SARTA ANALYTIC JACOBIANS

Sergio DeSouza-Machado[1] and Larrabee Strow[1,2]
[1] Joint Center For Earth Systems Technology, University of Maryland Baltimore County
[2] Department of Physics, University of Maryland Baltimore County
contact : sergio@umbc.edu

1 Introduction

SARTA is a fast Radiative Transfer code with TwoSlab Cloud scattering ability [De Souza-Machado et al., 2018] that has been extensively tested for clear sky accuracy against *e.g.,*kCARTA [De Souza-Machado et al., 2020] and clouds [Aumann et al., 2018, 2023, De Souza-Machado et al., 2018].

It currently takes about 4 minutes to generate radiances for one granule of AIRS L1C data (2645 channels × 12150 FOVS), given input atmospheric profiles (thermodynamic and cloud) supplied by *e.g.,*radiosondes or Numerical Weather Prediction (NWP) model fields that are put into a 100 layer atmosphere model.

This is more than adequate for processing observations for comparison purposes (given that line by line codes such as kCARTA take about 40 seconds to process one spectrum!). But if used for retrievals where estimates of jacobians are needed, the times become very large (for example would need to run it 100 times for T(z) jacobians, 100 times for WV(z) jacobians and another 100 times for O3(z) jacobians, or about 20 hours for all channels; more if you need *e.g.,*CH4, N2o etc jacobians).

For these purposes, an analytic SARTA jacobian code has been written. This document outlines the changes to the SARTA code, and the ideas behind propagating the thermodynamic (temperature and constituent gases) analytic derivatives at layer j to the Top of Atmosphere (TOA); cloud jacobians still need to be done by finite difference.

2 SARTA original flow

Figure 1 shows the "regular" SARTA flow diagram. After opening an rtp file and reading in the 7 sets of coefficients, it then generates all the predictors for each of the 7 breakout sets (ycalparX, $X = 1,7$) and then produces the ODs for all 7 sets (ycaltX_od, $X = 1,7$). After that it loops over each required channel to do the radiative transfer, pulling in the necessary ODs for the ycaltX_od as needed. We remind the reader the scattering radiative transfer is a weighted sum over 4 radiance streams : clear, cloud 1 only,cloud2 only and cloud1,2

3 SARTA analytic jacobian flow

Figure 2 shows the "analytic jacobian" SARTA flow diagram. The main difference is that the loops over channels start at the (ycaltX_od(i), $X = 1,7$), continuing with the radiative transfer.

4 The Analytic Derivatives of the Predictors

The SARTA clear sky details are encapsulated in *e.g.,*[Strow et al., 2003] where based on coefficients C_j and predictors P_j based on the atmospheric profiles and view angles *T(z), WV(z), O3(z), sec(θ)* so one can form effective optical depths at each layer L

$$\tau(i,L) = \sum_{j=1}^{J} C_j P_j(T(z), WV(z), O3(z), \theta)$$

with J typically being less than 10; the ODs are computed *at each layer L = 1,100* (where L = 1 = TOA and $L = N$ is GND) for *e.g.,*variable gases WV, CO2, O3, N2O, CO, CH4, SO2, HNO3 and the "rest of the uniform fixed gases" which include O2, N2 etc, using the atmospheric profiles *T(z),WV(z),O3(z)* and view angle information θ.

This formulation makes it evident that the jacobian is '"simply" related to the derivative of τ with respect to any thermodynamic variable X (where X can be temperature, water vapor amount, ozone amount, etc), given by

$$\frac{\partial\tau(i,L)}{\partial X_L} = \sum_{j=1}^{J} C_j \frac{\partial P_j(T(z), WV(z), O3(z), sec(\theta))}{\partial X_L} \tag{1}$$

and so can "easily" be calculated, using the same coefficients $C(j)$. *Note these derivatives are for optical depth at angle θ*

4.1 "Simply" the derivative

We will typically denote the $T, WV, O3$ derivatives as *_T,_1,_3.* Most of the predictors simply depend on layer L, and the derivatives of $P(L)$ can indeed be obtained very simply by using the product rule, for example as seen in Table 1

Table 1: Easy derivatives

PREDICTOR NAME *P(L)*	THERMODYNAMIC + CONSTITUENT DEPENDENCE	$\frac{\partial P}{\partial T_l}$ P(L)_T	$\frac{\partial P}{\partial WV_l}$ P(L)_1	$\frac{\partial P}{\partial O3_l}$ P(L)_3
TR	T(L)/TREF(L)	$\frac{1}{TREF(L)}$	0	0
DT	PTEMP(L) - RTEMP(L)	1	0	0
A_W	PWAMNT(L)/RWAMNT(L)	0	1/RWAMNT(L)	0
A_O	POAMNT(L)/ROAMNT(L)	0	0	1/ROAMNT(L)
TJUNKS	TR*TR	2*TR*TR_T	0	0
FPRED1(1)	SECANG(L)*TJUNKS	SECANG(L)*TJUNKS_T	0	0
OPRED1(3)	SECANG(L)*A_O*DT	SECANG(L)*A_O*DT_T = SECANG(L)*A_O	0	SECANG(L)*DT*A_O_3 = SECANG(L)*DT/ROAMNT(L)

4.2 Not so "Simply" the derivative

Of course things are not always so simple. This is especially so in the case of the predictor depending on "all the layers above layer L" $P(L, L-1, L-2, ...1)$, with layer $L = 1$ being the TOA. For example

$$\begin{array}{rcl} PDP(L) & = & PRES(L)(PRES(L) - PRES(L-1)) \\ WZREF(L) & = & \sum_{j=1}^{L} PDP(J)WAMNT(J)^{ref} \\ WZ(L) & = & \sum_{j=1}^{L} PDP(J)WAMNT(J) \\ AZ\_W(L) & = & \frac{WZ(L)}{WZREF(L)} \end{array}$$

So technically if you perturb layer L you also "perturb" all layers above it. Which means if the amount of water vapor in layer L is increased by α, so is that of all the layers above it $WZ(L) \rightarrow WZ(L)(1+\alpha)$, while of course the reference amount(s) stay unchanged. This means the original amounts (denoted by "0" below)

$$AZ\_W0(L) = \frac{WZ0(L)}{WZREF(L)} = \frac{\sum_{j=1}^{L} PDP(j)WAMNT(j)}{\sum_{j=1}^{L} PDP(j)WAMNT(j)^{ref}}$$

are changed to amounts X below

$$AZ\_WX(L) = \frac{WZX(L)}{WZREF(L)} = \frac{\sum_{j=1}^{L} PDP(j)WAMNT(j)(1+\alpha)}{\sum_{j=1}^{L} PDP(j)WAMNT(j)^{ref}}$$

and so the change in $AZ\_W(L)$ which is the cumulative amount at layer L is now

$$\begin{aligned}
\frac{\Delta(AZ\_W(L))}{\Delta(GasAmount(L))} &= \frac{AZ\_WX(L)-AZ\_W0(L)}{\alpha WAMNT(L)} \\
&= \frac{\alpha \sum PDP(j)WAMNT(j)}{\sum PDP(j)WAMNT(j)^{ref}} \frac{1}{\alpha WAMNT(L)} \\
\frac{\partial AZ\_W(L)}{\partial WAMNT(L)} &= \frac{AZ\_W(L)}{WAMNT(L)}
\end{aligned}$$

