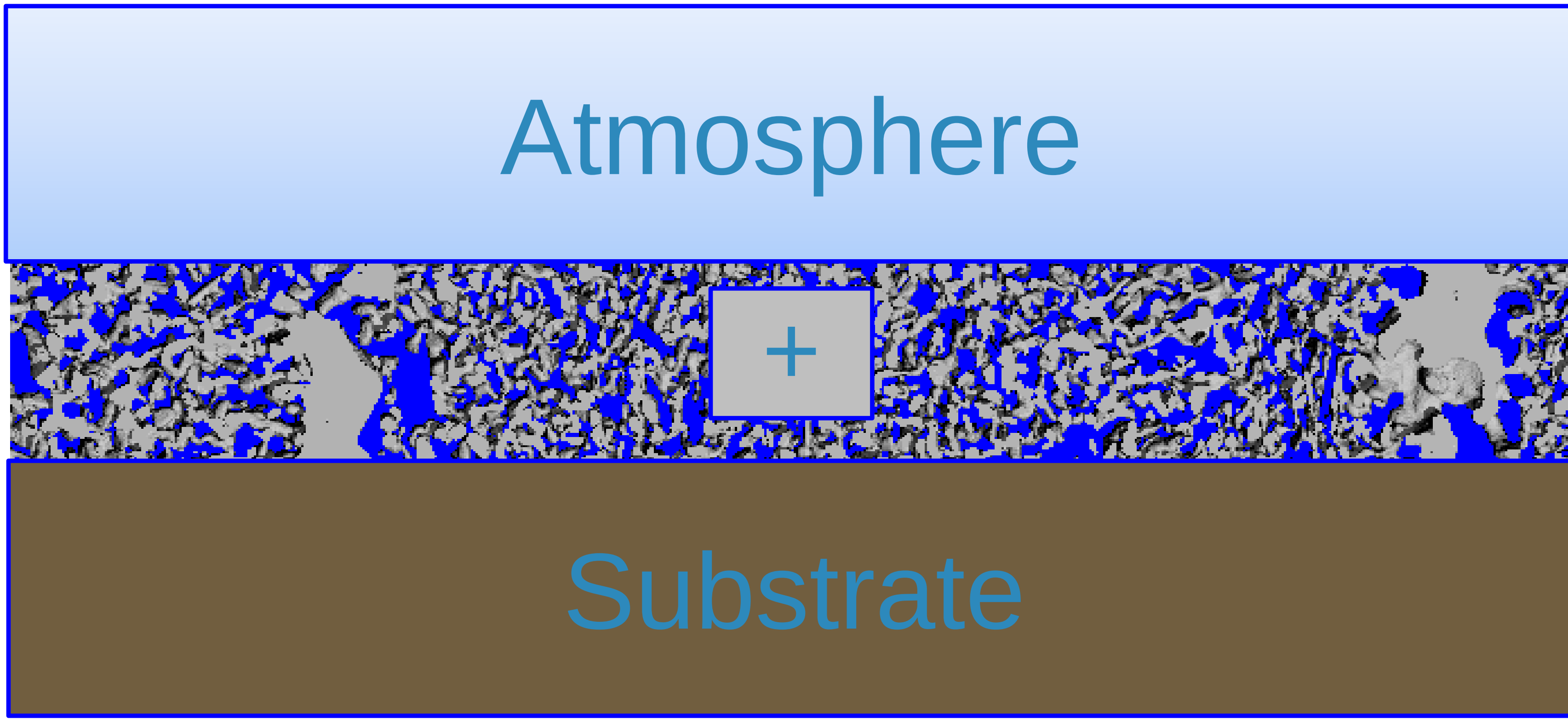

Atmosphere
+
Substrate

Outline

- Substrate
  - Models (Fresnel, rough soil)
  - Specify substrate in SMRT
- Atmosphere
  - Structure
  - Representation in SMRT

Need a lower boundary!

Types of substrate

SMRT devel

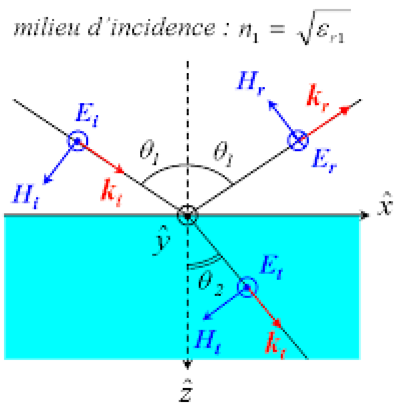
A way to specify the lower boundary: what is underneath the lowest layer

Specialist materials: reflector plate, absorber….

