SASKTRAN

A spherical radiative transfer tool
for the limb community.

Nick Lloyd, Doug Degenstein, Adam Bourassa

University of Saskatchewan

Motivation

Spherical radiative transfer models are a key component of the limb scatter technique. They are essential for accurate for forward modeling in various inversion schemes.

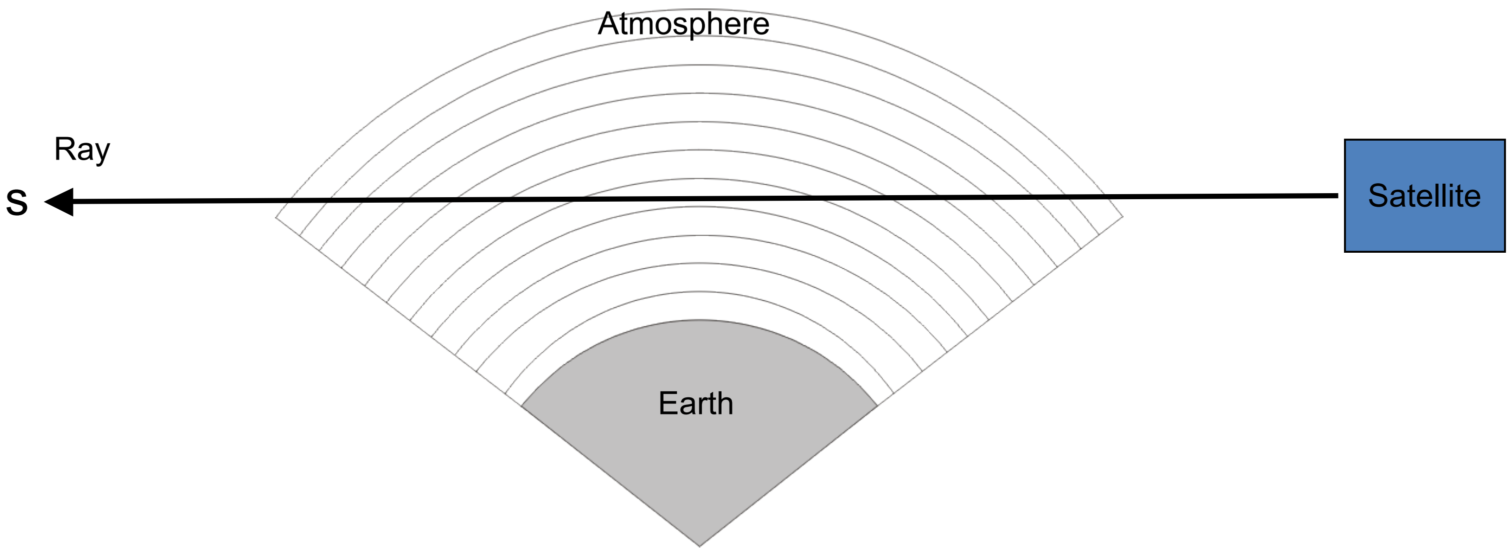
Several radiative transfer models have been developed by various groups in the limb community but we are still at a point where no single model satisfies everyone.

SASKTRAN is the radiative transfer model developed at the University of Saskatchewan as part of the Odin-OSIRIS operations and data processing.

This is our story …..

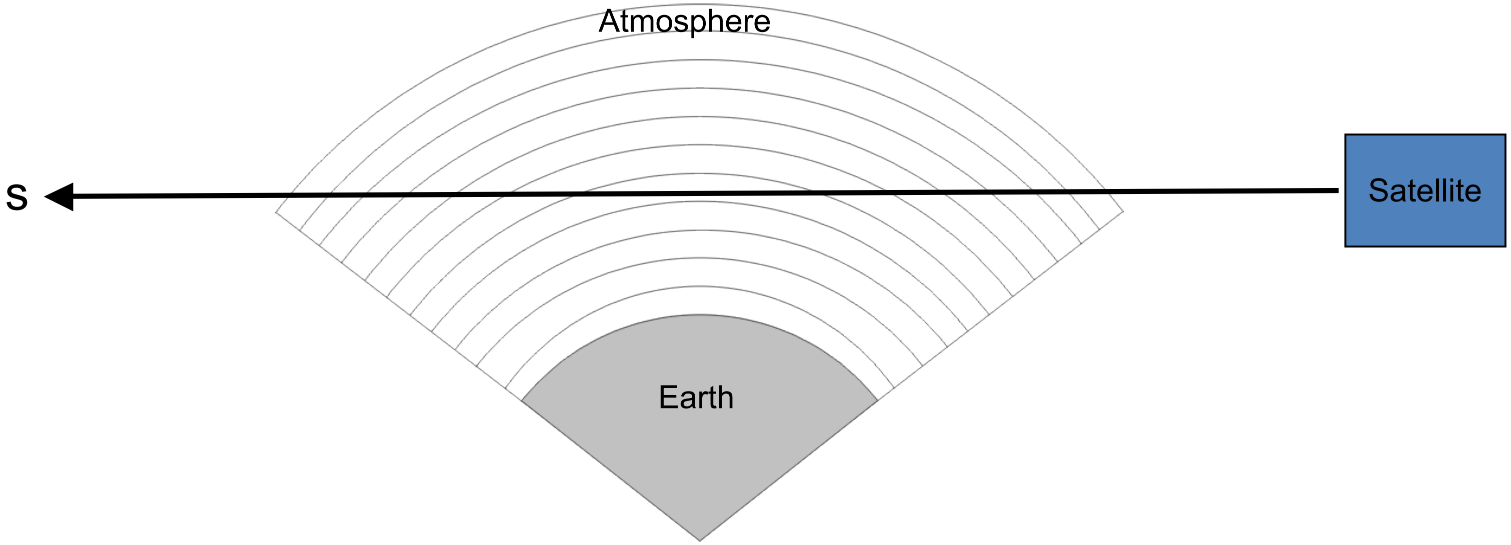
Radiative Transfer

Problem: Calculate the radiance seen by a satellite looking through the atmosphere.



Radiative Transfer

Use the equation of radiative transfer.

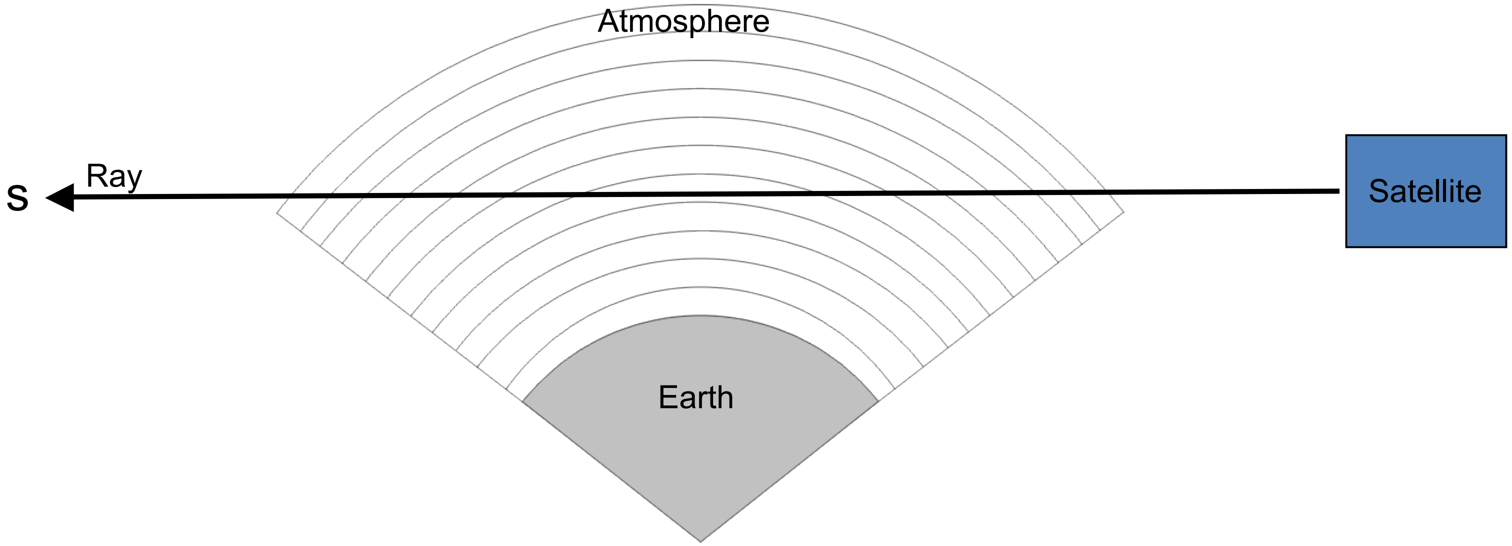


$$I = I_\infty e^{-\tau_\infty} + \int_0^\infty J(s)\, e^{-\tau(s)} ds$$

A three letter problem (I, J, τ). How hard can that be!

