relative_azimuth-satellites

July 16, 2020

This notebook is intended to plot the solar and satellite angles from each of my regression tests, to explain the physical situation that creates that geometry, and to justify the values of relative azimuth returned.

Quoting Greg: RAA is 180 when sun and satellite are on opposite sides of the pixel, and RAA = 0 when sun and satellite are on the same side of the pixel. This is standard in plane-parallel RT and is assumed throughout the main processor and in the lookup tables. Let's call this the DISORT standard. Let's call the opposite the anti-DISORT standard.

The Cartesian coordinate system origin is in the centre of the observed pixel.

The x-axis points west to east
The y-axis points south to north
The x and y-axis are undefined at the north & south pole

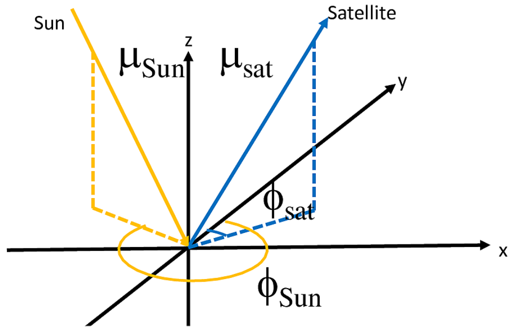
The z-axis is normal to the x-y plane except at the poles where it is undefined. At the poles the z-axis points along the Earth's rotation axis away from the Earth.

The optical depth, τ, is measure from the top-of-atmosphere (TOA) so is related to z by $\tau = (z_{toa}-z)\beta^{ext}$ where z_{toa} is the height of the TOA and β^{ext} is the volume extinction coefficient.

μ_{sat} is the cosine of the satellite zenith angle
μ_{Sun} is the cosine of the solar zenith angle

ϕ_{sat} is the satellite azimuth angle (i.e. North is zero). Note that ϕ_{sat} is undefined when the satellite is above the pixel on the z axis.

ϕ_{Sun} is the solar azimuth angle. Note that ϕ_{Sun} is undefined when the Sun is directly above the pixel on the z axis.



As the solution of the 1D radiative transfer equation is the same for a azimuth difference of $\phi_{Sun} - \phi_{sat}$ and $\phi_{sat} - \phi_{Sun}$ only differences in the range [0,180] need be considered. To facilitate this $\Delta\phi$ is defined as the interior angle between the solar and satellite azimuth angles.

Exactly the same geometry is used to describe the reflection by the surface. However reflection may need to be expressed over the full range of ϕ_{Sun} and ϕ_{sat}.

Look-up tables describing the reflection and transmission of the atmosphere can be constructed in terms of (μ_{sat}, μ_{sun}, $\Delta\phi$)

Figure 1: A definition of azimuth and zenith angles by Don.

In our look-up tables, we define:

- Satellite zenith angle over 10 points from $0 - 89°$;
- Solar zenith angle over 10 points from $0 - 89°$;
- Relative azimuth angle over 11 points from $0 - 180°$.

As the first two axes are unevenly spaced (in the last point), positions within each LUT are identified in `SetGZero()` using the construction `max(min(locate(SAD_LUT%Grid%ANGLE(1:n), value), n-1, 1)`. That is horrendously inefficient.

I am looking for references to relative azimuth in the pre-processor,

```
atmlxdt054:~/orac$ grep relazi -il pre_processing/*
allocate_imager_structures.F90
allocate_preproc_structures.F90
build_preproc_fields.F90
cloud_typing_pavolonis.F90
deallocate_imager_structures.F90
deallocate_preproc_structures.F90
get_surface_reflectance.F90
imager_structures.F90
netcdf_output_create_file.F90
netcdf_output.F90
netcdf_output_write_swath.F90
preproc_structures.F90
read_aatsr_l1b.F90
read_abi_funcs.F90
read_abi_main.F90
read_agri.F90
read_avhrr_time_lat_lon_angles.F90
read_himawari.F90
read_modis_time_lat_lon_angles.F90
read_seviri.F90
read_slstr_funcs.F90
read_viirs_Iband.F90
read_viirs_Mband.F90
```

The `allocate_*`, `deallocate_*`, and `*_structures` references are simply declarations. The `netcdf_output*` references write our results into output files and don't manipulate the data. `build_preproc_fields.F90` aggregates data onto the grid on which we evaluate RTTOV.

Looking for satellite and solar azimuth also highlights,

```
atmlxdt054:~/orac$ grep satazi -il pre_processing/*
rttov_driver.F90
rttov_driver_gfs.F90
```

Thus, we need to evaluate the various satellite read routines, the surface reflectance calculation, the cloud type neural net, and the call to RTTOV.