We can now use these ideas to get the derivatives of the ‘predictors which depend on these ‘layer above” predictors! for example as seen in Table 2

Table 2: Not so Easy derivatives

PREDICTOR NAME *P(L)*	THERMODYNAMIC + CONSTITUENT DEPENDENCE	$\frac{\partial P}{\partial T_l}$ P(L)_T	$\frac{\partial P}{\partial WV_l}$ P(L)_1	$\frac{\partial P}{\partial O3_l}$ P(L)_3
A_W	PWAMNT(L)/RWAMNT(L)	0	1/RWAMNT(L)	0
WJUNKA	SECANG(L)*A_W	0	SECANG(L)*A_W_1	0
WJUNKZ	WJUNKA*A_W/AZ_W	0	WJUNKA_1*A_W/AZ_W + WJUNKA*(AZ_W*A_W_1 - A_W*AZ_W_1)/AZ_W/AZ_W	0

4.3 OPTRAN

The above coefficients and predictors are based on a pressure scale (ie given index L you always know the pressure $P(L)$; it is the gas amount $Q(L)$ and temperature $T(L)$ which must be given to you by *e.g.*,a sonde measurement or a NWP model, in order for you to estimate the gas optical depth $OD(L) = OD(q(L), T(L), p(L)) = OD(q(L(p)), T(L(p)))$ ie temperature and gas amounts are predictors.

The gas amounts and mixing ratios of variable gases such as WV vary by orders of magnitude over the 0-80 km of a typical atmosphere model. The OPTRAN [McMillin et al., 1995] methodology estimates optical depths using layer to space gas amount as the grid, with pressure being a predictor $OD(L) = OD(Q(A), T(A), p(A)) = OD(T(A(a)), P(A(a)))$ ie temperature and pressures are predictors. SO one has to interpolate p, T and the weighted p, T onto these grids, do the OPTRAN fast model coefficient × predictor magic, then interpolate back ont other usual AIRS 100 layers pressure grid.

Which means the jacobians need to involve all these stages ... complicating things quite a bit. But the ideas remain the same; some are simple derivatives, most are “layer above” derivatives. Once they are computed, they need to be used in differentiating the linear weighting terms that put the P(L) onto the A(a) grid. Once that is done, the optical depths need to be interpolated onto the layer-to-space absorption grid; so once again the derivatives need to be computed there. Quite a mess.

5 Propagating analytic derivatives from layer j to TOA

The one layer clear sky equation governing radiative transfer ([De Souza-Machado et al., 2020, Goody and Yung, 1989, Liou, 1980]) is given by

$$\mu(\theta)\frac{dr(\nu)}{d\tau_0} = -r(\nu) + B(\nu, T) \tag{2}$$

where μ is the cosine of the zenith angle θ, $B(T)$ is the Planck function at layer temperature T and $r(\nu)$ is the Planck radiance at wavenumber ν. Finally τ_0 is the nadir optical depth; the optical depth at angle

θ is given by

$$\tau(\theta) = \frac{\tau_0}{cos(\theta)} = \frac{1}{\mu}\tau_0 \tag{3}$$

At the top of layer i, the solution to is

$$r_i(\nu) = r_{i-1}(\nu)e^{-\frac{\tau_0(i,\nu)}{\mu}} + B(\nu, T_i)(1 - e^{-\frac{\tau_0(i,\nu)}{\mu}}) \tag{4}$$

where r_{i-1} is the radiation incident at the bottom of this layer from the top of the previous layer $i-1$ (or if $i = 1$, the surface emission $\epsilon(\nu)B(\nu, T_S)$).

Propagating this up through many layers N then gives the radiance at the TOA

$$r_N = \epsilon(\nu)B(\nu, T_S)e^{-\sum_i=1^N \frac{\tau_0(i,\nu)}{\mu}} + \sum_i^N B(\nu, T_i)(1 - e^{-\frac{\tau_0(i,\nu)}{\mu}})e^{-\sum_{j=i+1}^N \frac{\tau_0(j,\nu)}{\mu}} \tag{5}$$

5.1 Derivatives for one layer

We are interested in temperature and gas constituent derivatives for a clear sky; Remember optical depth depends on gas amount q and temperature T (since layer i implicitly also gives a pressure dependence); Then for one layer we can differentiate Equation 4 with respect to temperature T and gas amount q to get (after some algebra, and dropping layer index i and wavenumber dependence ν from the equations)

$$\frac{\partial r}{\partial T} = \left(\mu \frac{\partial r}{\partial \tau_0}\right)\left(\frac{1}{\mu}\frac{\partial \tau_0}{\partial T}\right) + \frac{\partial B(T)}{\partial T}(1 - e^{\frac{\tau_0}{\mu}}) = A(r)E(T) + C(T) \tag{6}$$

$$\frac{\partial r}{\partial q} = \left(\mu \frac{\partial r}{\partial \tau_0}\right)\left(\frac{1}{\mu}\frac{\partial \tau_0}{\partial q}\right) = A(r)G(q) \tag{7}$$

Looks complicated but we know everything (also see *e.g.*,Equation 8 in [Liu et al., 2006]) :

- the radiance derivatives wrt OD $A(r) = \mu \frac{\partial r}{\partial \tau_0}$ from the right hand side of Equation 2 above
- the right hand side of Equation 4 above says what top-of-layer radiance to use in $A(r)$
- the OD partial derivatives wrt thermodynamic variables $E(T) = \frac{1}{\mu}\frac{\partial \tau_0}{\partial T}, G(q) = \frac{1}{\mu}\frac{\partial \tau_0}{\partial q}$ come from Equation 1 above!
- it is trivial to compute $C(T) = \frac{\partial B(T)}{\partial T}(1 - e^{\frac{\tau_0}{\mu}})$

5.2 Derivatives for one layer at TOA

But you need to propagate it up! Remember the radiative transfer is done iteratively, or if you like recursively; since the radiation from surface goes through layer 1, then to layer 2 and so on till layer N at TOA. Which means

$$r_N(\nu) = r_{N-1}(\nu)e^{-\frac{\tau_0(N,\nu)}{\mu}} + B(\nu, T_N)(1 - e^{-\frac{\tau_0(N,\nu)}{\mu}}) \tag{8}$$