It's a choice!

$$\frac{\partial T}{\partial z}? \quad \lambda?$$

Generic, flat surface (Fresnel)

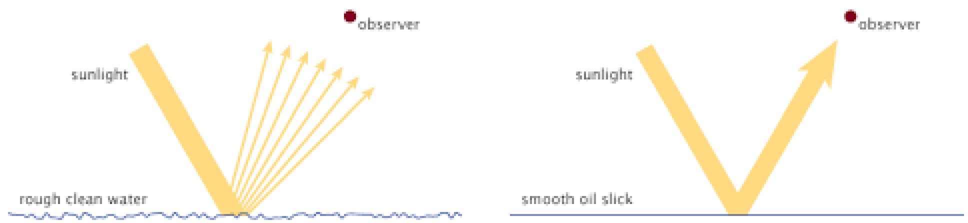


Use law of refraction: $n_1 \sin \theta_1 = n_2 \sin \theta_2$

$$\mathbf{R}^{bottom,(l),[specular]}(\mu) = \begin{bmatrix} \left(\frac{\epsilon_{eff}^{(l+1)} \cos\theta - \sqrt{\epsilon_{eff}^{(l)}}\sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}}{\epsilon_{eff}^{(l+1)} \cos\theta + \sqrt{\epsilon_{eff}^{(l)}}\sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}} \right)^2 & 0 \\ 0 & \left(\frac{\sqrt{\epsilon_{eff}^{(l)}} \cos\theta - \sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}}{\sqrt{\epsilon_{eff}^{(l)}} \cos\theta + \sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}} \right)^2 \end{bmatrix}$$

- SMRT layers are numbered from top = 0
- No cross-pol terms!!

Modification for effect of roughness



IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 37, NO. 3, MAY 1999 1391

Rough Bare Soil Reflectivity Model

Urs Wegmüller, *Member, IEEE,* and Christian Mätzler, *Member, IEEE*

***Abstract*— A semiempirical model for the reflectivity of rough bare soil is presented. One of the main objectives of this new model development was to derive a simple model with few model parameters and a wide applicability. A large number of ground-based measurements in the 1–100-GHz range at H- and V-polarization and incidence angles between 20° and 70° were used for the model development.**

***Index Terms*— Bare soil, emissivity, model, reflectivity, soil moisture, surface roughness.**

practice, this often turns out to be critical as the assumptions made in the modeling of bistatic scattering are often not valid over the entire range of angles. In addition, theoretical models tend to be complicated and require very detailed knowledge on the surface geometry. Therefore, a simple semiempirical model may be preferred in many cases.

One often used semiempirical expression for the rough surface reflectivity, as described for example by Wang and

Modification for effect of roughness

Wegmüller and Mätzler, 1999

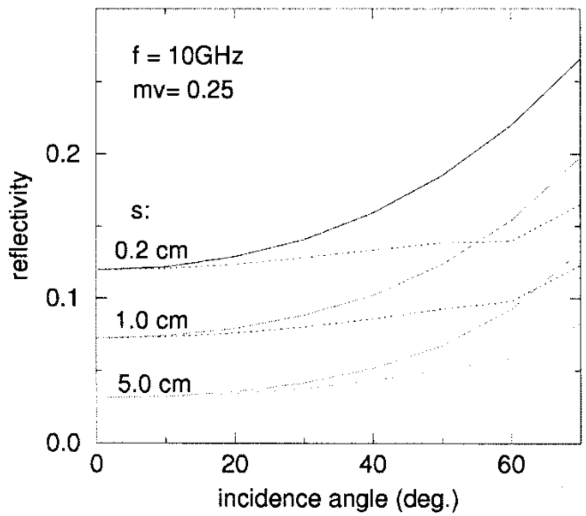


Fig. 1. Rough bare soil reflectivity model. Simulated incidence angle dependence of rough bare soil reflectivity at H- (solid) and V–polarization (dotted) for different standard deviations of the surface height s.

The new rough bare soil reflectivity model is defined by

$$r_{h,\,\mathrm{mod}}(mv, ks, \theta) = r_{h,\,\mathrm{Fresnel}} \cdot \exp\left\{-(ks)^{\sqrt{0.10\cos\theta}}\right\} \quad (12)$$

$\theta \leq 60°$:

$$r_{v,\,\mathrm{mod}}(mv, ks, \theta) = r_{h,\,\mathrm{mod}}(mv, ks, \theta) \cdot (\cos\theta)^{0.655} \quad (13a)$$

$60° \leq \theta \leq 70°$:

$$r_{v,\,\mathrm{mod}}(mv, ks, \theta) = r_{h,\,\mathrm{mod}}(mv, ks, \theta) \cdot (0.635 - 0.0014 \cdot (\theta - 60°)). \quad (13b)$$

The range of validity is restricted to the 1–100-GHz range at H- and V-polarization and incidence angles between 0° and 70°. The range of validity with respect to the standard