[1]:
```python
# Load the required Python modules
import cartopy.crs as ccrs # Mapping
import matplotlib.pyplot as plt # Plotting library
import numpy as np # Array support and maths
import seaborn as sns # Pretty formatting

from netCDF4 import Dataset # Read NetCDF files
from pyorac.swath import Swath # Read ORAC files
```

[2]:
```python
# Define a function that saves me writing out full paths each time
# Feel no need to read it
```

```python
def regression_filename(compiler, test, rev, typ="", land=False,
                        secondary=False, local=False):
    """Locates ORAC outputs in Adam's regression test collection."""
    from glob import glob
    from os.path import join

    PREPROC_EXT = ("config", "loc", "geo", "lsf", "clf", "alb", "prtm", "swrtm",␣
 ↪"lwrtm", "msi")
    PROC_EXT = ("primary", "secondary")
    CLOUD_TYPES = ("WAT", "ICE", "ICE_WAT")
    AEROSOL_TYPES = tuple("A{:2d}".format(i+70) for i in range(10))
    ASH_TYPES = ("EYJ", )
    ALL_TYPES = CLOUD_TYPES + AEROSOL_TYPES + ASH_TYPES + ("", )

    if local:
        root = "/data/povey/outputs"
    else:
        root = "/network/aopp/matin/eodg/shared/orac/data/testoutput"

    parts = [root, compiler + "_cld", test]
    try:
        suffix = "R{:4d}".format(rev)
    except ValueError:
        raise ValueError("revision must be an integer.")

    # Identify subfolder
    if typ in PREPROC_EXT:
        parts.append("pre")
        suffix += "." + typ + ".nc"
    elif typ in ALL_TYPES:
        suffix += typ + "."
        suffix += "secondary" if secondary else "primary"
        suffix += ".nc"
        if typ.startswith("A"):
            parts.append("land" if land else "sea")
        elif typ:
            parts.append("cld")
    else:
        raise ValueError("typ must be a label in PREPROC_EXT or ALL_TYPES: "+typ)

    # Add suffix of the desired filename
    parts.append("*" + suffix)
    search = join(*parts)
    filename = glob(search)

    try:
        return filename[0]
    except IndexError:
        raise FileNotFoundError("Could not locate " + search)
```

[3]:
```
# Make a basic plot to load the settings so I can change them
plt.plot((0,1))
plt.close()

# Use paper-like fonts and the math font designed to go with Times
sns.set_style("whitegrid", {
    "axes.axisbelow" : False, "font.family":"serif",
    "font.serif":["Times New Roman"]
})
sns.set_palette("tab10", 10)
plt.rcParams.update({"mathtext.fontset": "stix", "font.size": 22, "image.cmap":␣
 ↪"nipy_spectral"})
```

First, MODIS. It's June 20th 2008 around Papa New Guinea and northern Australia.

Azimuth is defined as clockwise from north to east. It calculates the relative azimuth here with,

```
imager_angles%relazi(:,:,1) = abs(imager_angles%solazi(:,:,1) - temp)
where (imager_angles%relazi(:,:,1) .gt. 180.)
    imager_angles%relazi(:,:,1) = 360. - imager_angles%relazi(:,:,1)
end where
```

which was changed by me here.

Technically, plotting these is a pain because the 10-line scans overlap slightly and I haven't dealt with that in the `Swath` class. However, it doesn't make a notecable difference to these images and so has been ignored.

[4]:
```
compiler = "gfort"
test = "DAYMOD"
revision = 1965

with Swath(regression_filename(compiler, test, revision)) as pri_set, \
     Dataset(regression_filename(compiler, test, revision, "geo")) as geo_set:
    fig, axes = plt.subplots(1, 5, figsize=(25, 4),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))
    for ax, var in zip(axes.ravel(), ("solzen", "satzen", "solaz", "sataz",␣
 ↪"relazi")):
        ax.coastlines("10m")
        im = pri_set.map(ax, geo_set[var][0])
        cb = fig.colorbar(im, ax=ax)
        ax.set_title(var)
    fig.tight_layout()
    plt.show()
```

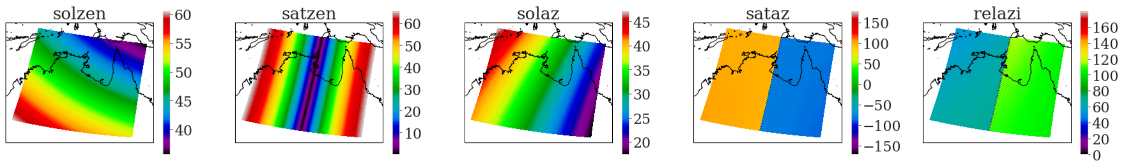


Figure 2: Solar and satellite zenith and viewing angles over northern Australia from MODIS Terra at 11:15 on 20/06/2008. The satellite is moving from the top to the bottom of the image.

Imagine an observer on the ground looking north from various points within the satellite swath. In this scene, the sun is ahead and right of them as the solar azimuth is $20-45°$. The satellite azimuth is dominated by two values, 100 and $-80°$ which place the satellite either right and a little in behind of or left and a little in front of the observer.

Therefore, the relative azimuth is small on the left side of the swath and large on the right.