But since

$$r_{N-1}(\nu) = r_{N-2}(\nu)e^{-\frac{\tau_0(N-1,\nu)}{\mu}} + B(\nu, T_{N-1})(1 - e^{-\frac{\tau_0(N-1,\nu)}{\mu}}) \tag{9}$$

that means

$$r_N(\nu) = r_{N-1}(\nu, r_{N_2}(\nu))e^{-\frac{\tau_0(N,\nu)}{\mu}} + B(\nu, T_N)(1 - e^{-\frac{\tau_0(N,\nu)}{\mu}}) \tag{10}$$

and so on to the layer you are interested in J

$$r_N(\nu) = r_{N-1}(\nu, r_{N_2}(\nu, r_{N_2}(\nu(....(r_J(\nu, T_j, q_j))))))e^{-\frac{\tau_0(N,\nu)}{\mu}} + B(\nu, T_N)(1 - e^{-\frac{\tau_0(N,\nu)}{\mu}}) \tag{11}$$

Recall that radiances at the bottom of layer i are attenuated by $e^{-\tau_0(i,\nu)\mu}$ as they propagate to the top of the layer. This means differentiating Equation 11 gives an attenuation from every layer above that ie from the top of the layer J to the TOA

$$\frac{\partial r_N}{\partial X_J} = \frac{\partial r_N}{\partial r_{N-1}}\frac{\partial r_{N-1}}{\partial r_{N-2}}\frac{\partial r_{N-2}}{\partial r_{N-3}}...\frac{\partial r_{J+1}}{\partial r_J}\frac{\partial r_J}{\partial X_J} \tag{12}$$

$$\frac{\partial r_N}{\partial X_J} = e^{-\frac{\tau_0(N,\nu)}{\mu}}e^{-\frac{\tau_0(N-1,\nu)}{\mu}}e^{-\frac{\tau_0(N-2,\nu)}{\mu}}...e^{-\frac{\tau_0(J+1,\nu)}{\mu}}\frac{\partial r_J}{\partial X_J} \tag{13}$$

$$\frac{\partial r_N}{\partial X_J} = e^{-\sum_{k=J+1}^{N}\frac{\tau_0(k,\nu)}{\mu}}\frac{\partial r_J}{\partial X_J} \tag{14}$$

where we know $\frac{\partial r_J}{\partial X_J}$ from the one layer derivatives in Equations 6,7 above! Again, also See Equation 8 in [Liu et al., 2006]!!!

6 Various : surface and background thermal and solar contribution and NLTE

6.1 Surface

The surface emission term, seen at TOA, is given by

$$r_{surface}(\nu) = \epsilon(\nu)B(\nu, T_S)e^{-\sum_1^N\frac{\tau_0(\nu,i)}{\mu}}$$

which means the derivative at the TOA is given by

$$\frac{\partial r_{surface}(\nu)}{\partial T_s} = \epsilon(\nu)\frac{\partial B(\nu, T_S)}{\partial T}e^{-\sum_1^N\frac{\tau_0(\nu,i)}{\mu}}$$

6.2 Background thermal

This is technically complicated, but lets assume we have calculated the term propagating downwards and have it as $r_{background}(\nu)$ just as it is incident downwards at the surface. It is then reflected and propagates to the TOA, attenuated at each layer as it goes up. So the term at the TOA is

$$r_{reflected\_thermal(\nu)} = \rho(\nu)r_{background}(\nu)e^{-\sum_1^N\frac{\tau_0(\nu,i)}{\mu}}$$

where typically $\rho(\nu) = (1-\epsilon(\nu))/\pi$ (Lambertian reflectance). The derivatives at each layer are then added on to the jacobians above, given by

$$\frac{\partial r_{reflected\_thermal(\nu)}}{\partial X_J} = -\frac{1}{\mu}\tau_0(\nu, J)\frac{\partial \tau_0(\nu, J)}{\partial X_J}r_{reflected\_thermal(\nu)}$$

6.3 Solar

This is technically complicated, but lets assume we have calculated the solar term propagating downwards and have it as $r_{solar}(\nu)$ just as it is incident downwards at the surface. It is then reflected and propagates to the TOA, attenuated at each layer as it goes up. So the term at the TOA is

$$r_{reflected\_solar(\nu)} = \rho(\nu)r_{solar}(\nu, \theta_{sun})e^{-\sum_1^N\frac{\tau_0(\nu,i)}{\mu}}$$

where typically $\rho(\nu) = (1-\epsilon(\nu))/\pi$ (Lambertian reflectance). The derivatives at each layer are then added on to the jacobians above, given by

$$\frac{\partial r_{reflected\_solar(\nu)}}{\partial X_J} = -\frac{1}{\mu}\tau_0(\nu,J)\frac{\partial \tau_0(\nu,J)}{\partial X_J} r_{reflected\_solar(\nu)}$$

Note we have assume $\rho(\nu)$ is the same for reflected thermal and for reflected solar; this is not necessarily correct as eg sun glint would be almost mirror like.

6.4 NLTE

There is a correction term to the 4.3 um radiance due to NLTE in the upper atmosphere. It currently depends only on the temperature of the top 5 layers, the satzen and solzen angles, and CO_2 gas amount (or more accurately, deviation from 385 ppm). This is modeled as

$$\delta r_{NLTE}(\nu) = (\sum_{j}^{6} N_j(\nu) NLTE\_PRED(j))(N_7(co2ppm-385)+1)$$

The temperature dependence is only in predictor 5 via NLTE_PRED5 = 1/5(T1+T2+T3+T4+T5), so

$$\frac{\partial \delta r_{NLTE}(\nu)}{\partial T_i} = N_5(0.2)(1)(N_7(co2ppm-385)+1), i = 1,2,3,4,5$$

and can be trivially added on; similarly the CO_2 gas amount jacobian can be done but it is tiny (at least over about 50 ppmv changes).

7 Two slab clouds : weighted sum over four radiance streams

We use the PCLSAM parametrization for clouds and aerosols in SARTA and kCARTA (see [De Souza-Machado et al., 2018, 2020]). Here the cloud scattering effects are re-parametrized into an effective optical depth. It's not as accurate as *e.g.,*DISORT [Stamnes et al., 1988] but its fast and simple, and the accuracy can be improved [Tang et al., 2018]. All this means our final radiation is

- a weighted sum over clear, cloud1 fraction, cloud 2 fraction, cloud 1+2 overlap
- more importantly, clouds 1 and 2 go into our atmosphere effectively as two additional non-scattering "gases"; so their effects can be added in as an additional "fixed" gas.

All the above equations remain fundamentally unchanged, except we *e.g.,*increase the total OD of layer(s) $i1$ by that of cloud 1, and the total OD of layer(s) $i2$ by that of cloud 2, while doing the cloud1 only, cloud 2 only, and cloud 1+2 portions of the radiative transfer.