Radiative Transfer

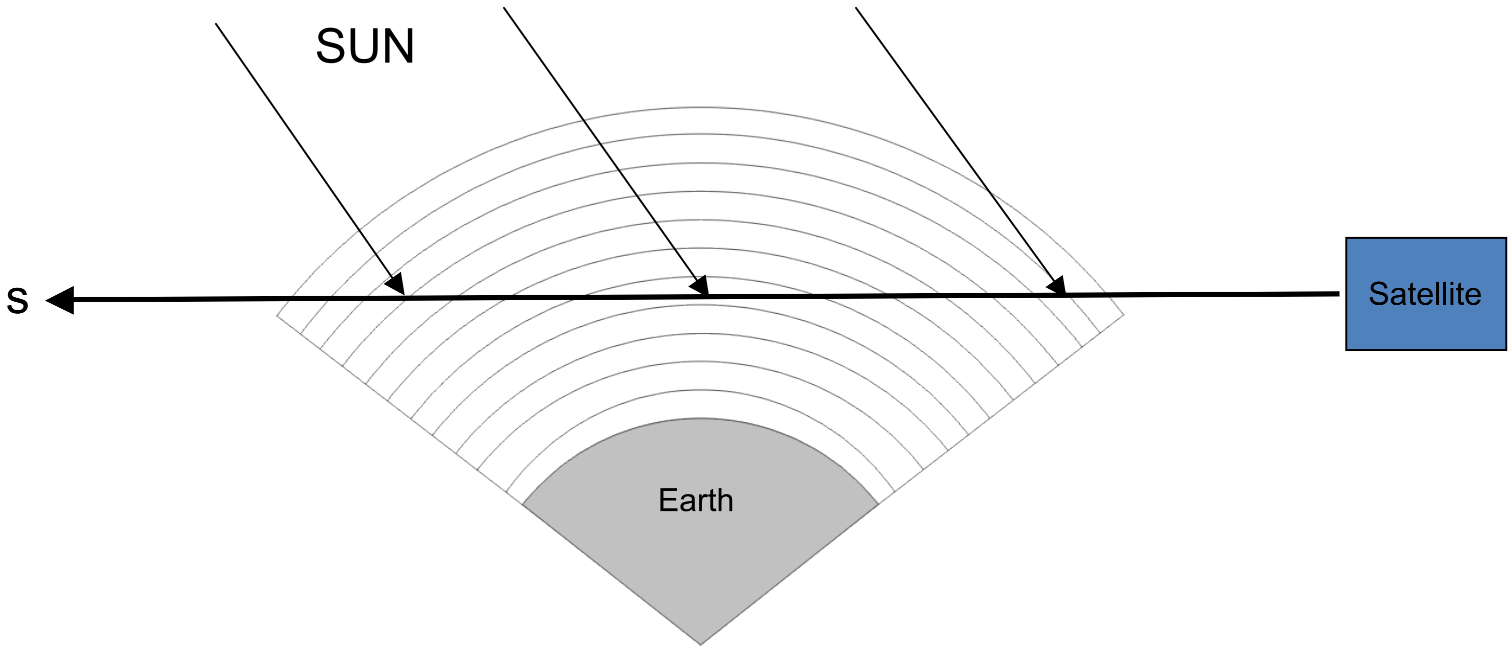
The problem reduces to finding J(s) and $\tau(s)$ and performing the integral.



$$I = I_\infty e^{-\tau_\infty} + \int_0^\infty J(s)\, e^{-\tau(s)} ds$$

Radiative Transfer

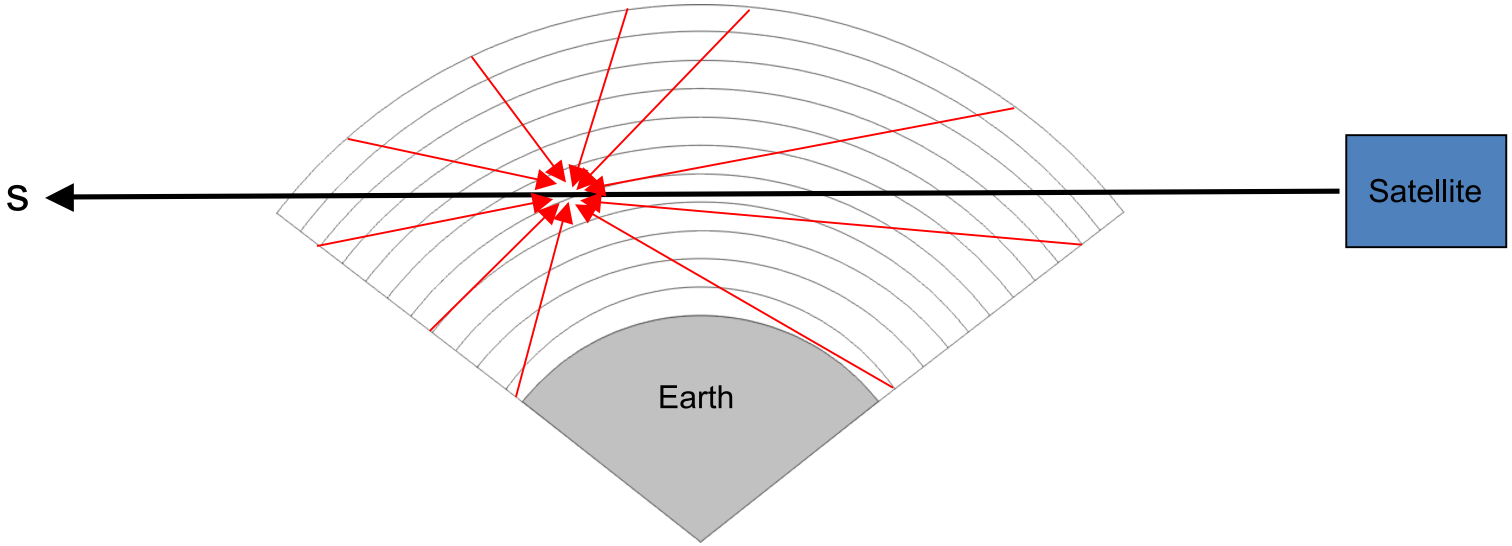
J(s) has some contribution from single scatter.