[5]:
```
test = "DAYMYD"

with Swath(regression_filename(compiler, test, revision)) as pri_set, \
     Dataset(regression_filename(compiler, test, revision, "geo")) as geo_set:
    fig, axes = plt.subplots(1, 5, figsize=(25, 4),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))
    for ax, var in zip(axes.ravel(), ("solzen", "satzen", "solaz", "sataz",␣
 ↪"relazi")):
        ax.coastlines("10m")
        im = pri_set.map(ax, geo_set[var][0])
        fig.colorbar(im, ax=ax)
        ax.set_title(var)
    fig.tight_layout()
    plt.show()
```

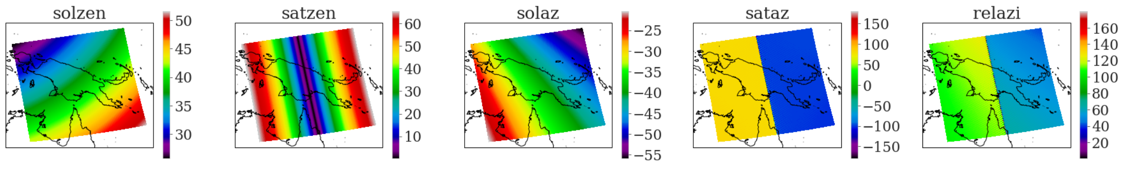


Figure 3: MODIS Aqua at 04:05 on 20/06/2008. The satellite is moving from the bottom to the top of the image.

For Aqua, the sun is now left of the observer as it's the afternoon (so the solar azimuth is negative). The satellite azimuths are now 80 or $-100°$ so the sensor is instead right and slightly in front of the observer on the left of the orbit but left and slightly behind them on the right. Thus we get large relative azimuths on the left and smaller ones on the right.

Second, AATSR. I've selected a section of orbit around my regression test to make the details easier to see. My comment in the code states that "azimuth (is) measured from the y-axis towards x and elevation above the x-y plane. . . the x- and y-axes point east and north, respectively."

```
[6]: compiler = "gfort"
     test = "AATSR"
     revision = 1965
     # 'in nadir view'
     with Swath(regression_filename(compiler, test, revision)) as pri_set, \
          Dataset(regression_filename(compiler, test, revision, "geo")) as geo_set:
         # We'll want to re-open this in a sec
         aatsr_l1b = pri_set.nc_files["pri"].Level1b_File

         fig, axes = plt.subplots(2, 5, figsize=(25, 11),
                                  subplot_kw=dict(projection=ccrs.PlateCarree()))
         for i in range(2):
             for ax, var in zip(axes[i], ("solzen", "satzen", "solaz", "sataz",␣
      ↪"relazi")):
                 ax.coastlines("10m")
                 im = pri_set.map(ax, geo_set[var][i,21000:22000],
                                  slices=(slice(21000,22000), slice(None)))
                 fig.colorbar(im, ax=ax)
                 if i == 0:
                     ax.set_title(var)
         fig.text(0., 0.75, "Nadir", rotation="vertical")
         fig.text(0., 0.25, "Oblique", rotation="vertical")
     fig.tight_layout()
     plt.show()
```

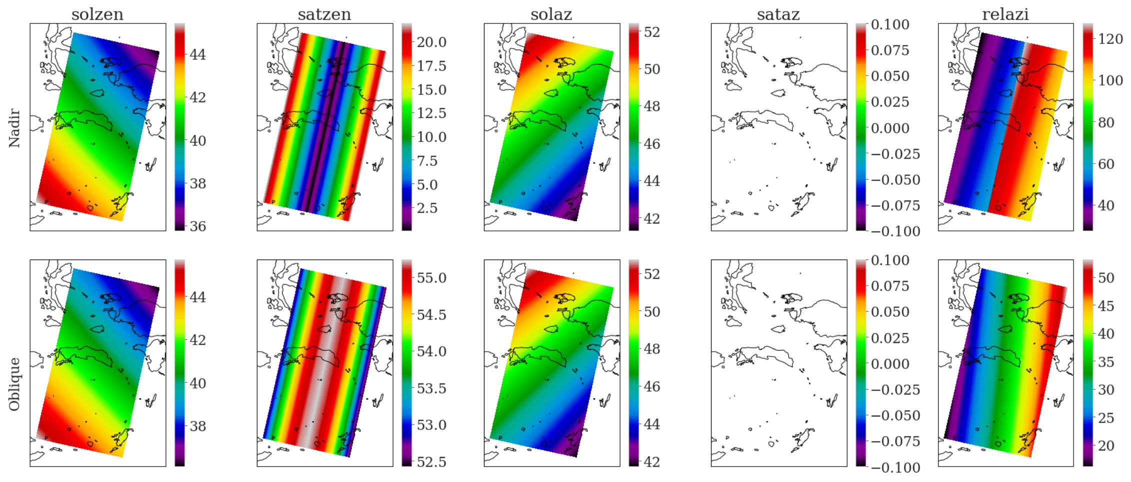


Figure 4: Solar and satellite zenith and azimuth angles from AATSR at 00:23 on 20/06/2008 from the nadir and oblique views (top and bottom, respectively). The satellite is moving from the top

to bottom of the image.

We are reminded that `sataz` is an output variable Simon added in, but is not filled for all instruments. If we want that, we have to read the value out of the L1 file (though that has interpolation errors near the middle that we fix inside ORAC).

Relative azimuth is calculated here from

```
phi = saz[k] - iaz[k];
if (phi < 0.0) phi = -phi;
if (phi > 180.0)
     raz[k] = 360.0 - phi;
else
     raz[k] = phi;
```

which was changed by me here.