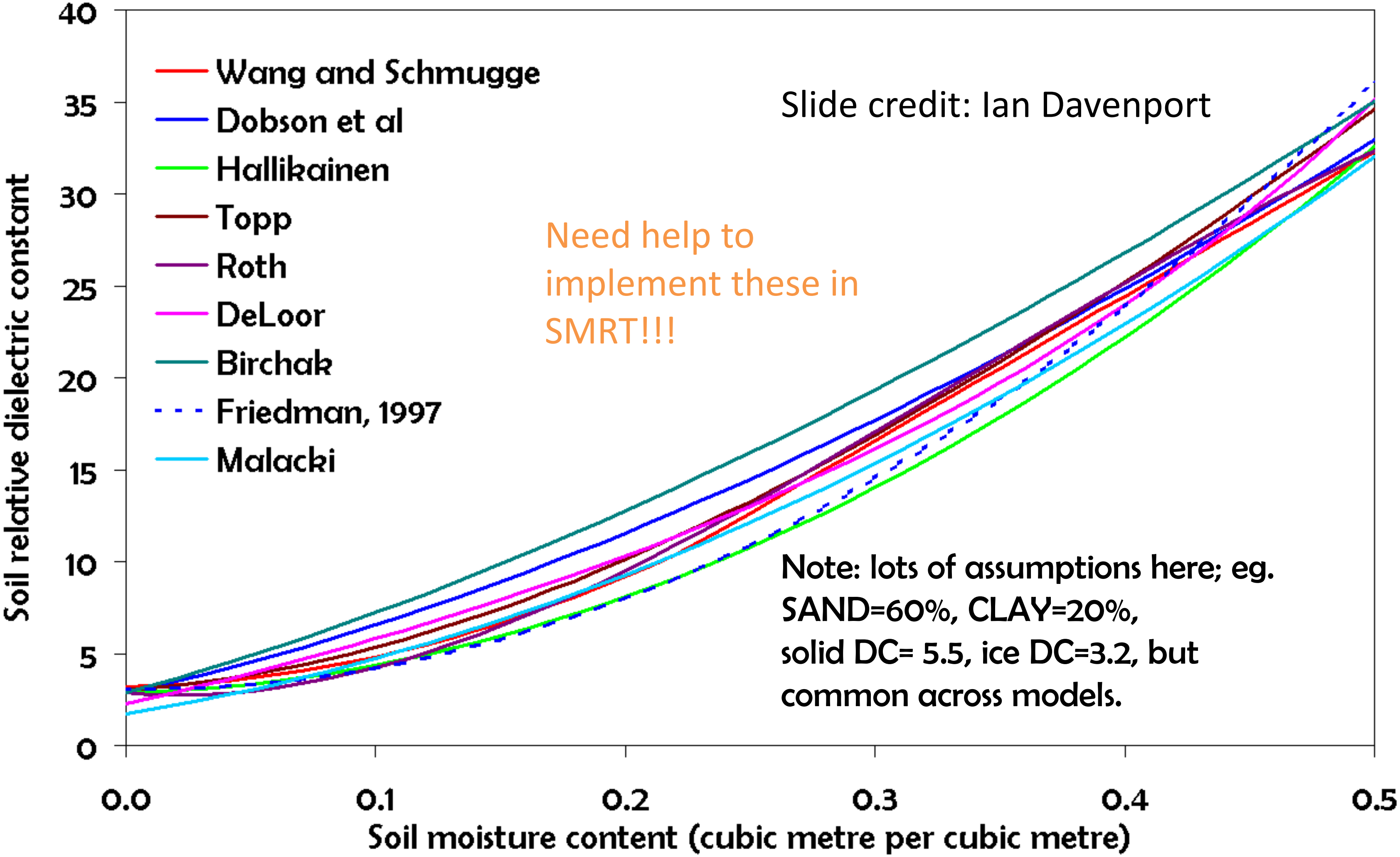
SMRT: Passive only

Generic, flat surface (Fresnel)

$$\mathbf{R}^{bottom,(l),[specular]}(\mu) = \begin{bmatrix} \left(\frac{\epsilon_{eff}^{(l+1)} \cos\theta - \sqrt{\epsilon_{eff}^{(l)}}\sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}}{\epsilon_{eff}^{(l+1)} \cos\theta + \sqrt{\epsilon_{eff}^{(l)}}\sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}} \right)^2 & 0 \\ 0 & \left(\frac{\sqrt{\epsilon_{eff}^{(l)}} \cos\theta - \sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}}{\sqrt{\epsilon_{eff}^{(l)}} \cos\theta + \sqrt{\epsilon_{eff}^{(l+1)} - \epsilon_{eff}^{(l)} \sin^2\theta}} \right)^2 \end{bmatrix}$$

Need to define soil permittivity. Depends on:
- Soil moisture
- Soil type
- Roughness
- Bulk soil density