Then jacobians in our TwoSlab atmosphere, are simply done as above, but for each of the four streams; and then just as final radiance is the weighted sum of the four radiance streams, the final jacobian is the weighted sum of the four jacobians.

The rtp file allows us to have the following variables : (ctype1,cfrac1,cngwat1,csze1,cprtop1, cprbot1),(ctype2,cfrac2,cn cprbot2),(cfrac12). Based on these, assuming all the fractions are non-zero the code then defines *cx1 = c1 - c12, cx2 = c2 - c12, fclr = 1 - (c1x+c2x+c12) = 1 - (c1+c2-c12)*

Then we find the radiance as

$$\begin{aligned} r(\nu) &= fclr\ \ rclr(\nu) + c1x\ \ r1(\nu) + c2x\ \ r2(\nu) + c12\ \ r12(\nu) \\ &= (1-(c1+c2-c12))\ \ rclr(\nu) + (c1-c12)\ \ r1(\nu) + (c2-c12)\ \ r2(\nu) + c12\ \ r12(\nu) \end{aligned}$$

The PCLSAM effective nadir cloud optical depth [Chou et al., 1999] is given by

$$\begin{aligned}\tau_{cld}(\nu, sze, cng, \theta = 0) &= cng \times \left(\tau_{ext}^{1g/m2}(\nu, sze) - \frac{1}{2}\tau_{sca}^{1g/m2}(\nu, sze)(1 + g(\nu, sze))\right) \\ &= cng \times \tau^{1g/m2}(\nu, sze, \theta = 0) \\ &= cng \times f(\nu, sze)\end{aligned}$$

where cng is the cloud loading in g/m2, sze is effective particle diameter in um, $g(\nu)$ is the asymmetry and $\omega = \tau_{sca}(\nu)/\tau_{ext}(\nu) = (\tau_{ext}(\nu) - \tau_{abs}(\nu))/\tau_{ext}(\nu)$. The superscript $1g/m2$ means everything has been saved/normalized to this cloud loading.

This means for example

1. the nadir cloud optical depth is related to cloud optical depth at angle θ by

$$\begin{aligned}\tau_{cld}(\nu, sze, cng, \theta) &= \frac{1}{cos(\theta)}\tau_{cld}(\nu, sze, \theta = 0) \\ &= \frac{1}{\mu}\tau_{cld}(\nu, sze, \theta = 0)\end{aligned}$$

2. the derivative with respect to cloud amount (ignoring angles) is trivially

$$\begin{aligned}\frac{\partial \tau_{cld}}{\partial cng} &= \frac{\tau_{cld}(\nu)}{cng} \\ &= \tau^{1g/m2}(\nu, sze, \theta = 0) = f(\nu, sze)\end{aligned}$$

3. the derivative with respect to cloud particle size is

$$\begin{aligned}\frac{\partial \tau_{cld}}{\partial sze} &= cng \times \left(\frac{\partial \tau_{ext}^{1g/m2}(\nu, sze)}{\partial dze}\right) - \\ &\quad \frac{cng}{2}\left(\frac{\partial \tau_{sca}^{1g/m2}(\nu, sze)}{\partial dze}(1 + g(\nu, sze))\right) - \\ &\quad \frac{cng}{2}\left(\tau_{sca}^{1g/m2}(\nu, sze)\frac{\partial g(\nu, sze)}{\partial sze})\right) \\ &= cng\frac{\partial f}{\partial sze}\end{aligned}$$

4. furthermore suppose the cloud is defined between $cprtop$ mb and $cprbot$ mb. Then it typically straddles more than one layer. The (cloud loading cng) fraction in each layer L that the cloud straddles is given by (exept at top most/bottom most cloud layer)

$$frc_{cng}(L) = \frac{plev(L+1) - plev(L)}{cprtop - cprbot}, LCLDTOP \le L \le LCLDBOT$$

so that each layer has fraction $frc_{cng}(L)\tau_{cld}(\nu, sze, \theta)$ of the total cloud optical depth. The derivatives of $frc_{cng}(L)$ with respect to $cprtop, cprbot$ are trivially (which of course means its not so trivial and the jacs are awful)

$$\frac{\partial frc_{cng}(L)}{\partial cprtop} = (-1)\frac{plev(L+1)-plev(L)}{(cprtop-cprbot)^2} \times (+1); \frac{\partial frc_{cng}(L)}{\partial cprbot} = (-1)\frac{plev(L+1)-plev(L)}{(cprtop-cprbot)^2} \times (-1);$$

(though compared to SARTA finite differences, the $cprtop, cprbot$ derivatives are sometimes off by -1 ... but this is probably due to finite difference perturbation in pressure ensuring cloud top/bottom layer is slightly different.

5. This partitioning of the total cloud OD into the individual AIRS pressure layers according to $frc_{cng}(L)$ has couple of implications for the jacobians : for any one of the four streams (clear, cloud1, cloud2, cloud1+2),

- to get the *cpsize,cngwat* column jacobian, we sum over each of the cloud layers (from LCTOP to LCBOT) with the layer contribution weighted by $frc_{cng}(L)$.
- to get the *cprtop,cprbot* parameters we need the the $frc_{cng}(L)$ changes at each layer, and so to get the column sum, *we weight each layer by 1*

7.1 Cloud Fraction jacs

From above, the jacobians wrt cloud fraction (ie a unity change!! pretty large!!!) are immediately

$$\begin{array}{rcr}
\frac{\partial r(v)}{\partial c1} & = & -rclr(v) + r1(v) \\
\frac{\partial r(v)}{\partial c2} & = & -rclr(v) + r2(v) \\
\frac{\partial r(v)}{\partial c12} & = & +rclr(v) - r1(v) - r2(v) + r12(v)
\end{array}$$

7.2 Cloud Amount and Cloud Particle Size jacs

As mentioned above our look-up tables are in terms of $\tau_{cld}(v) = cldOD(v) = cng \quad f(v, sze)$ from which (and use the formulation above)