[7]:
```
from epr import Product

product = Product(aatsr_l1b)
view_az0 = product.get_band("view_azimuth_nadir").read_as_array()
sun_az0 = product.get_band("sun_azimuth_nadir").read_as_array()
view_az1 = product.get_band("view_azimuth_fward").read_as_array()
sun_az1 = product.get_band("sun_azimuth_fward").read_as_array()

with Swath(regression_filename("gfort", "AATSR", 1965)) as dset:
    fig, axes = plt.subplots(2, 3, figsize=(12, 8),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))

    axes[0,0].coastlines("10m")
    im0 = dset.map(axes[0,0], sun_az0[21000:22000],
                   slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im0, ax=axes[0,0])
    axes[0,0].set_title("Sun azimuth")

    axes[0,1].coastlines("10m")
    im1 = dset.map(axes[0,1], view_az0[21000:22000],
                   slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im1, ax=axes[0,1])
    axes[0,1].set_title("View azimuth")

    axes[0,2].coastlines("10m")
    im2 = dset.map(axes[0,2], np.abs(sun_az0[21000:22000]-view_az0[21000:22000]),
                   slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im2, ax=axes[0,2])
    axes[0,2].set_title("Sun - view azimuth")

    axes[1,0].coastlines("10m")
    im0 = dset.map(axes[1,0], sun_az1[21000:22000],
                  slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im0, ax=axes[1,0])

    axes[1,1].coastlines("10m")
    im1 = dset.map(axes[1,1], view_az1[21000:22000],
                  slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im1, ax=axes[1,1])

    axes[1,2].coastlines("10m")
    im2 = dset.map(axes[1,2], np.abs(sun_az1[21000:22000]-view_az1[21000:22000]),
                  slices=(slice(21000,22000), slice(None)))
    fig.colorbar(im2, ax=axes[1,2])

fig.text(0., 0.75, "Nadir", rotation="vertical")
fig.text(0., 0.25, "Oblique", rotation="vertical")
fig.tight_layout()
plt.show()
```

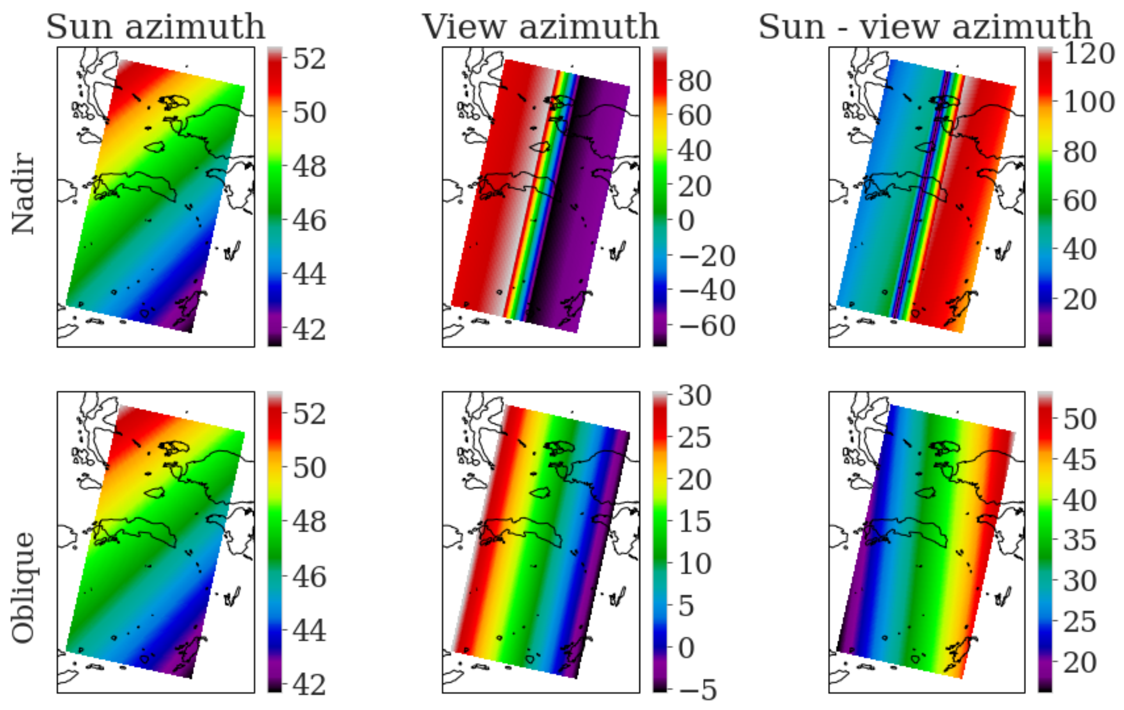


Figure 5: Solar and satellite azimuth angles reported on the sensor grid within the Level 1 file from Fig. 4. These incorrectly interpolate the data near the satellite track, which is fixed within ORAC but not output.

Solar azimuth is around 45°, which means the sun is to the right and ahead of an observer on the ground looking north as it's morning in June just below the equator.

In the nadir view, the satellite (view) azimuth is around 80° on the left side of the image and around $-60°$ on the right side. That means the satellite is right and a little in front of someone on the left side of the image and the satellite is left and ahead of someone on the right side of the image. The observer ends up in front of the sensor on both sides, unlike MODIS, because AATSR has a conical scan, as shown below.