$$I = I_{\infty} e^{-\tau_{\infty}} + \int_{0}^{\infty} J(s)\, e^{-\tau(s)} ds$$

Radiative Transfer

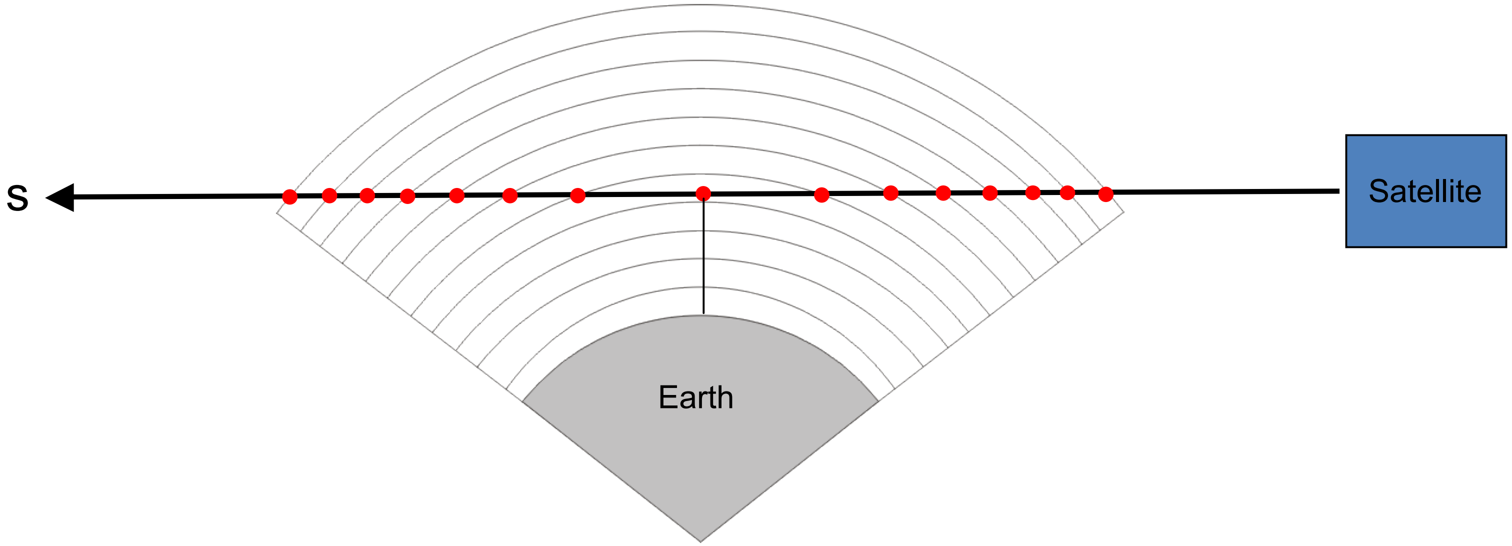
J(s) has some contribution from diffuse, multiple scatter.



$$I = I_{\infty} e^{-\tau_{\infty}} + \int_{0}^{\infty} J(s)\, e^{-\tau(s)} ds$$

Radiative Transfer

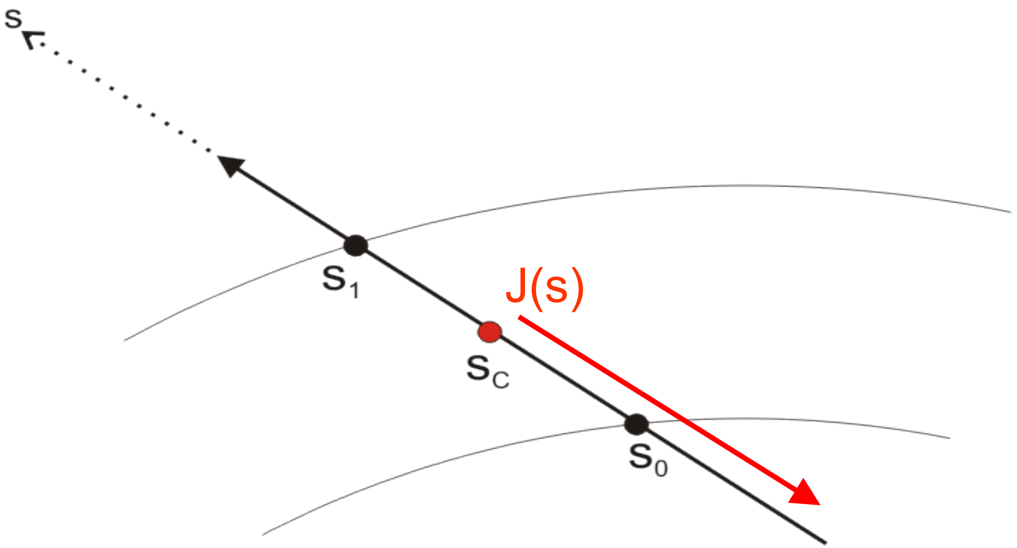
The integral can be broken into discrete sections without any loss of accuracy.



$$I = I_\infty e^{-\tau_\infty} + \sum_i \int_{s_i}^{s_{i+1}} J(s)\, e^{-\tau(s)} ds$$

Radiative Transfer

The integral across each ray segment can be approximated using a quadrature technique.



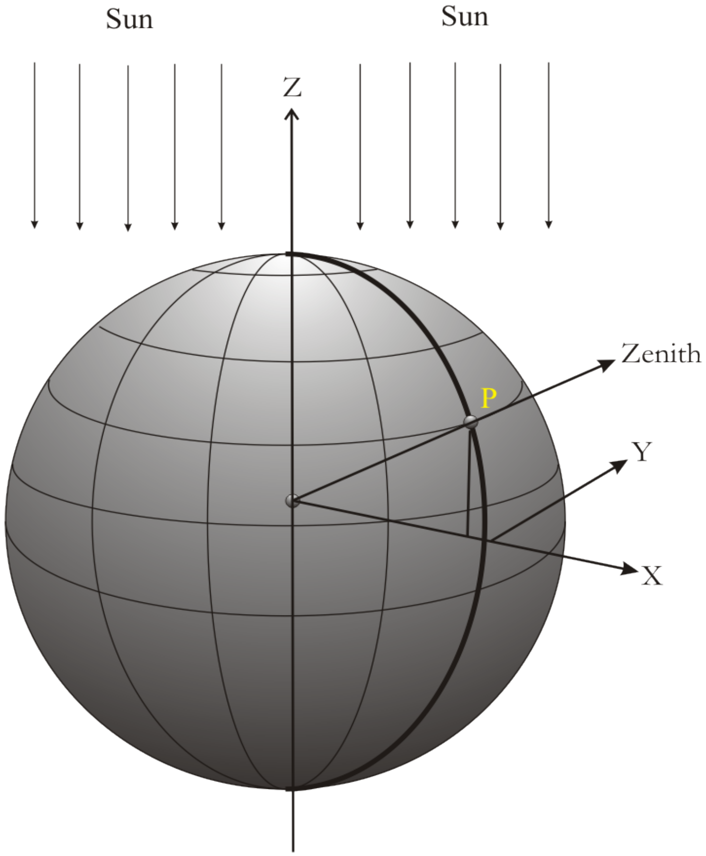
$$\int_{s_i}^{s_{i+1}} J(s)\, e^{-\tau(s)} ds \approx \sum_j q_j J(s_j)\, e^{-\tau(s_j)}$$

We need the single scatter contribution to J(s) at any point in the atmosphere.

so …..

Radiative Transfer

Place Sun on Z axis. Colatitude is now solar zenith angle. To first order, solutions are independent of solar longitude.