Empirical models of soil dielectric constant



Slide credit: Ian Davenport

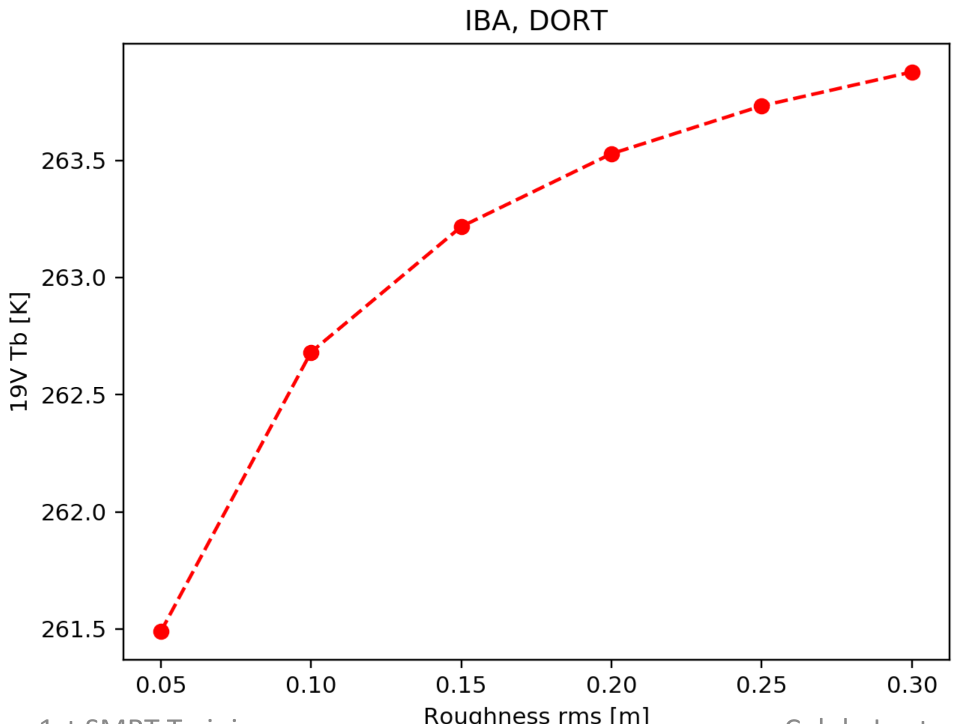
Need help to implement these in SMRT!!!

Note: lots of assumptions here; eg. SAND=60%, CLAY=20%, solid DC= 5.5, ice DC=3.2, but common across models.

SMRT substrate: soil

Make a soil substrate with Wegmüller and Mätzler (1999) model

```
In [1]: from smrt import make_soil, make_snowpack

In [2]: soil = make_soil('soil_wegmuller', 'dobson85', temperature=265, roughness_rms=0.25,
                         moisture=0.25, sand=0.01, clay=0.7, drymatter=1300)

In [3]: snow_with_soil = make_snowpack([1], "exponential", temperature=[265], density=[280], corr_length=[5e-5], substrate=soil)
```