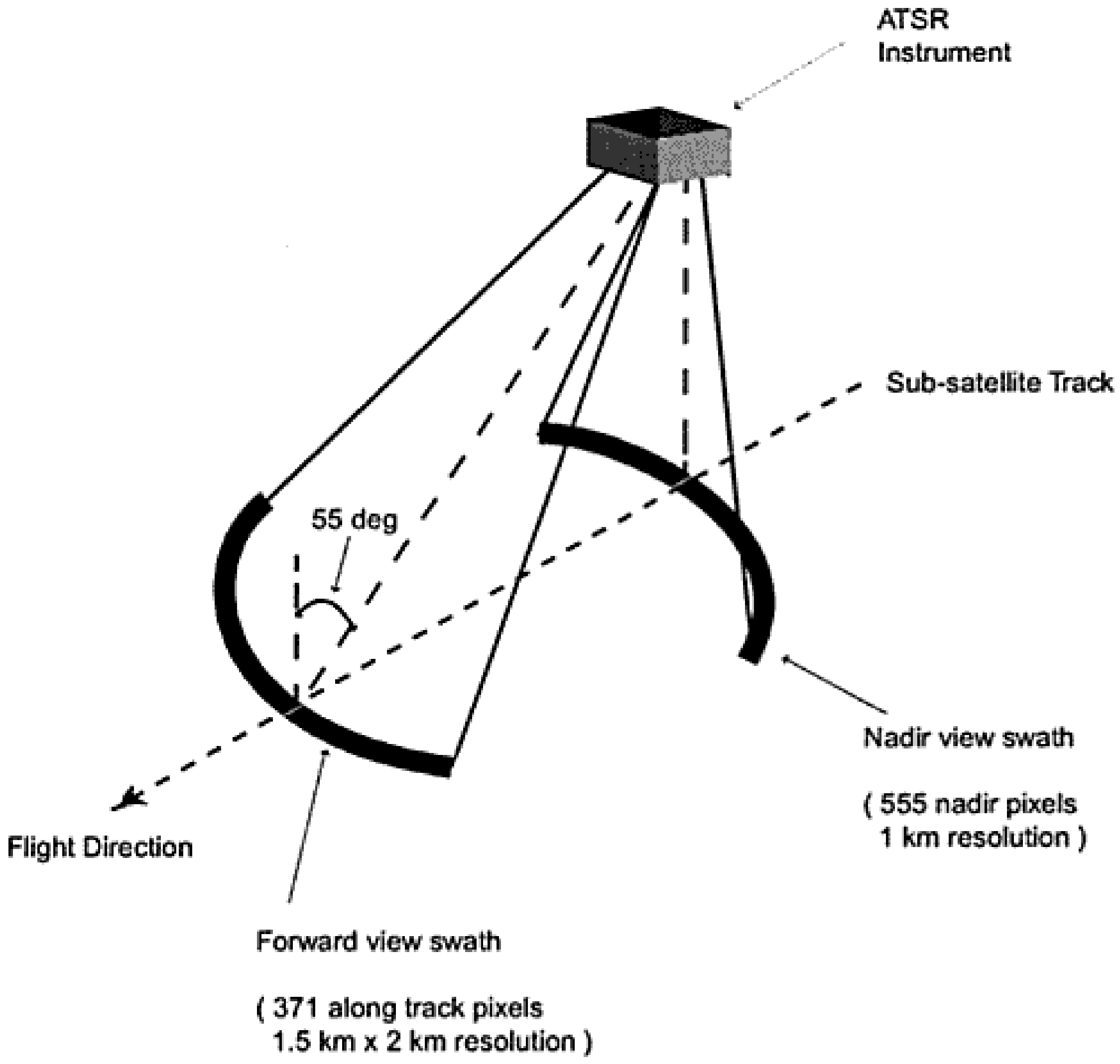


Figure 6: A sketch of the scanning geometry of AATSR.

Thus, we get larger relative azimuth angles on the right side of the orbit as the sun and satellite are on different sides of the observer while they are on the same side for an observer on the left of the image. Hence, I agree with the values shown in Fig. 5.

In the oblique view, the view azimuth is between 30 and $-5°$, putting the satellite roughly in front of our observer. Hence, the relative azimuth varies less in this view, being minimal on the left side of the image where the sun and satellite are on the same side.

Third, SLSTR. As best I can tell, it follows the EUMETSAT standard and so should define azimuth in the same direction as AATSR.

I've made some changes to this code. Before starting this document, the relative azimuth was calculated here by

```
! Rescale zens + azis into correct format
where(imager_angles%solazi(:,:,view) .ne. sreal_fill_value .and. &
      imager_angles%satazi(:,:,view) .ne. sreal_fill_value)
   ! This line converts sol+sat azi to relazi
   imager_angles%relazi(:,:,view) = abs(imager_angles%solazi(:,:,view)-&
                                        imager_angles%satazi(:,:,view))
end where
where (imager_angles%relazi(:,:,view) .gt. 180.)
   imager_angles%relazi(:,:,view) = 360. - imager_angles%relazi(:,:,view)
end where
imager_angles%relazi(:,:,view) = abs(180. - imager_angles%relazi(:,:,view))
where (imager_angles%relazi(:,:,view) .lt. 0. .and. &
       imager_angles%relazi(:,:,view) .ne. sreal_fill_value )
   imager_angles%relazi(:,:,view) = 0. - imager_angles%relazi(:,:,view)
end where
```

I have deleted the last two clauses as redundant.