$$\begin{array}{ccc}
\hline
cldOD(v) & = & cng \quad f(v,sze) \\
\hline
\frac{\partial cldOD}{\partial sze} & = & cng\frac{\partial f(v,sze)}{\partial sze} \\
\frac{\partial r(v)}{\partial sze1} & = & \frac{\partial r1(v)}{\partial sze1} + \frac{\partial r12(v)}{\partial sze1} \\
 & = & \frac{\partial r1(v,sze)}{\partial cldOD1}\frac{\partial cldOD1}{\partial sze1} + \frac{\partial r12(v)}{\partial cldOD1}\frac{\partial cldOD1}{\partial sze1} \\
 & = & cng1\left(\frac{\partial r1(v,sze)}{\partial cldOD1}\frac{\partial f1(v,sze1)}{\partial sze1} + \frac{\partial r12(v)}{\partial cldOD1}\frac{\partial f1(v,sze1)}{\partial sze1}\right) \\
\frac{\partial r(v)}{\partial sze2} & = & \frac{\partial r2(v)}{\partial sze2} + \frac{\partial r12(v)}{\partial sze2} \\
 & = & \frac{\partial r2(v,sze)}{\partial cldOD2}\frac{\partial cldOD2}{\partial sze2} + \frac{\partial r12(v)}{\partial cldOD2}\frac{\partial cldOD2}{\partial sze2} \\
 & = & cng2\left(\frac{\partial r2(v,sze)}{\partial cldOD2}\frac{\partial f2(v,sze2)}{\partial sze2} + \frac{\partial r12(v)}{\partial cldOD2}\frac{\partial f2(v,sze2)}{\partial sze2}\right) \\
\hline
\frac{\partial cldOD}{\partial cng} & = & f(v,sze) \\
\frac{\partial r(v)}{\partial cng1} & = & \frac{\partial r1(v)}{\partial cng1} + \frac{\partial r12(v)}{\partial cng1} \\
 & = & \frac{\partial r1(v,cng)}{\partial cldOD1}\frac{\partial cldOD1}{\partial cng1} + \frac{\partial r12(v)}{\partial cldOD1}\frac{\partial cldOD1}{\partial cng1} \\
 & = & \frac{\partial r1(v,cng)}{\partial cldOD1} f1(v,cng1) + \frac{\partial r12(v)}{\partial cldOD1} f1(v,sze1) \\
\frac{\partial r(v)}{\partial cng2} & = & \frac{\partial r2(v)}{\partial cng2} + \frac{\partial r12(v)}{\partial cng2} \\
 & = & \frac{\partial r2(v,cng)}{\partial cldOD2}\frac{\partial cldOD2}{\partial cng2} + \frac{\partial r12(v)}{\partial cldOD2}\frac{\partial cldOD2}{\partial cng2} \\
 & = & \frac{\partial r2(v,cng)}{\partial cldOD2} f2(v,cng2) + \frac{\partial r12(v)}{\partial cldOD2} f2(v,sze2)
\end{array}$$

7.3 Cloud Top and Cloud Bottom jacs

Scott has a cool way of implementing the cloud boundaries across pressure level boundaries and so can occupy a fraction of an AIRS layer; kCARTA "fills" in AIRS pressure levels so the cloud occupies one or more *full* pressure layers; all the cloud jacobian stuff I do for single footprint retrievals, ensure that (a) cld1 bottom and cld2 top do not merge/overlap (b) cld1 top or cld2 top does not become less than 0 mb (c) cld1 bottom or cld2 bottom does not become more than surf pres. But I did make a stab at it!!! Recall for any layer L the total optical depth is given by

$$\begin{aligned}
\tau_{total}(T,q_1,q_2,q_3,....,q_{fixed},cld_1,cld_2) &= \tau_1(T,q_1)+\tau_2(T,q_2)+....\tau_{12}(T,q_{12})+\\
&\quad \tau_{cld1}(cng1,sze1,cprtop1,cprbot1,species1)+\\
&\quad \tau_{cld2}(cng2,sze2,cprtop2,cprbot2,species2)
\end{aligned}$$

which means (assuming variable we are interested in X_i is not the temperature T)

$$\frac{\partial\tau_{total}}{\partial X_i}=\frac{\partial\tau_i}{\partial X_i}, i=q_1,q_2,....q_{12},cld1params,cld2params$$

From above, we have cloud loading fractions $frc_{cng}(L)$ and their derivatives, so in layer L

$$\begin{aligned}
\frac{\partial r(v,L)}{\partial cprtop/bot} &= \frac{\partial r(v,L)}{\partial\tau_{total}(v)}\frac{\partial\tau_{total}(v)}{\partial cprtop/bot}\\
\frac{\partial r(v,L)}{\partial cprtop/bot} &= \frac{\partial r(v,L)}{\partial\tau_{total}(v)}\frac{\partial\tau_{cld}(v)}{\partial cprtop/bot}\\
\frac{\partial r(v,L)}{\partial cprtop/bot} &= \left(\mu\frac{\partial r(v,L)}{\partial\tau_{total}(v)}\right)\left(\frac{1}{\mu}\frac{\partial\tau_{cld}(v)}{\partial cprtop/bot}\right)\\
\frac{\partial r(v,L)}{\partial cprtop/bot} &= (-r_L+B_L)\left(\frac{1}{\mu}\frac{\partial\tau_{cld}(v)}{\partial cprtop/bot}\right)
\end{aligned}$$

We know

$$\tau_{cld}(v,L)=frc_{cng}(L)\tau_{cld}^{total}(v)=frc_{cng}(L)cng\times\tau^{1g/m2}(v,sze,\theta=0)$$

which means

$$\begin{aligned}
\frac{\partial\tau_{cld}(v)}{\partial frc_{cng}(L)} &= cng\times\tau^{1g/m2}(v,sze,\theta=0)\\
\frac{\partial\tau_{cld}(v)}{\partial cprtop} &= \frac{\partial\tau_{cld}(v)}{\partial frc_{cng}(L)}\frac{\partial frc_{cng}(L)}{\partial cprtop}\\
\frac{\partial\tau_{cld}(v)}{\partial cprbot} &= \frac{\partial\tau_{cld}(v)}{\partial frc_{cng}(L)}\frac{\partial frc_{cng}(L)}{\partial cprbot}
\end{aligned}$$

and then we can use the same formulation as for $T(z), G1(z),...$ etc derivatives!!

8 Running SARTA Analytic Jacobian

This is very similar to running SARTA as usual, except there are couple additional command line arguments, and you read in the output jacobians using Matlab. A typical run would be

```
  [name of exec] fin=blah.op.rtp fout=blah.rad.rtp                          RADS ONLY
  [name of exec] fin=blah.op.rtp fout=blah.rad.rtp listj=-1 jacunit=0   RADS,JACS (dr/dq)
  [name of exec] fin=blah.op.rtp fout=blah.rad.rtp listj=-1 jacunit=1   RADS, JACS (dBT/dq)
  [name of exec] fin=blah.op.rtp fout=blah.rad.rtp listj=-1             RADS, JACS (q dBT/dq)
                                                                        <default>
```

8.1 Command Line arguments

Command line arguments include what Scott had *e.g.,*"rhot" with a couple new important ones; run times change appropriately as you change any or all of num of chans/jacs/profiles