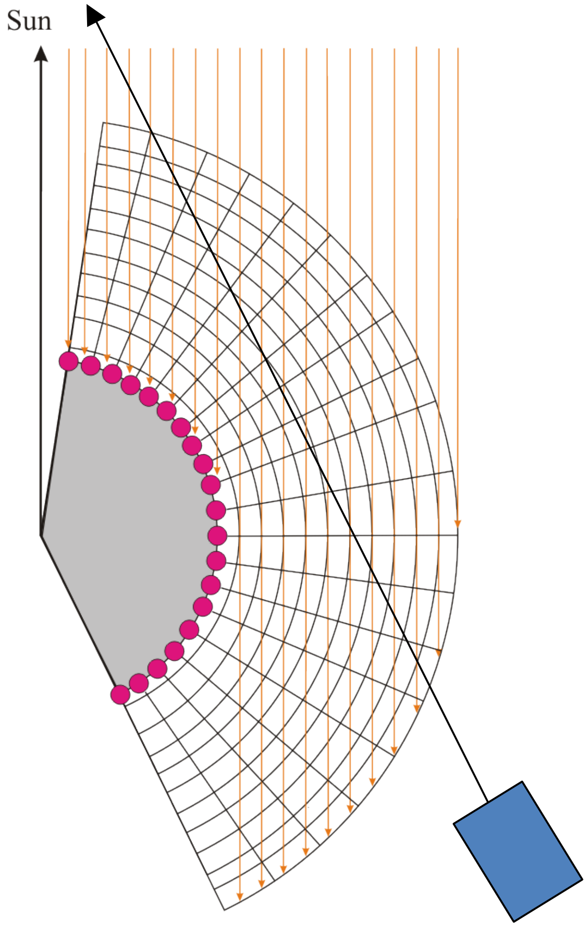


Single Scatter Terms: Solar Transmission Table

Single scatter source terms are quickly evaluated by interpolating the optical depth of the Sun stored in the solar transmission table.

$$J_{SS}(s) = \sum_i w_i S_i$$

2-D table is a function of height and solar zenith angle.



We need the multiple scatter contribution to J(s) at every point in the atmosphere.

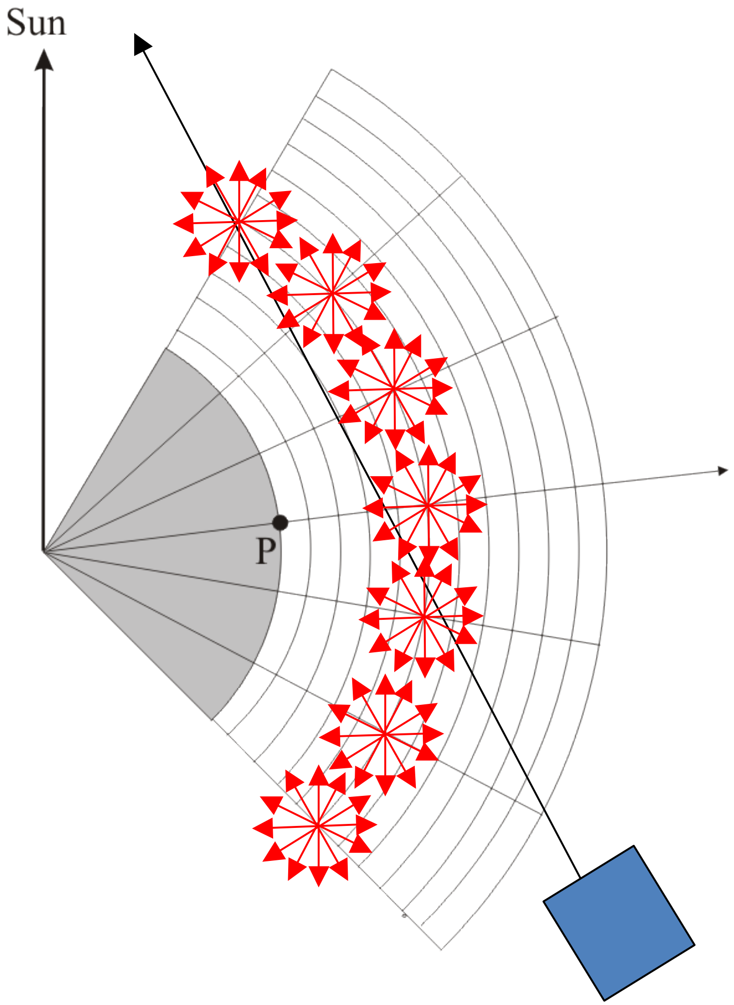
so …..

Multiple Scatter Terms: Diffuse Points Table

Diffuse source functions terms are quickly evaluated by interpolating the outbound diffuse source functions stored at each point in the Diffuse Points Table

$$J_{MS}(s) = \sum_j w_j D_j$$

Linearly interpolate in height and solar zenith angle, up to 4 points. Three point vertex interpolation on each outbound unit sphere. Thus "J" can have as many as 12 components.



Radiative Transfer

The equation of radiative transfer

$$I = I_\infty e^{-\tau_\infty} + \int_0^\infty J(s)\, e^{-\tau(s)} ds$$

becomes

$$I = I_{SS} + I_{MS}$$

The total radiance at a point is the sum of a single scatter term and a multiple scatter term

where,

$$I_{SS} = \sum_k w_k S(k)$$

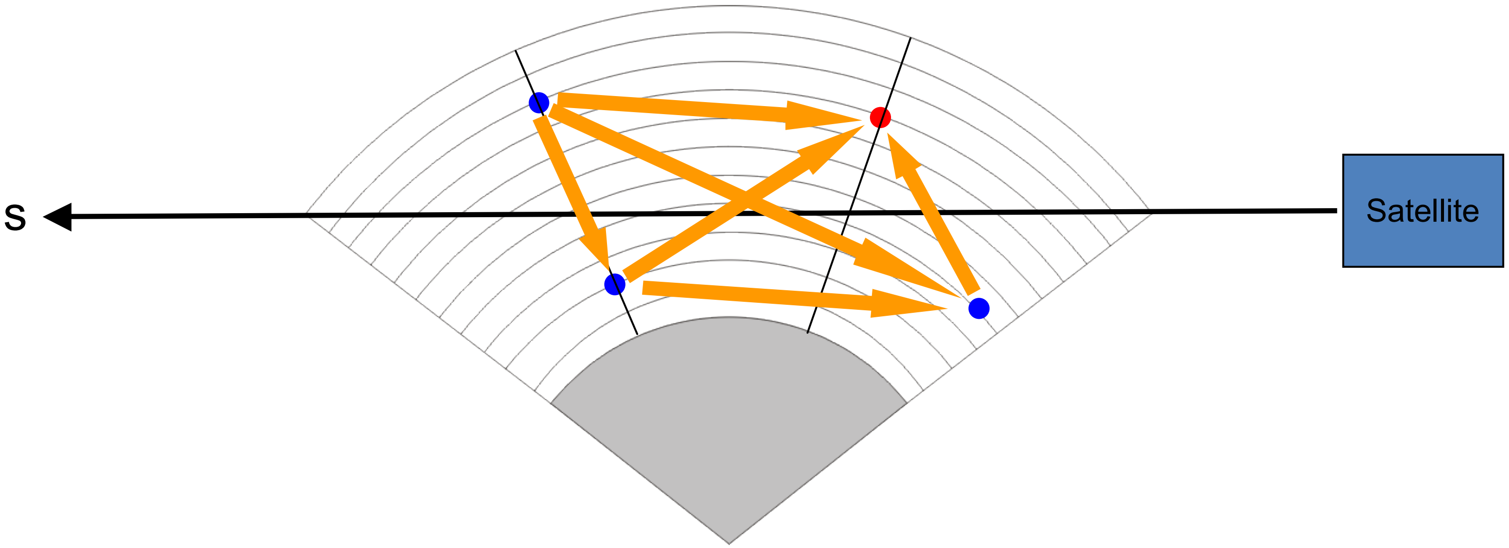
The weighted sum of all the interpolated single scatter source terms that contribute to all of the quadrature points on all of the segments along the ray.

$$I_{MS} = \sum_j w_j D(j)$$

The weighted sum of all the diffuse source terms that contribute to all of the quadrature points on all of the segments along the ray.