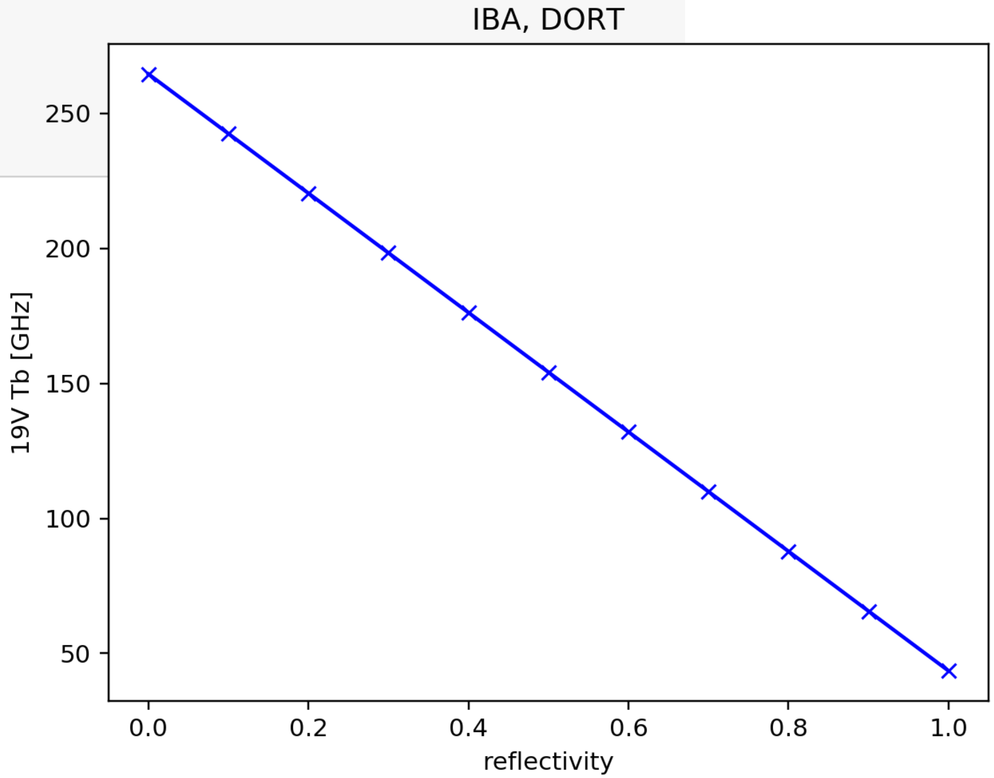


This way of specifying soil available in DMRT-ML

Reflector

SMRT substrate: reflector

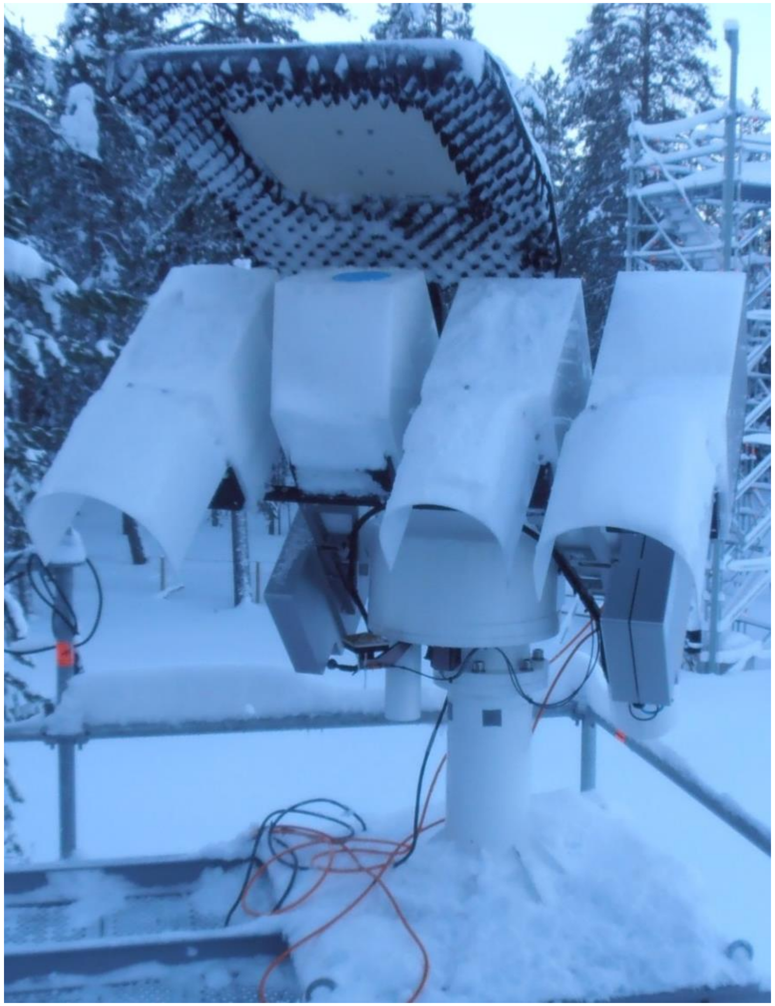
```
from smrt.substrate.reflector import make_reflector
refl = np.arange(0, 1.1, 0.1)
reflector = [make_reflector(temperature=265, specular_reflection=r) for r in refl]
snowpacks = [make_snowpack([1], "exponential", temperature=[265], density=[280],
                            corr_length=[5e-5], substrate=r) for r in reflector]

# Run model on snowpacks
results = m.run(rad, snowpacks)

# Plot results
plt.plot(refl, results.TbV(),'bx-')
plt.xlabel('reflectivity')
plt.ylabel('19V Tb [GHz]')
plt.title('IBA, DORT')
```



MEMLS uses this approach

Useful for SMRT evaluation…

ASMEX: RADIOMETRIC

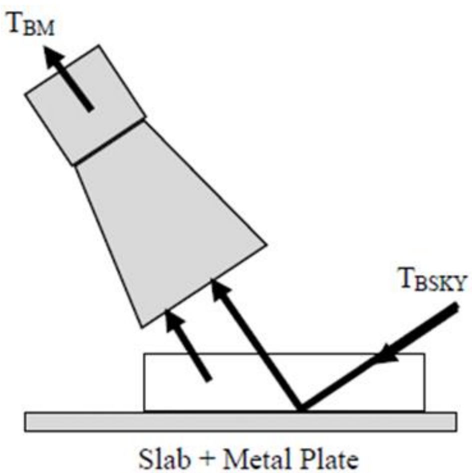


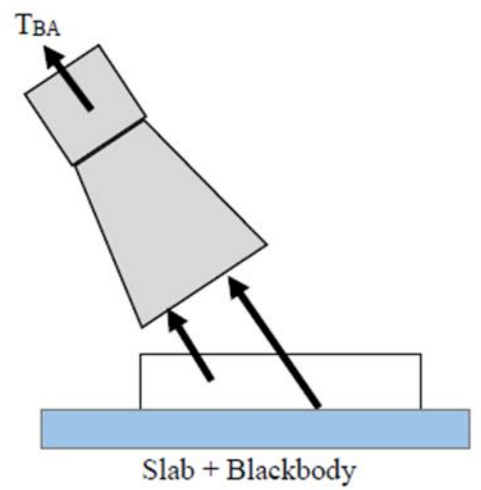
Slide credit: *William Maslanka | Department Seminar*

COEFFICIENT CALCULATION

Currently aiming to calculate the scattering and absorption coefficients via method laid out by Wiesmann et al. 1998, using a Flux Coefficient Model.

- Calculating the reflectivities of the slab upon an absorbing and reflecting base
- .



