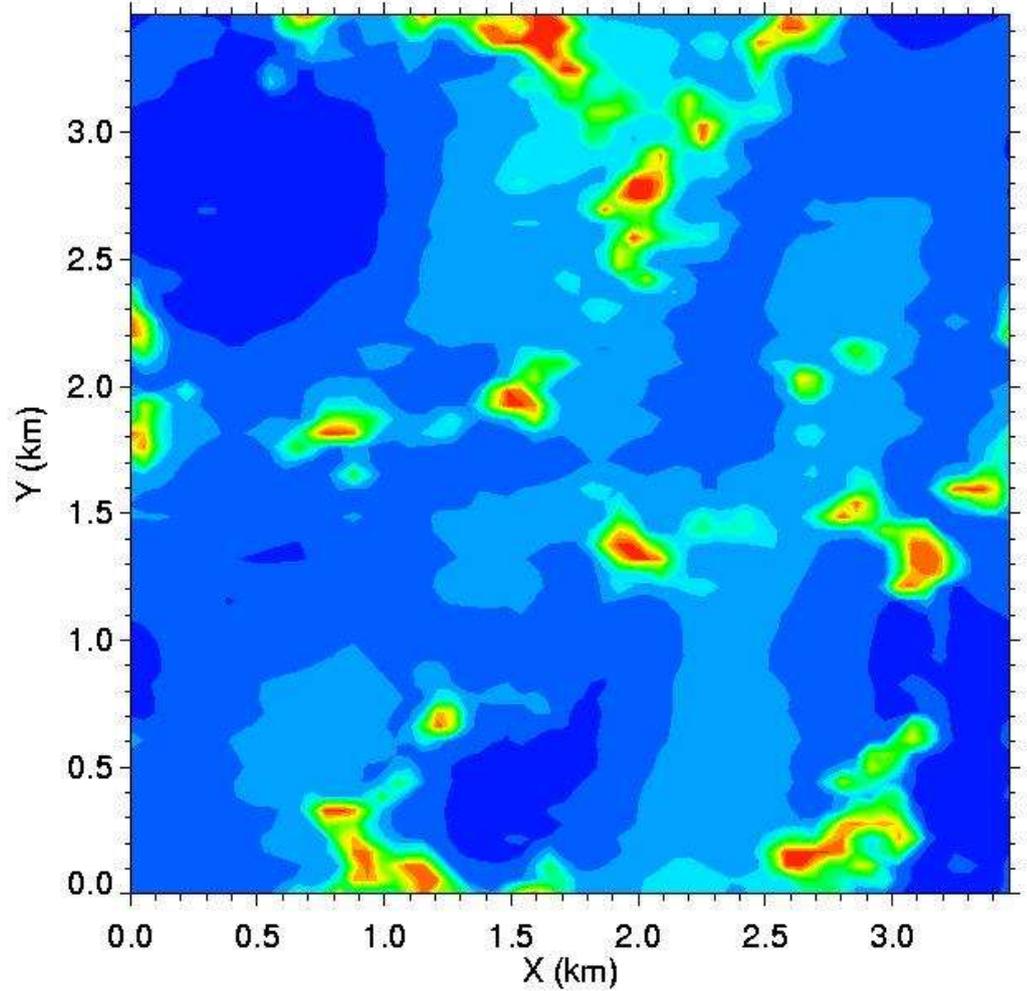
I3RC MC: $N_{\text{phot}}=10^8$



0.00 0.40 0.80 1.20

Downwelling Flux at Z=0.00 km

SHDOM: $N_{\mu}=12$ splitacc=0.03



0.00 0.40 0.80 1.20 1.60

Downwelling Flux at Z=0.00 km