[8]:
```python
compiler = "gfort"

test = "DAYSLSTRA"
revision = 1971

with Swath(regression_filename(compiler, test, revision, local=True)) as␣
 ↪pri_set, \
     Dataset(regression_filename(compiler, test, revision, "geo", local=True))␣
 ↪as geo_set:
    fig, axes = plt.subplots(2, 5, figsize=(25, 8),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))
    for i in range(2):
        for ax, var in zip(axes[i], ("solzen", "satzen", "solaz", "sataz",␣
 ↪"relazi")):
            ax.coastlines("10m")
            im = pri_set.map(ax, geo_set[var][i])
            fig.colorbar(im, ax=ax)
            ax.set_title(var)
fig.text(0., 0.75, "Nadir", rotation="vertical")
fig.text(0., 0.25, "Oblique", rotation="vertical")
fig.tight_layout()
plt.show()
```

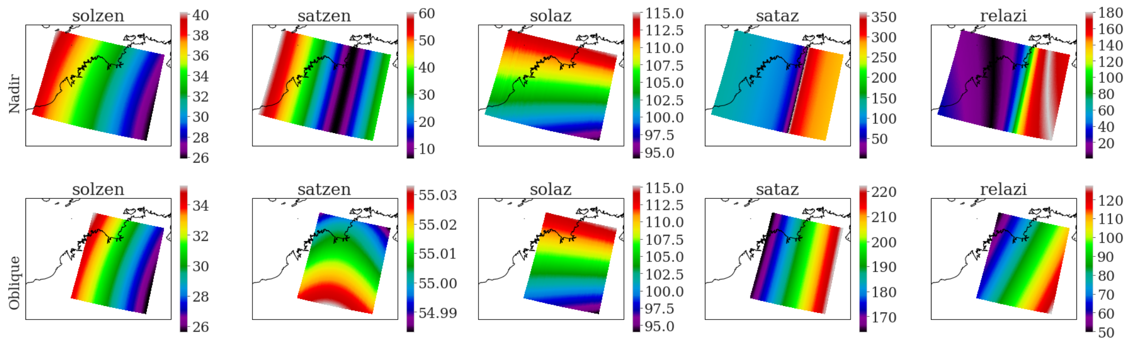


Figure 7: Solar and satellite angles over northern Australia at 01:10 on 21/12/2018 from the nadir and oblique views of SLSTR-A (top and bottom, respectively). The satellite is moving from the top to bottom of the image.

It's a December morning in northern Australia, so the sun is right of and behind a north-facing observer (from a solar azimuth of ~100°).

For the nadir view, with an azimuth of ~120°, the satellite is behind and right of an observer on the left side of the image and, with an azimuth of ~275°, is in front of and left of an observer on the right side of the image. For the oblique view, the azimuth is $165 - 225°$, which puts the satellite behind our observer. Values smaller than 180° are on the left side of the swath, as desired for a satellite looking backwards as it moves down the image.

Thus, in the nadir we have small relative azimuths on the left side of the image where the satellite and sun are in a similar part of the sky. On the right side, we have the satellite and sun on opposite sides of the observer, giving large relative azimuth. In the oblique view, there is a less stark change in relative azimuth as both satellite and sun are vaguely behind the observer.

It should also be noted here that MODIS defined satellite azimuth on $[-180, 180]$ but SLSTR is defined over $[0, 360]$.

[9]:
```
test = "DAYSLSTRB"

with Swath(regression_filename(compiler, test, revision, local=True)) as␣
 ↪pri_set, \
     Dataset(regression_filename(compiler, test, revision, "geo", local=True))␣
 ↪as geo_set:
    fig, axes = plt.subplots(2, 5, figsize=(25, 8),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))
    for i in range(2):
        for ax, var in zip(axes[i], ("solzen", "satzen", "solaz", "sataz",␣
 ↪"relazi")):
            ax.coastlines("10m")
            im = pri_set.map(ax, geo_set[var][i])
            fig.colorbar(im, ax=ax)
            ax.set_title(var)
fig.text(0., 0.75, "Nadir", rotation="vertical")
fig.text(0., 0.25, "Oblique", rotation="vertical")
fig.tight_layout()
plt.show()
```

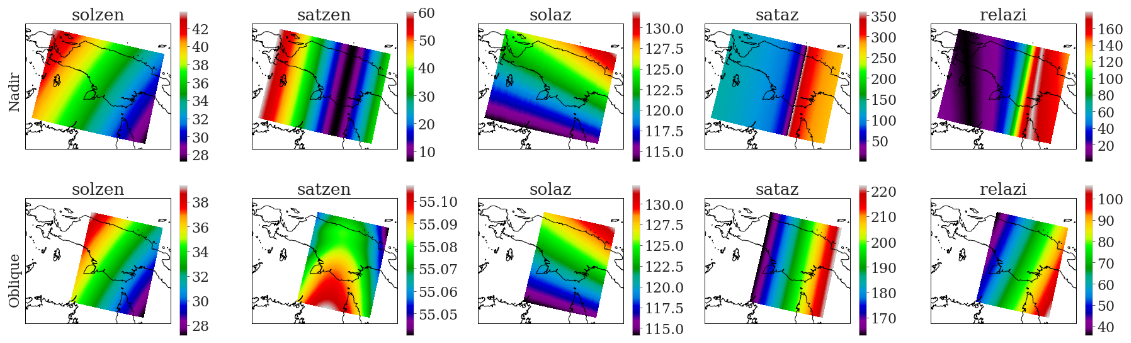


Figure 8: Solar and satellite angles over northern Australia at 00:27 on 21/12/2018 from the nadir and oblique views of SLSTR-B (top and bottom, respectively). The satellite is moving from the top to bottom of the image.