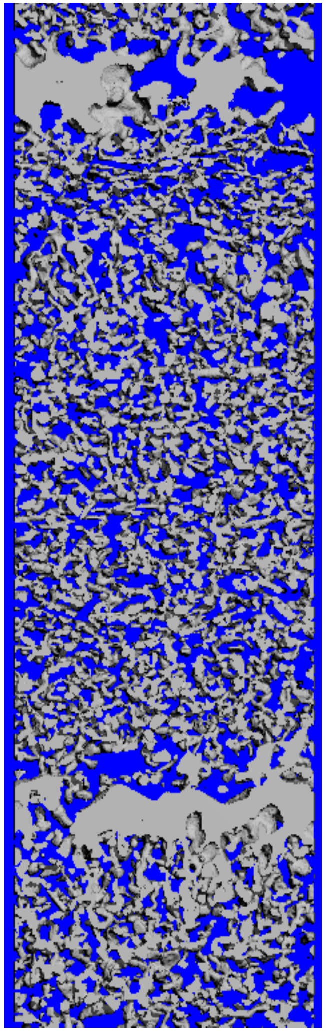


$$r_{met} = \frac{T_{BM} - T_{phys}}{T_{BSKY} - T_{phys}}$$

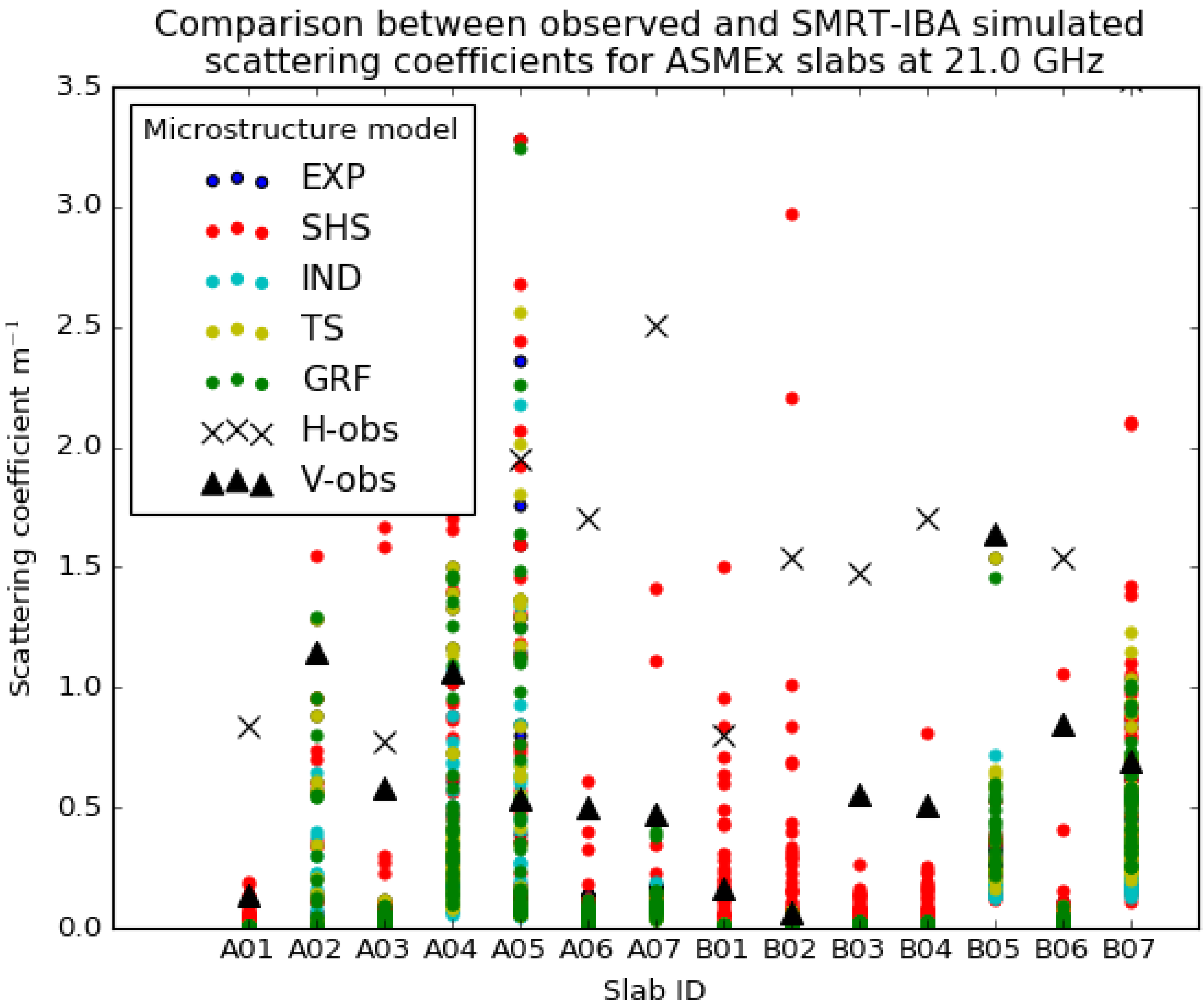
$$r_{abs} = \frac{T_{BA} - T_{phys}}{T_{BSKY} - T_{phys}}$$