Successive Orders Algorithm

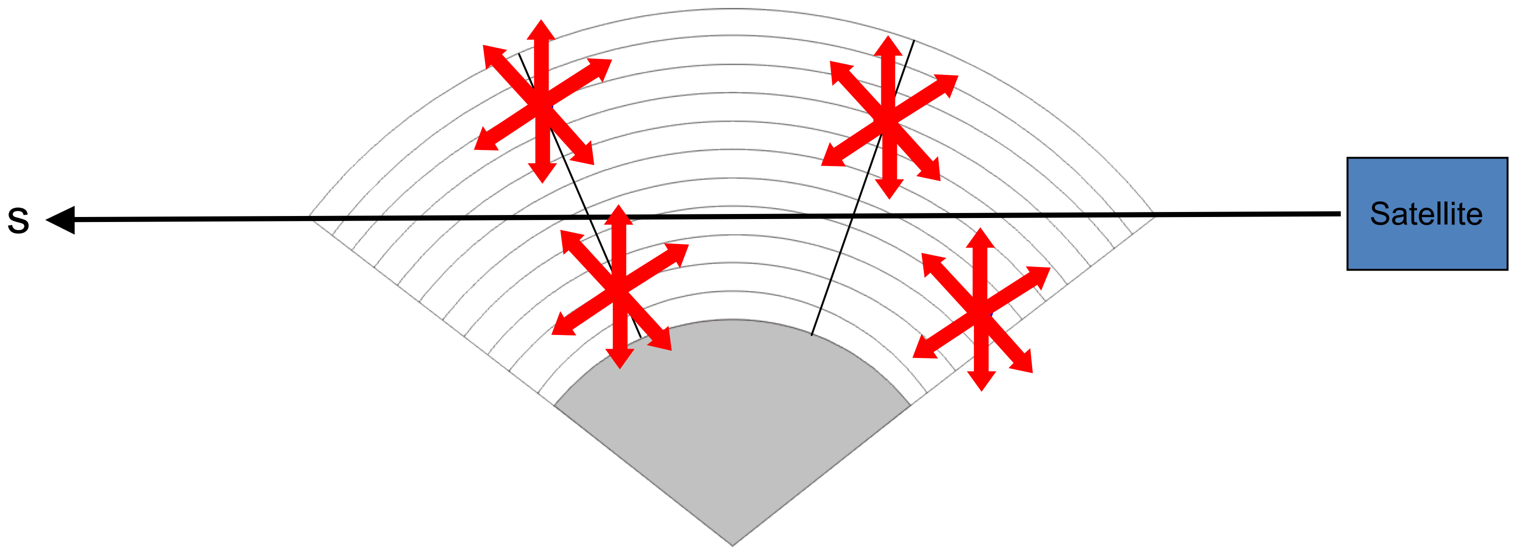
Step 1. Calculate incoming radiance at all the diffuse points.



Use the outbound source functions at each diffuse point to calculate the incoming radiances at other points

Successive Orders Algorithm

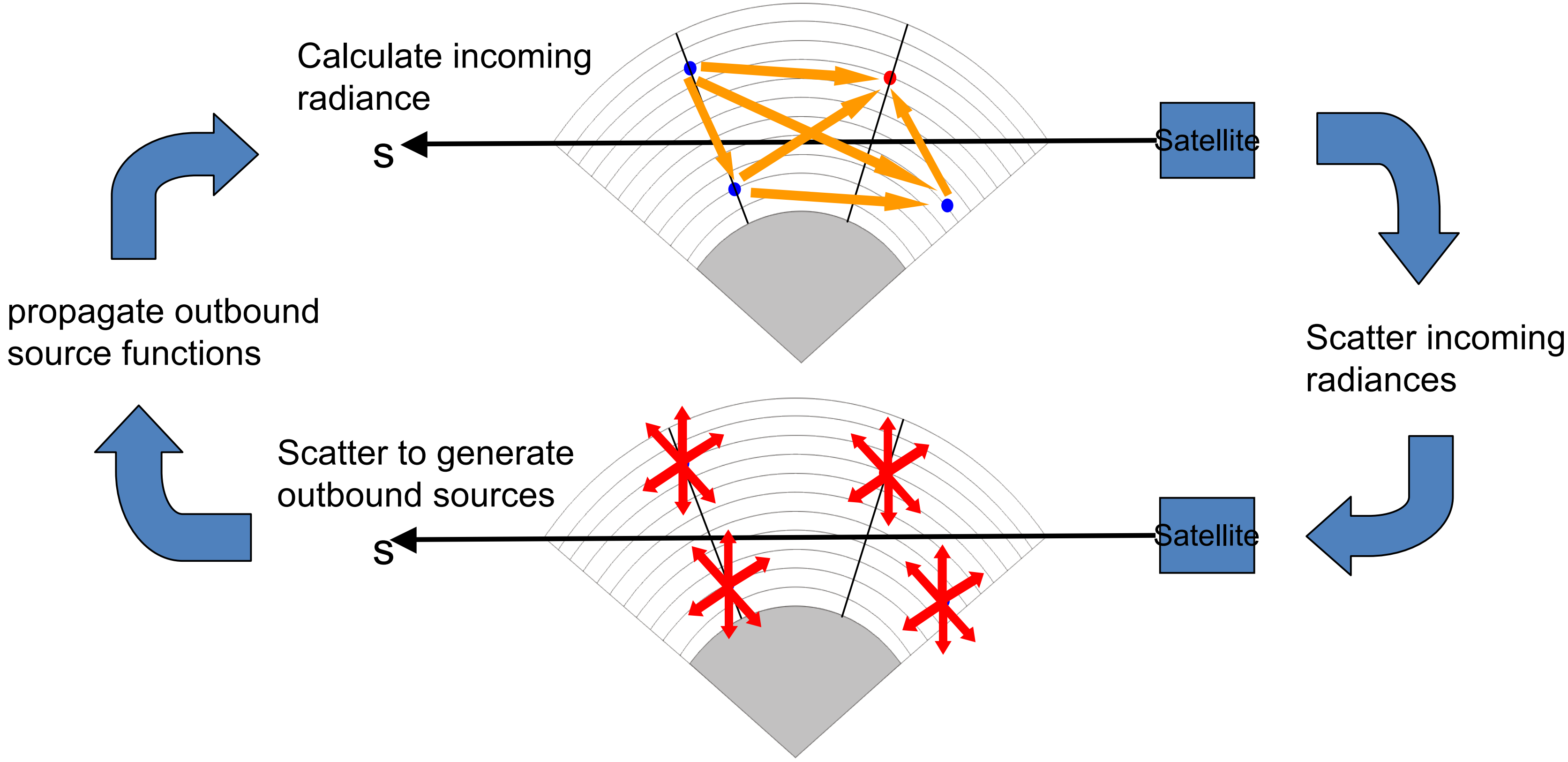
Step 2. Calculate outbound source functions at all diffuse points



Scatter the incoming radiances to generate next order of outbound source functions