Just checking SLSTR-B isn't different. These plots are quite similar to SLSTR-A.

Fourth, AVHRR (specifically NOAA-18, an afternoon satellite). This returns to June 20th but we are now looking off the east coast of Japan. Angles are determined here incorrectly, though the comments correctly describe what the code does. I replaced that with,

```
where ( temp2 .ne. sreal_fill_value .AND. temp .ne. sreal_fill_value )
   imager_angles%relazi(:,:,1) = abs( temp2 - temp )
   imager_angles%satazi(:,:,1) = temp
   where ( imager_angles%relazi(:,:,1) .gt. 180. )
      imager_angles%relazi(:,:,1) = 360. - imager_angles%relazi(:,:,1)
   end where
end where
```

[10]:
```
compiler = "gfort"
test = "AVHRR"
revision = 1971

with Swath(regression_filename(compiler, test, revision, local=True),␣
 ↪central_longitude=180) as pri_set, \
     Dataset(regression_filename(compiler, test, revision, "geo", local=True))␣
 ↪as geo_set:
    fig, axes = plt.subplots(1, 5, figsize=(20, 4),
                             subplot_kw=dict(projection=ccrs.PlateCarree()))
    for ax, var in zip(axes, ("solzen", "satzen", "solaz", "sataz", "relazi")):
        ax.coastlines("10m")
        im = pri_set.map(ax, geo_set[var][0,12000:13000],
                         slices=(slice(12000,13000), slice(None)))
        fig.colorbar(im, ax=ax)
        ax.set_title(var)
fig.tight_layout()
plt.show()
```

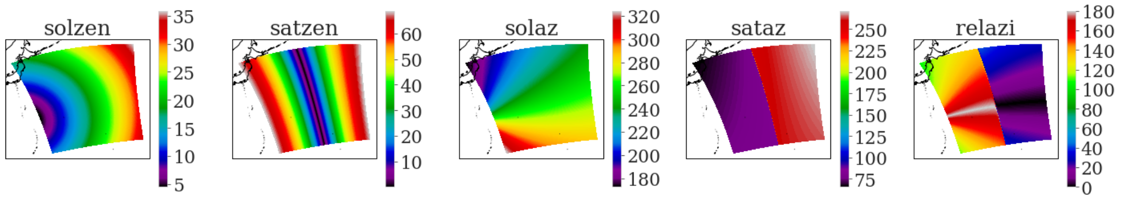


Figure 8: Solar and satellite angles from NOAA-18 at 00:50 on 20/06/2008. The satellite is moving from the bottom to the top of the image.

The Tropic of Cancer cuts through the bottom of the image. Thus, there is a solar azimuth of around 180° at the top-left of the image, where it is near local solar noon and the sun is behind the observer. At the bottom of the image, solar azimuth is $> 320°$ as it is early afternoon but the sun is in front of the observer as we are below the Tropic at the height of summer in the northern hemisphere.

For the satellite azimuth, the sensor is right and slightly ahead of our observer on the left of the image and left and slightly behind them on the right. With my changes, the relative azimuth is now correctly rendered: the relative azimuth is around 100° in the top-left of the image, increases to 180° at the Tropic, and decreases to back around 100° at the bottom-left. On the right side of the image, the relative azimuth is small as the sun and satellite are on the same side of the observer but there is still variation down the image, with a minima at the Tropic.

Fifth, SEVIRI. Simon updated the relative azimuth calculation here to be,

```
where(imager_angles%solazi(startx:,:,1) .ne. sreal_fill_value .and. &
      imager_angles%satazi(startx:,:,1) .ne. sreal_fill_value)
   imager_angles%solazi(:,:,1) = imager_angles%solazi(startx:,:,1) - 180.
   where(imager_angles%solazi(:,:,1) .lt. 0.)
      imager_angles%solazi(:,:,1) = imager_angles%solazi(:,:,1) + 360.
   end where
   imager_angles%relazi(:,:,1) = abs(imager_angles%satazi(startx:,:,1) - &
                                     imager_angles%solazi(startx:,:,1))
   where (imager_angles%relazi(:,:,1) .gt. 180.)
      imager_angles%relazi(:,:,1) = 360. - imager_angles%relazi(:,:,1)
   end where
end where
```

SEVIRI fields can't be plot using my Mappable class as there are NaNs in the SEVIRI lat/lon fields, so the interpolation to cell edges doesn't work. However, there's a `cartopy` class specificaly for this job.

```
[11]: compiler = "gfort"
      test = "SEVIRI"
      revision = 1965

      with Dataset(regression_filename(compiler, test, revision, "geo")) as geo_set:
          fig, axes = plt.subplots(
              1, 5, figsize=(20, 4),
              subplot_kw=dict(projection=ccrs.Geostationary(satellite_height=35786000))
          ) # Use the Geostationary map projection
          for ax, var in zip(axes, ("solzen", "satzen", "solaz", "sataz", "relazi")):
              ax.coastlines("10m")
              im = ax.imshow(
                  geo_set[var][0],extent=(5500000, -5500000, -5500000, 5500000),
                  origin="lower right"
              ) # Put in approximate values for the extremes of the SEVIRI swath
              fig.colorbar(im, ax=ax)
              ax.set_title(var)
              ax.set_global() # Force a sensible map orientation
      fig.tight_layout()
      plt.show()
```