- jacunit = 0,1,2
  - 0 : units are d(rad)/dX (ie d(rad)/dT or d(rad)/dq)
  - 1 : units are d(BT)/dX (ie d(BT)/dT or d(BT)/dq)
  - 2 : units are q(dBT)/dX (ie d(BT)/dT or d(BT)/dq) <DEFAULT>
- listp = list of profiles [Scott had this]
  - listp allows you to choose upto 20 profiles (default -1 = all, or just leave out this arg)
- listc = list of channels [new]
  - listc allows you to choose upto 20 channel IDs *e.g.,*1291 for the 1231 cm=1, or 449 for 791 cm-1. (default -1 = all, or just leave out this arg)
- listj = list of jacobians [new]
  - If you leave out listj, it does NO jacobians (ie radiance calcs) default
  - If you want jacs, then listj = -1 gives WGT, T, GID 1,2,3,4,5,6,9,12 or listj = your list *e.g.,*300,100,1,2,3 would give CLD, T and GID 1,2,3

9 Output readers

The main reader (in the MATLABCODE directory) is $readsarta\_jac.m$ and $readsarta\_jacV2.m$. The difference between the two is that the former outputs the jacobians as (numprof,numchan,X) while the latter outputs the jacobians as (X,numchan,numprof) where $X = 12,100,101$ for clouds, gases and temperature/surface temperature. In addition the readers ensure they output reals (4 bytes) and not doubles (8 bytes).

Right now the default is to change $\frac{\partial r(v,TOA)}{\partial T_J}$, $\frac{\partial r(v,TOA)}{\partial Q_J}$, $\frac{\partial r(v,TOA)}{\partial CLD_J}$ to $\frac{\partial BT(v,TOA)}{\partial T_J}$, $\frac{Q_J \partial BT(v,TOA)}{\partial Q_J}$, $\frac{CLD_J \partial BT(v,TOA)}{\partial CLD_J}$ but this can easily be coded out. The output jac files (f77 binary files) would have root name of fout, followed by _jacT or _jacGX or _jacCLD appended to them

```
Reader : /home/sergio/KCARTA/MATLAB/ or MATLABCODE
[w,d,iaProf,iaNumLay] = readsarta_jac(fname,type)
  eg.   'newdayx.rtp_WGTFCN',200  would read in weight functions
        'newdayx,rtp_jacTZ',100                 temp jacs
        'newdayx.rtp_jacG1,',1                  G1 jacs etc

For G1,G2, ... jacs (and weight functions) d would, for example, be
```

```
    24x2645x100    (24 profiles, 2645 channels, 100 layers)
For T          jacs, d would, for example, be
    24x2645x101    (24 profiles, 2645 channels, 100 layers+surface temp)
For Cloud      jacs, d would, for example, be
    24x2645x012    (24 profiles, 2645 channels, 12 jacs)
    01-05 : [cfrac1,cngwat1,cpsize1,cprtop1,cprbot1],
    06-10 : [cfrac2,cngwat2,cpsize2,cprtop2,cprbot2],
    11    : cfrac12
    12    : stemp

If you do
  [h,ha,p,pa] = rtpread('blah.rad.rtp');
then $w = h.vchan$ (ie the $h.vchan$ from the output radiance file)
```

iaProf lists the profiles you asked to calculate jacs for (from listp above), while iaNumLay is basically (for WGT FCN and gas jacs) p.nlevs-1 for those profiles, and an extra one for stemp for temp jacs (so has length p.nlevs)

10 The End (sort of)

Finite difference SARTA jacobians agree with kCARTA analytic jacobians except in the window region, since kCARTA needs the cloud top and bottoms to be at pressure level boundaries, full cloud layers while SARTA allows the top and bottom pressures to be "anywhere".

The SARTA analytic jacobians have been compared to SARTA finite difference jacobians, and kCARTA analytic jacobians. The agreement is very good, except sometimes the *cprtop or cprbot* derivatives can be flipped by a sign since the finite difference jacobian straddles (or loses) an additional cloud layer.

Figure 3 shows a typical comparison of SARTA analytic versus SARTA finite difference column jacobians.

Feel free to test and give feedback. Its a mix of old f77 SARTA code (the June 20, 2022 commit) first rewritten to loop over ichan) and then jacobians with f90 implicit loops. No modules (yet, dunno if ever). If the incFTC/fnmie links are changed to CRIS or IASI (ie at the command line, call the make -f CRIS or IASI) and the code re-compiled, it works for CRIS and IASI as well. And fingers crossed the breakouts/predictors are kept the same forever ie never ever change the bloody predictors. You can compare to finite diff jacs, done for only your chosen profile, using /home/sergio/MATLABCODE_Git/quicksartajac.m (make sure the finite diff jacobians use eg dQ=0.01 or dT = 0.1, else the numerical noise will creep in and ratios of jacs will look awful).

The testing code wrappers which call both the finite difference jacobian code, and the analytic jacobian code, are in the MATLABCODE subdirectory, and have the names

```
 test_jac_AIRS.m, test_jac_CRIS.m,test_jac_IASI.m
```

The results are of comparing eg

$$\begin{array}{ccc}
\sum_{i=1}^{N} TZ^{(i)}_{finite}(\nu) & to & \sum_{i=1}^{N} TZ^{(i)}_{analytic}(\nu) \\
\sum_{i=1}^{N} G1^{(i)}_{finite}(\nu) & to & \sum_{i=1}^{N} G1^{(i)}_{analytic}(\nu) \\
CldX_{finite}(\nu) & to & CldX_{analytic}(\nu)
\end{array}$$

etc should be great : the ratios should be very close to one and if plotted they should lie on top of each other.

Run times for 12150 profiles/2645 AIRS channels is 4 minutes, while adding in all jacobians and weighting functions (T jacs, G1,2,3,4,5,6,9,12 jacs) increases the run time by × 10. This is compared to running SARTA over and over for about 10 extra perturbations × 100 layers to get the corresponding finite difference jacobians, which is about 66 hours or about 2.75 days.

NOTE : For clear sky, the RTA integral is

$$r(\nu) = \epsilon(\nu) B(\nu, T_s) e^{-\tau_{total}(\nu)} + \int_{psurface}^{0} B(T(p)) \frac{\partial \tau}{\partial p} dp$$

which for numerical integration reduces to (see eg [De Souza-Machado et al., 2018])

$$r(\nu) = \epsilon(\nu) B(\nu, T_s) e^{-\tau_{total}(\nu)} + \sum B(T(i)) \frac{(1 - e^{-\tau_i}) e^{-\tau_{i+1 \to TOA}(\nu)}}{\delta p} \delta p$$

$$r(\nu) = \epsilon(\nu) B(\nu, T_s) e^{-\tau_{total}(\nu)} + \sum B(T(i)) (1 - e^{-\tau_i}) e^{-\tau_{i+1 \to TOA}(\nu)}$$

where B is the planck function. The weighting functions we save are simply the term in front of $B(T(i))$ ie $(1-e^{-\tau_i}) e^{-\tau_{i+1 \to TOA}(\nu)} = \delta \tau_{i \to TOA}(\nu)$. You still need to divide the output weighting function by $\delta(pressure)$ at each layer to get the true weighting function.