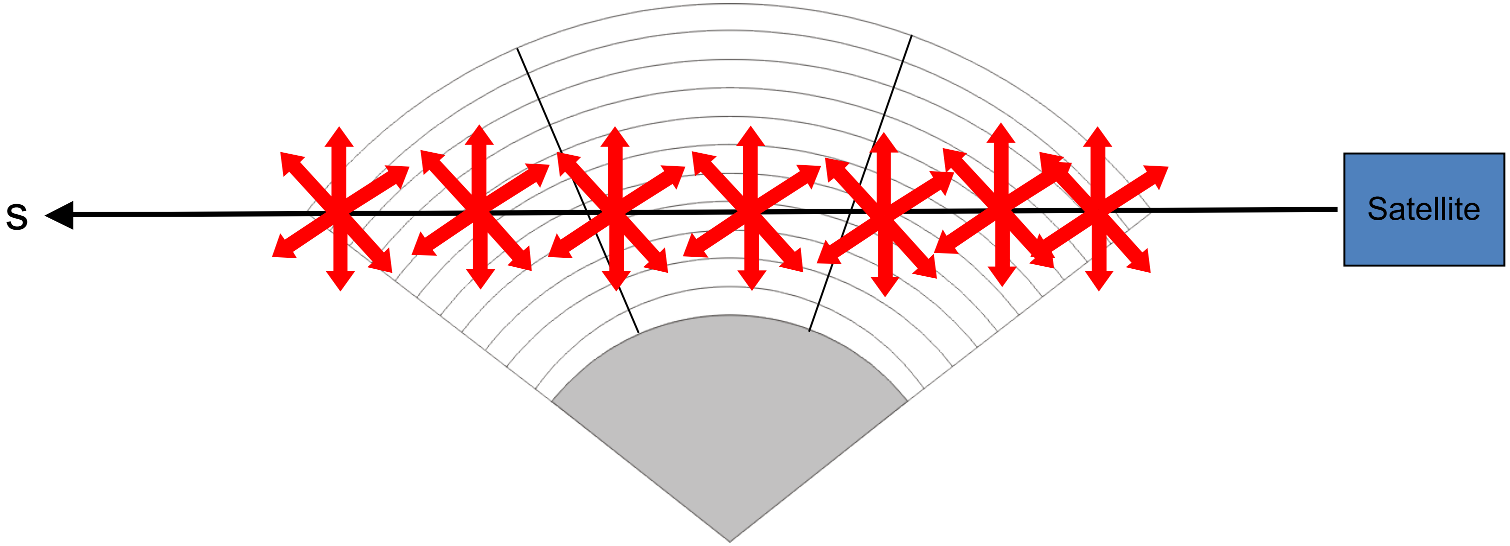
Successive Orders Algorithm

Step 3. Repeat until convergence.


Calculate incoming radiance
s
Satellite
Scatter incoming radiances
propagate outbound source functions
Scatter to generate outbound sources
s
Satellite

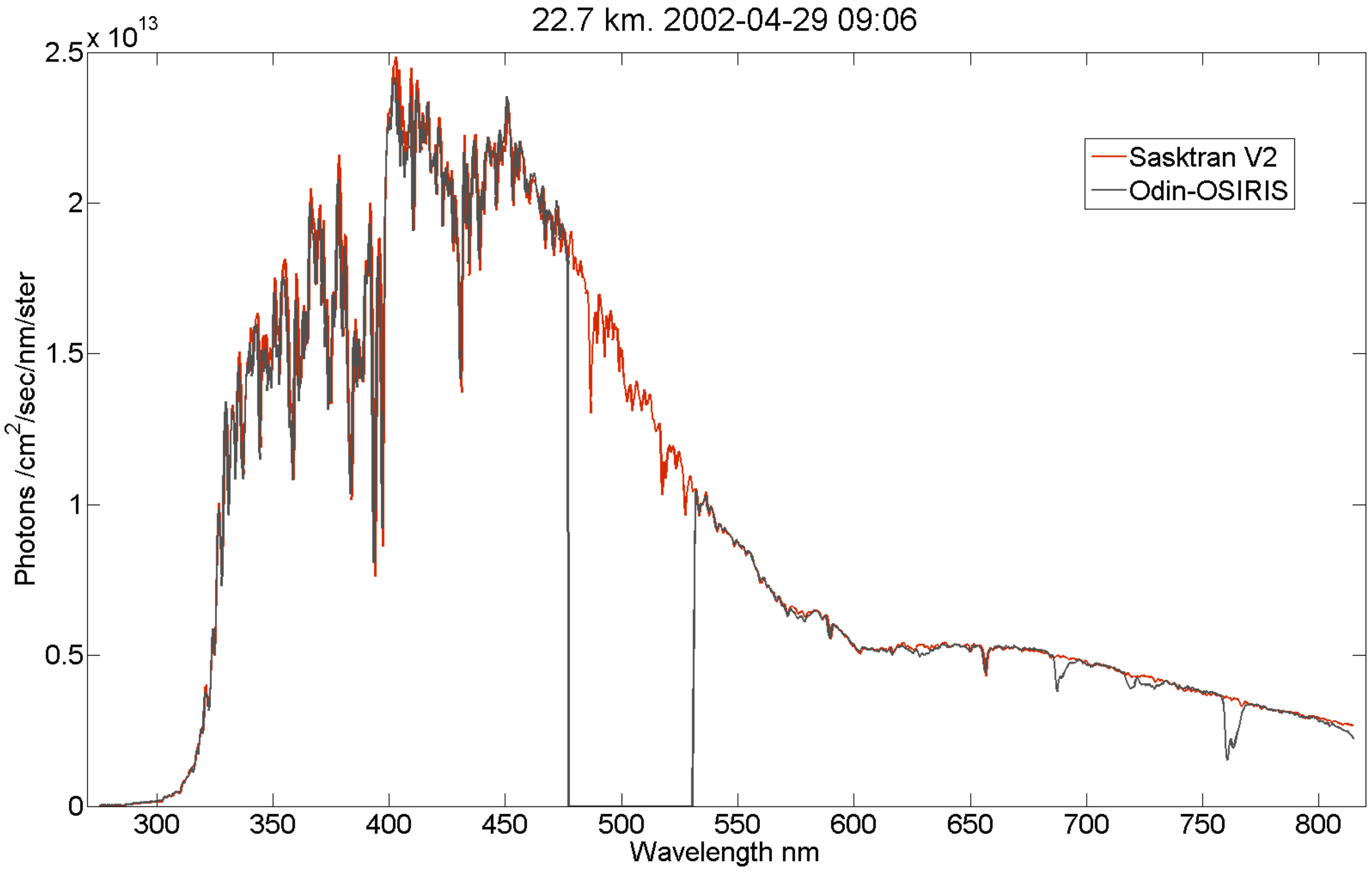
Successive Orders Algorithm

Final Step. Calculate the radiative transfer integral along the rays of interest.