Slide credit: *William Maslanka | Department Seminar*

ASMEx scattering coefficients



A02B-B



SMRT substrate: active reflector

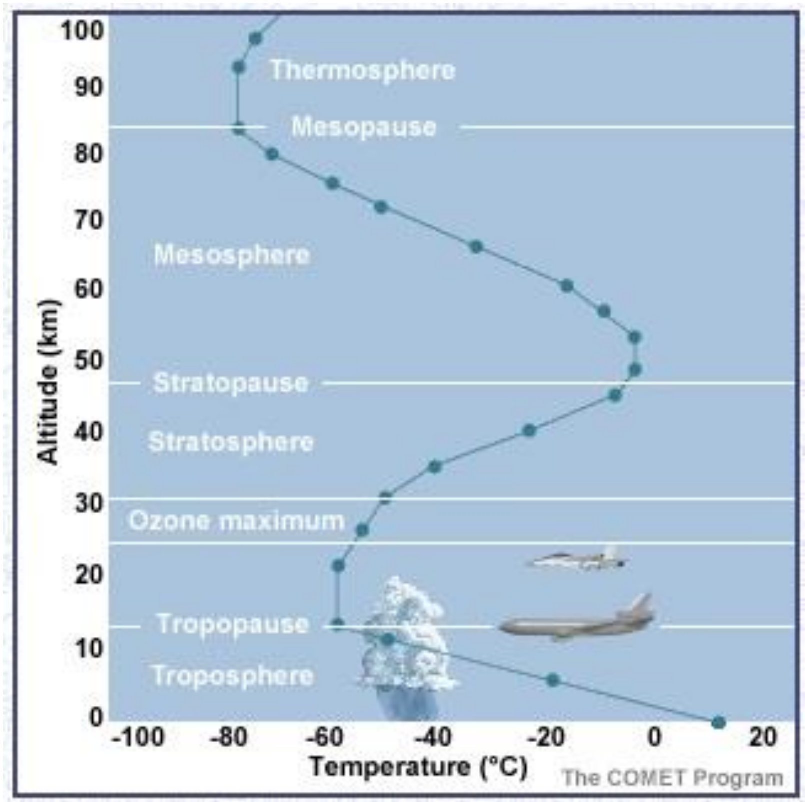
```
from smrt.inputs.sensor_list import active
from smrt.utils import dB
from smrt.substrate.reflector_backscatter import make_reflector

# Need to define an active sensor
scatt = active(13e9, 45)

# Make active reflector
reflector = make_reflector(temperature=265, specular_reflection=0.,
                           backscattering_coefficient={'VV': 0.1, 'HH': 0.1})

# Make snowpack
snow_active = make_snowpack([1], "exponential", temperature=[265],
                            density=[280], corr_length=[5e-5], substrate=reflector)
dB(m.run(scatt, snow_active).sigmaVV())
```

```
active model is not yet fully implemented, need modification for the third component

array([-12.23784209])
```

Atmosphere

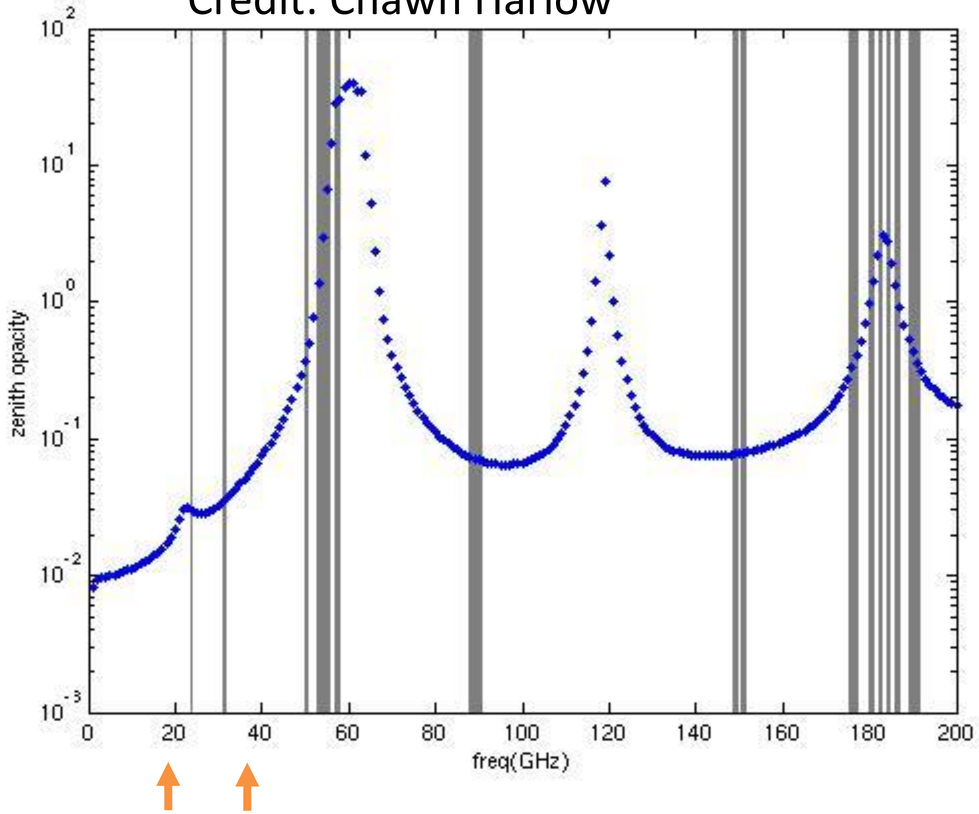
http://maxpixel.freegreatpicture.com/Blue-Atmosphere-Space-Clouds-Flying-Milky-Way-2295189