11 Acknowledgment

Scott Hannon wrote/developed most of the original SARTA code with Larrabee Strow and Howard Motteler. He worked closely with me on the interfacing with TwoSlab clouds. Tobias Wehr wrote parts of a HIRS Fast model analytic code and put that into a write up, which inspired me to do this! Seeing Xu Liu's expressions for PCRTM derivatives are a confirmation that I was on the right track in "propagating the derivatives".

12 Bibliography

Hartmut H. Aumann, S. DeSouza-Machado, S. Havemann, J. Vidot, M. Matricardi, X. Huang, C. Wilson, Manning. E., G. Liuzzi, Masiello. G., C. Serio, I. Moradi, X. Liu, V. Natraj, Y. Yung, X. Chen, A. Geer, L. Strow, and Fishbein. E. Evaluation of Radiative Transfer Models with Clouds. *J. Geophys. Res.*, 10: 10.1029/2017JD028063, 2018.

Hartmut H. Aumann, C. Wilson, A. Geer, X. Huang, X. Chen, S. DeSouza-Machado, and X. Liu. Global Evaluation of the Fidelity of Clouds in the ECMWF Integrated Forecast System. *Earth and Space Science*, 10:doi: 10.1029/2022EA002652, 2023.

M.-D. Chou, K.-T. Lee, S.-C. Tsay, and Q. Fu. Parameterization for Cloud Longwave Scattering for use in Atmospheric Models. *J. Climate*, 12:159–169, 1999.

S. De Souza-Machado, L. L. Strow, A. Tangborn, X. Huang, X. Chen, X. Liu, X. Wu, and Q. Yang. Single-footprint retrievals for AIRS using a fast TwoSlab cloud-representation model and the SARTA all-sky infrared radiative transfer algorithm. *Atmos. Meas. Tech.*, 11:529–550,https://doi.org/10.5194/amt–11–529–2018, 2018.

S. De Souza-Machado, L. L. Strow, H.E. Motteler, and S.E Hannon. kCARTA : A Fast Pseudo Line by Line Radiative Transfer Algorithm with Analytic Jacobians, Fluxes, Non-Local Thermodynamic Equilibrium and Scattering. *Atmos. Meas. Tech.*, 31:323–339, https://doi.org/10.5194/amt–13–323–2020, 2020.

R.M. Goody and Y.L. Yung. *Atmospheric Radiation: Theoretical Basis*, page 519 pages. Oxford University Press, 1989.

K.N. Liou. *An Introduction to Atmospheric Radiation*, page 583 pages. Academic Press, 1980.

X. Liu, W.L Smith, D.K. Zhou, and A.M. Larar. Principal component based radiative transfer model for hyperspectral sensors : theoretical concepts. *Appl. Opt*, 45:201–209, 2006.

L. M. McMillin, L. J. Crone, M. D. Goldberg, and T. J. Kleespies. Atmospheric transmittance of an absorbing gas 4:OPTRAN: A computationally fast and accurate transmittance model for absorbing gases with fixed and variable mixing ratios at variable viewing angles. *Appl. Opt.*, 34:6269–6274, 1995.

K. Stamnes, S.-C. Tsay, W. Wiscombe, and K. Jayaweera. Numerically stable algorithm for discrete ordinate method radiative transfer in multiple scattering and emitting layered media. *Appl. Opt*, 27: 2502–2509, 1988.

L. Strow, S. Hannon, S. DeSouza-Machado, D. Tobin, and H Motteler. An overview of the AIRS radiative transfer model. *IEEE Transactions on Geosciences and Remote Sensing*, 41:303–313, 2003.

G. Tang, P. Yang, G. W. Kattawar, X. Huang, E. J. Mlawer, B. A. Baum, and M. D. King. Improvement of the simulation of cloud longwave scattering in broadband radiative transfer models. *J. Atmos. Sc.*., 75: 2217–2233, 2018.

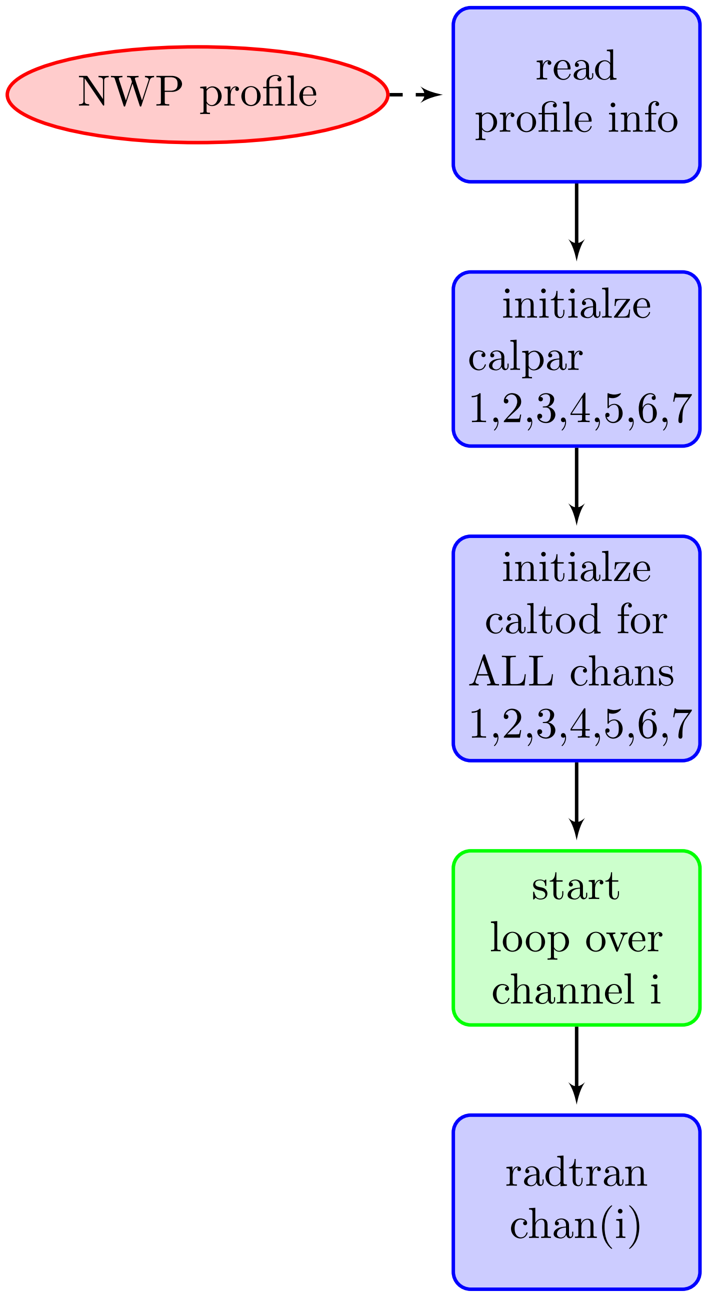
NWP profile
read profile info
initialze calpar 1,2,3,4,5,6,7
initialze caltod for ALL chans 1,2,3,4,5,6,7
start loop over channel i
radtran chan(i)