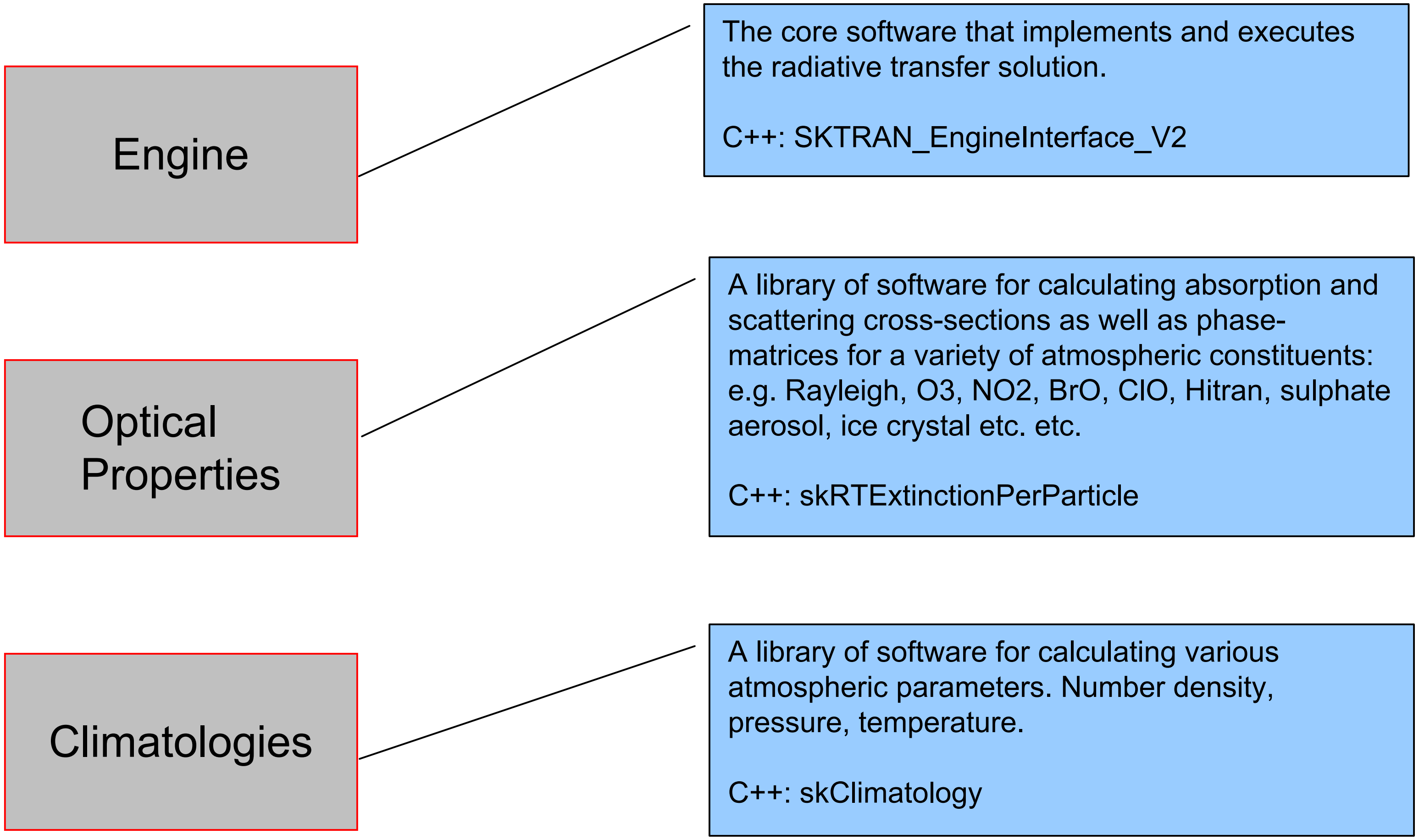


Comparison with Data

Result: A favourable comparison with measurements.



Software Organization



Engine

The core software that implements and executes the radiative transfer solution.

C++: SKTRAN_EngineInterface_V2

Optical Properties

A library of software for calculating absorption and scattering cross-sections as well as phase-matrices for a variety of atmospheric constituents: e.g. Rayleigh, O3, NO2, BrO, ClO, Hitran, sulphate aerosol, ice crystal etc. etc.

C++: skRTExtinctionPerParticle

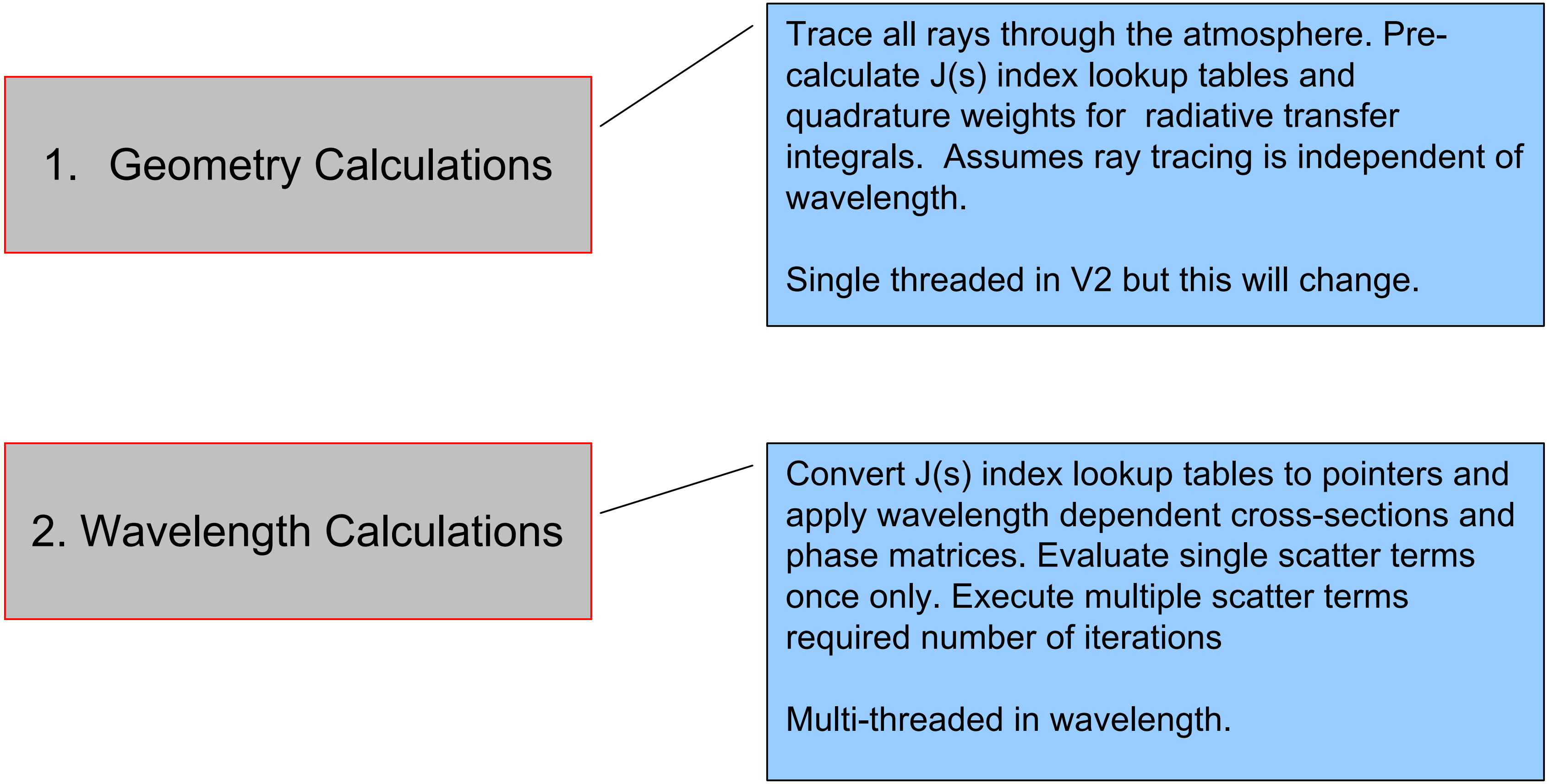
Climatologies

A library of software for calculating various atmospheric parameters. Number density, pressure, temperature.

C++: skClimatology

Software Organization: Engine

SASKTRAN Engine: A two stage, multi-threaded processing engine.



1. Geometry Calculations

Trace all rays through the atmosphere. Pre-calculate J(s) index lookup tables and quadrature weights for radiative transfer integrals. Assumes ray tracing is independent of wavelength.

Single threaded in V2 but this will change.

2. Wavelength Calculations

Convert J(s) index lookup tables to pointers and apply wavelength dependent cross-sections and phase matrices. Evaluate single scatter terms once only. Execute multiple scatter terms required number of iterations

Multi-threaded in wavelength.

Software Organization: Engine

Implemented in C++.