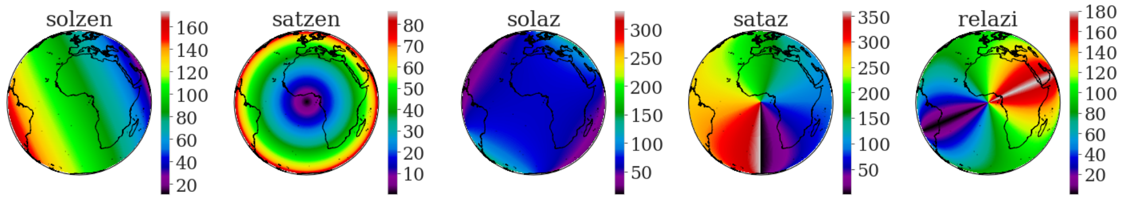


Figure 9: Solar and satellite angles from SEVIRI at 05:57 on 20/06/2017 using the pre-defined geometry files. The satellite is at the centre of the image and the sun is over the right side of the image.

The zenith angles place the sun and satellite in the right places. The solar azimuth is usually near 90° as the sun is typically east of our observer, but there are more extreme values at the edges of the image. The satellite azimuth is correctly 0° due south of the satellite and 180° due north of it. It is *incorrectly* 90° right of it and 270° left of it. However, the relative azimuth is correct: 0° left of the satellite and 180° along the line between the satellite and sun on the right of the image.

```
[12]: compiler = "gfort"
      test = "SEVIRI"
      revision = 1971

      with Dataset(regression_filename(compiler, test, revision, "geo", local=True))␣
       ↪as geo_set:
          fig, axes = plt.subplots(
              1, 5, figsize=(20, 4),
        subplot_kw=dict(projection=ccrs.Geostationary(satellite_height=35786000))
    ) # Use the Geostationary map projection
    for ax, var in zip(axes, ("solzen", "satzen", "solaz", "sataz", "relazi")):
        ax.coastlines("10m")
        im = ax.imshow(
            geo_set[var][0],extent=(5500000, -5500000, -5500000, 5500000),
            origin="lower right"
        ) # Put in approximate values for the extremes of the SEVIRI swath
        fig.colorbar(im, ax=ax)
        ax.set_title(var)
        ax.set_global() # Force a sensible map orientation
fig.tight_layout()
plt.show()
```

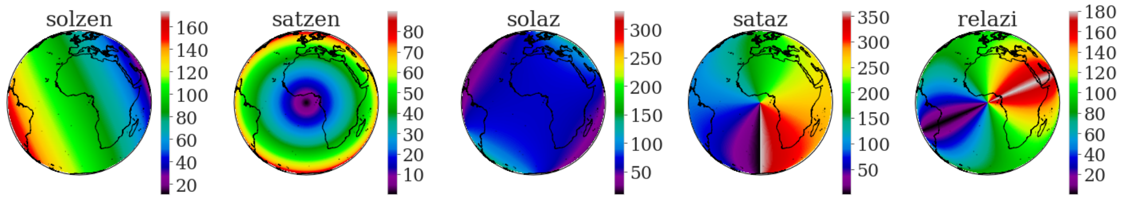


Figure 10: As Fig. 9, but using online calculation of the SEVIRI angles.

Calculating the angles on the fly (e.g. within ORAC using code Simon has adjusted) fixes the satellite azimuth without changing the relative azimuth, so this code is fine. I will update the file in which I store offline calculations of these angles.

Supported sensors not yet shown here: GOES, AGRI, Himawari, VIIRS

In `cloud_typing_pavolonis()`, the relative azimuth is only used in the calculation of the glint angle here,

```
! In PATMOS sunglint calculation:
if (imager_angles%solzen(i,j,cview) .ne. sreal_fill_value .and. &
    imager_angles%satzen(i,j,cview) .ne. sreal_fill_value .and. &
    imager_angles%RELAZI(i,j,cview) .ne. sreal_fill_value) then

   glint_angle = cos(imager_angles%solzen(i,j,cview) * d2r) * &
                 cos(imager_angles%satzen(i,j,cview) * d2r) + &
                 sin(imager_angles%solzen(i,j,cview) * d2r) * &
                 sin(imager_angles%satzen(i,j,cview) * d2r) * &
                 cos(imager_angles%RELAZI(i,j,cview) * d2r)

   glint_angle = max(-1.0, min(glint_angle, 1.0))
   glint_angle = acos(glint_angle) / d2r
```

```
else
   glint_angle = sreal_fill_value
end if
```

This is calculating the angle between the incident and reflected rays,

$$\cos\Omega = \cos\theta_v \cos\theta_s + \sin\theta_v \sin\theta_s \cos\Delta\phi.$$

A detailed explanation of that equation will accompany the evaluation of the surface reflectance, which will show that this code is using the correct convention.

The implementation of surface reflectance and RTTOV will be explored in separate documents.

```
[ ]:
```