Figure 1: Flow diagram of SARTA PGE etc

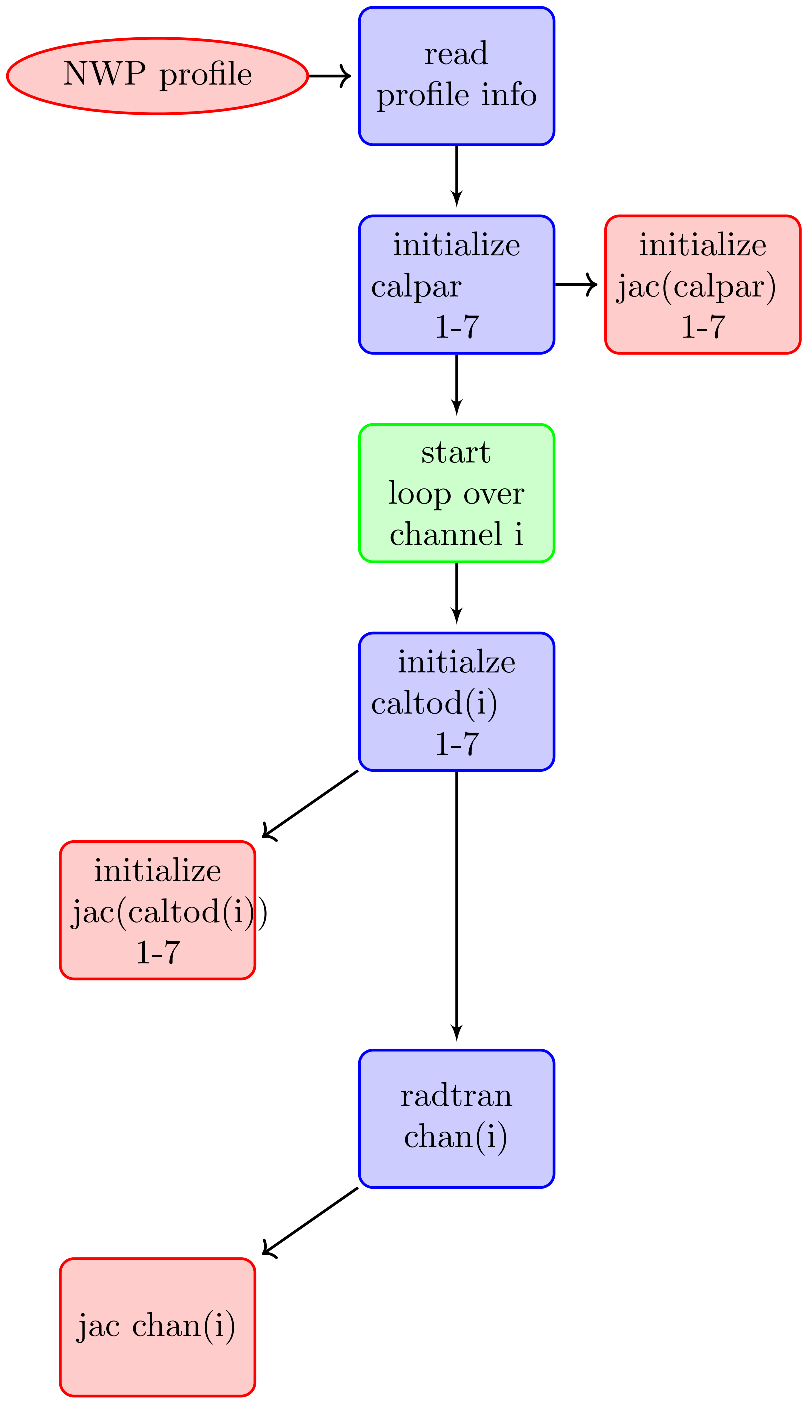

Figure 2: Flow diagram of SARTA Analytic Jacobian

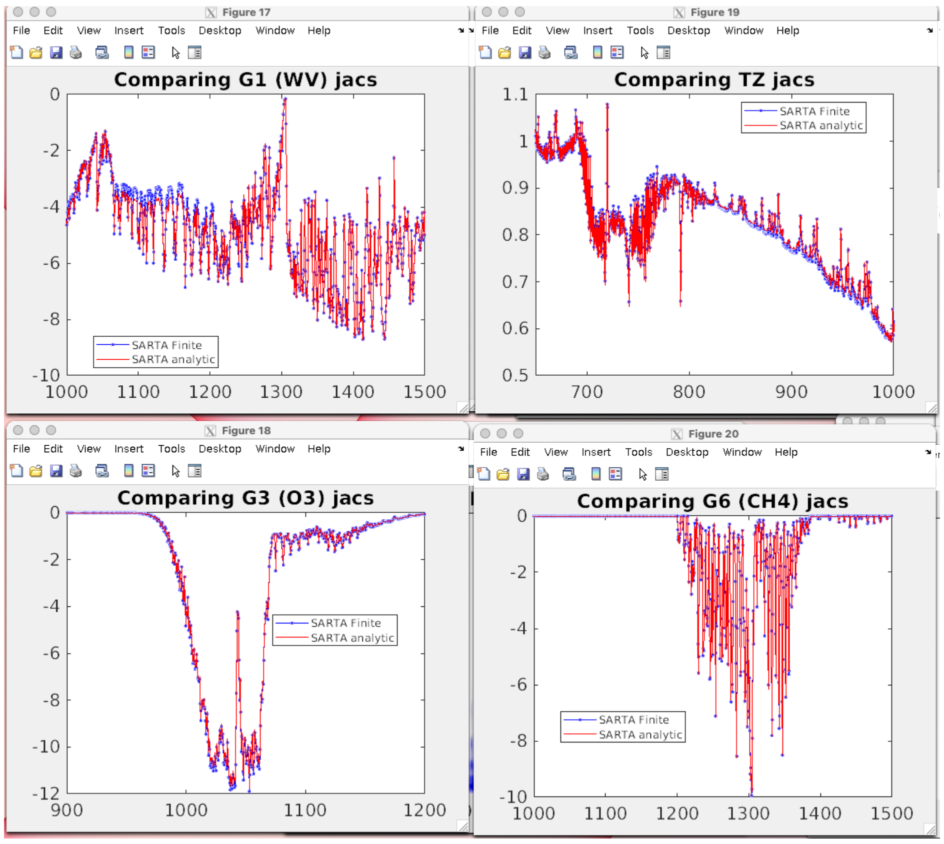

Figure 3: Comparison of SARTA finite diff VS analytic column jacobians