Structure of Atmosphere



Atmospheric Opacity

Credit: Chawn Harlow



Basic atmosphere in SMRT for higher transparency frequencies

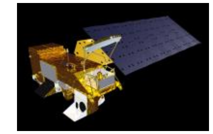
Grey bands: observation channels from AMSU-A and MHS

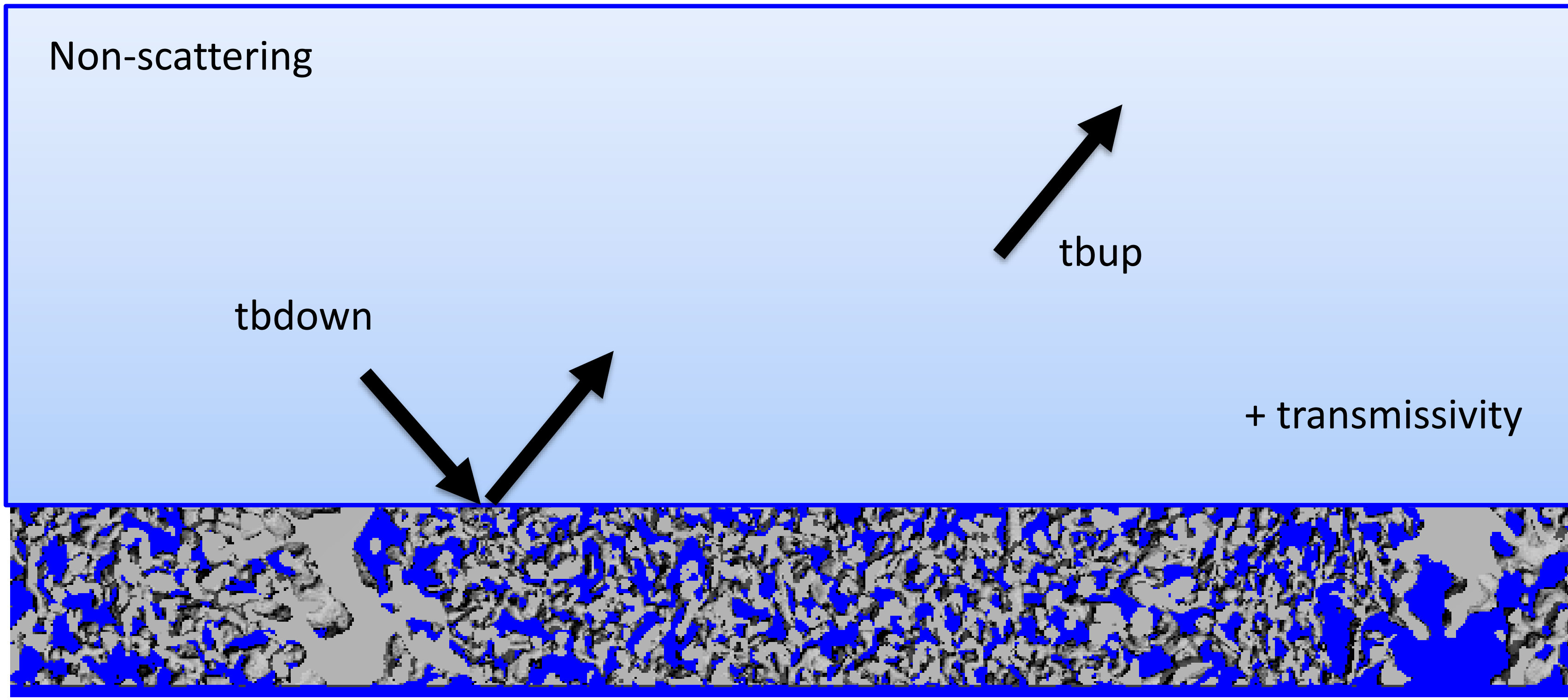
Blue: example atmospheric opacity

Need a good model of surface emissivity to assimilate TOA TB for NWP

SMRT Basic Atmosphere

(NASA image by Marit Jentoft-Nilsen.)





SMRT Basic Atmosphere

In [1]:
```
from smrt import make_snowpack, make_model
from smrt.inputs.sensor_list import passive
from smrt.atmosphere.simple_isotropic_atmosphere import SimpleIsotropicAtmosphere
```

In [2]:
```
# Create snowpack, sensor and model
snowpack = make_snowpack([10], 'independent_sphere', temperature=260., density=320., radius=0.5e-3)
rad = passive(21e9, 35)
model = make_model('rayleigh', 'dort')
```

In [3]:
```
atmos = SimpleIsotropicAtmosphere(tbdown=30., tbup=6., trans=0.90)
model.run(rad, snowpack, atmosphere=atmos).TbV()
```

Out[3]:
```
149.5455044014671
```

If no atmosphere specified: tbup = 0, tbdown = 0, trans = 1

Idea:

medium = make_atmosphere(...) + make_snowpack(...) + make_ice(...) + make_ocean(...)
m.run(sensor, medium)