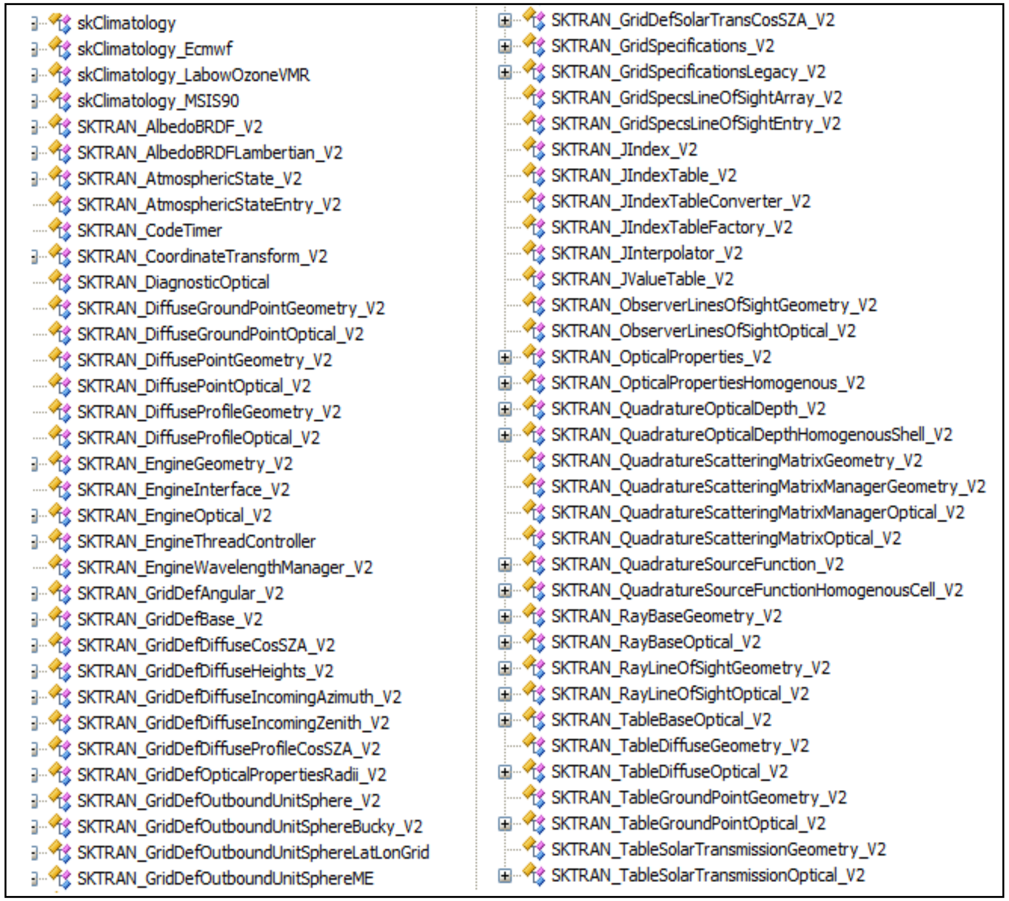
Linux and Windows. 32 bit and 64 bit

User defined everything!

- Ray tracing grids
- Diffuse point profiles
- Solar transmission tables
- Diffuse incoming lines of sight
- Diffuse outbound directions
- Quadrature schemes
- Unlimited list of species.

Sensible defaults are provided by the code but power is under the hood for the adventurous user.

~60 C++ classes define V2 engine



```
skClimatology
skClimatology_Ecmwf
skClimatology_LabowOzoneVMR
skClimatology_MSIS90
SKTRAN_AlbedoBRDF_V2
SKTRAN_AlbedoBRDFLambertian_V2
SKTRAN_AtmosphericState_V2
SKTRAN_AtmosphericStateEntry_V2
SKTRAN_CodeTimer
SKTRAN_CoordinateTransform_V2
SKTRAN_DiagnosticOptical
SKTRAN_DiffuseGroundPointGeometry_V2
SKTRAN_DiffuseGroundPointOptical_V2
SKTRAN_DiffusePointGeometry_V2
SKTRAN_DiffusePointOptical_V2
SKTRAN_DiffuseProfileGeometry_V2
SKTRAN_DiffuseProfileOptical_V2
SKTRAN_EngineGeometry_V2
SKTRAN_EngineInterface_V2
SKTRAN_EngineOptical_V2
SKTRAN_EngineThreadController
SKTRAN_EngineWavelengthManager_V2
SKTRAN_GridDefAngular_V2
SKTRAN_GridDefBase_V2
SKTRAN_GridDefDiffuseCosSZA_V2
SKTRAN_GridDefDiffuseHeights_V2
SKTRAN_GridDefDiffuseIncomingAzimuth_V2
SKTRAN_GridDefDiffuseIncomingZenith_V2
SKTRAN_GridDefDiffuseProfileCosSZA_V2
SKTRAN_GridDefOpticalPropertiesRadii_V2
SKTRAN_GridDefOutboundUnitSphere_V2
SKTRAN_GridDefOutboundUnitSphereBucky_V2
SKTRAN_GridDefOutboundUnitSphereLatLonGrid
SKTRAN_GridDefOutboundUnitSphereME
SKTRAN_GridDefSolarTransCosSZA_V2
SKTRAN_GridSpecifications_V2
SKTRAN_GridSpecificationsLegacy_V2
SKTRAN_GridSpecsLineOfSightArray_V2
SKTRAN_GridSpecsLineOfSightEntry_V2
SKTRAN_JIndex_V2
SKTRAN_JIndexTable_V2
SKTRAN_JIndexTableConverter_V2
SKTRAN_JIndexTableFactory_V2
SKTRAN_JInterpolator_V2
SKTRAN_JValueTable_V2
SKTRAN_ObserverLinesOfSightGeometry_V2
SKTRAN_ObserverLinesOfSightOptical_V2
SKTRAN_OpticalProperties_V2
SKTRAN_OpticalPropertiesHomogenous_V2
SKTRAN_QuadratureOpticalDepth_V2
SKTRAN_QuadratureOpticalDepthHomogenousShell_V2
SKTRAN_QuadratureScatteringMatrixGeometry_V2
SKTRAN_QuadratureScatteringMatrixManagerGeometry_V2
SKTRAN_QuadratureScatteringMatrixManagerOptical_V2
SKTRAN_QuadratureScatteringMatrixOptical_V2
SKTRAN_QuadratureSourceFunction_V2
SKTRAN_QuadratureSourceFunctionHomogenousCell_V2
SKTRAN_RayBaseGeometry_V2
SKTRAN_RayBaseOptical_V2
SKTRAN_RayLineOfSightGeometry_V2
SKTRAN_RayLineOfSightOptical_V2
SKTRAN_TableBaseOptical_V2
SKTRAN_TableDiffuseGeometry_V2
SKTRAN_TableDiffuseOptical_V2
SKTRAN_TableGroundPointGeometry_V2
SKTRAN_TableGroundPointOptical_V2
SKTRAN_TableSolarTransmissionGeometry_V2
SKTRAN_TableSolarTransmissionOptical_V2
```

Software Organization: Optical Properties

The optical properties is a library of C++ objects that provide the absorption, scattering and extinction cross-sections of various atmospheric constituents. Cross-sections are assumed to be a function of atmospheric state (T,P etc.)

They all derived from a single base class: **skRTExtinctionPerParticle**

- O3 High Resolution (Bogumil)
- O3 OSIRIS resolution (Bogumil)
- O3 FTS Voigt
- O3 Bass Paur
- O3 Daumont-Brion-Malicet
- O3 GOME Burrows
- NO2 High Resolution (Bogumil)
- NO2 OSIRIS resolution (Bogumil)
- Rayleigh Simple
- Rayleigh Dry Air (Bates 1984)
- Sulphate aerosols (MIE code)
- Ice Crystals (T-Matrix code)

and many more to come: Hitran, BrO, OClO ….

Software Organization: Climatologies

The climatology code is a library of C++ objects that provide the number density of the various atmospheric constituents. They are required to work out the total optical properties of any volume of the atmosphere.

They all derived from a single base class: **skRTClimatology**

- ECMWF (P, T, density)
- ECMWF along Odin orbit
- MSIS-CIRA
- Labow
- User defined tables (any species)

Software Resources

Memory:

The Sasktran code is memory hungry: A two thread application uses 400 MB of memory. An 8 thread application uses 800 MB of memory. This strategy allows the model to cache results in memory rather than repeatedly re-calculate intermediate numbers.

CPU:

The Sasktran code is optimized for inversion algorithms where the ray tracing code is constant but the optical properties change between iterations. Under these conditions it can calculate the radiance for N wavelengths with 20-50 orders of scatter in about 5 seconds where N is the number of processors on the machine (2.5 GHz Intel processors).

Odin-OSIRIS operations

The Odin-OSIRIS data centre is reprocessing the entire 8-9 year mission. We have acquired 12 rack mounted machines. 8 machines are quad dual-core (8 processors per machine) and 4 machines are dual, dual-core (4 processors per machine).

We anticipate these machines will be able reprocess the entire mission (O3 and aerosols) within 6-9 months.

We are happy to share this code with other groups. Please contact one of use for more information.

Thank you