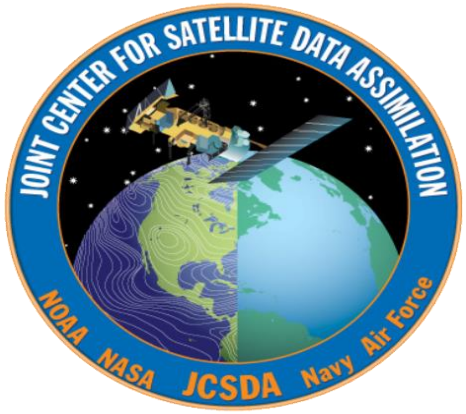
JCSDA
Community Radiative Transfer Model (CRTM) Framework



Yong Han, Quanhua Liu,

Paul van Delst

JCSDA 3rd Workshop on Satellite Data Assimilation,

20-21 April 2005

Overview

- User Interface
  - Forward, tangent-linear, adjoint, K-matrix (Jacobian)
  - *Public* data structures
- Developer Interface
  - The components
  - *Internal* data structures
  - Changes required for analytical Jacobians
- Shared Data
  - What are they? How to use them?
  - *Shared* data structures
- Testing
  - Code test software and datafiles.
  - Comparison with obs. Testing in GDAS.
- Utilities and Feedback

What is the CRTM Framework?

- At the simplest level, it's a collection of structure definitions, interface definitions, and stub routines.
- There are User and Developer interfaces, as well as Shared Data interfaces. (I/O functions for convenience).

Why do this?

- The radiative transfer problem is split into various components (e.g. gaseous absorption, scattering etc). Each component defines its own structure definition and application modules to facilitate independent development.
- Want to minimise or eliminate potential software conflicts and redundancies.
- Components developed by different groups can "simply" be dropped into the framework.
- Faster implementation of new science/algorithms.

User Interface

Current Forward CRTM Interface

```
Error_Status = CRTM_Forward( Atmosphere, &
                             Surface, &
                             GeometryInfo, &
                             ChannelInfo, &
                             RTSolution )
```

- All data contained in structures.
- Additional “arguments” can be added as required to the requisite structures.
- No impact on calling routine.

Allowable dimensionality

L = number of channels; M = number of profiles

INPUTS			OUTPUTS
Atmosphere	Surface	GeometryInfo	RTSolution
Scalar	Scalar	Scalar	L
M	M	M	L×M

Example: Definition of `Atmosphere` Structure

```
TYPE, PUBLIC :: CRTM_Atmosphere_type
  ! -- Dimension values
  INTEGER :: Max_Layers   = 0 ! K dimension
  INTEGER :: n_Layers     = 0 ! Kuse dimension
  INTEGER :: n_Absorbers  = 0 ! J dimension
  INTEGER :: Max_Clouds   = 0 ! Nc dimension
  INTEGER :: n_Clouds     = 0 ! NcUse dimension
  INTEGER :: Max_Aerosols = 0 ! Na dimension
  INTEGER :: n_Aerosols   = 0 ! NaUse dimension
  ! -- Climatology model associated with the profile
  INTEGER :: Climatology = INVALID_MODEL
  ! -- Absorber ID and units
  INTEGER, DIMENSION( : ), POINTER :: Absorber_ID    => NULL() ! J
  INTEGER, DIMENSION( : ), POINTER :: Absorber_Units => NULL() ! J
  ! -- Profile LEVEL pressure and LAYER quantities
  REAL( fp_kind ), DIMENSION( : ),    POINTER :: Level_Pressure => NULL() ! K
  REAL( fp_kind ), DIMENSION( : ),    POINTER :: Pressure       => NULL() ! K
  REAL( fp_kind ), DIMENSION( : ),    POINTER :: Temperature    => NULL() ! K
  REAL( fp_kind ), DIMENSION( :, : ), POINTER :: Absorber       => NULL() ! K x J
  ! -- Clouds associated with each profile
  TYPE( CRTM_Cloud_type ),   DIMENSION( : ), POINTER :: Cloud   => NULL() ! Nc
  ! -- Aerosols associated with each profile
  TYPE( CRTM_Aerosol_type ), DIMENSION( : ), POINTER :: Aerosol => NULL() ! Na
END TYPE CRTM_Atmosphere_type
```

PUBLIC CRTM Data Structures

Type Name	Description
`CRTM_ChannelInfo_type`	Sensor channel information filled during initialisation.
`CRTM_Atmosphere_type`	Atmospheric state profile data. Contains `Cloud` and `Aerosol` structures.
`CRTM_Surface_type`	Surface type and state information. Contains `SensorData` structure.
`CRTM_GeometryInfo_type`	Earth location, zenith and azimuth angles.
`CRTM_RTSolution_type`	Radiative transfer results.

Current K-Matrix CRTM Interface

```
Error_Status = CRTM_K_Matrix( Atmosphere, &
                              Surface, &
                              RTSolution_K, &
                              GeometryInfo, &
                              ChannelInfo, &
                              Atmosphere_K, &
                              Surface_K, &
                              RTSolution )
```

- Same structure definitions for both the forward and K-matrix structures.
- Channel dependencies are handled via the structure array dimensions.

Allowable dimensionality

L = number of channels; M = number of profiles

INPUTS			OUTPUTS	
Atmosphere Surface	RTSolution_K	GeometryInfo	Atmosphere_K Surface_K	RTSolution
Scalar	L	Scalar	L	L
M	L×M	M	L×M	L×M

Developer Interface

The CRTM Components

- Absorption by atmospheric gaseous constituents, e.g. water vapour, ozone, etc. **`AtmAbsorption`** functions.
  - Compact-OPTRAN is currently used.
  - OPTRAN-v7 has been implemented.
  - OSS has been implemented.
- Scattering and absorption. **`AtmScatter`** functions.
  - Aerosols
  - Clouds
- Surface Optics. **`SfcOptics`** functions.
  - Emissivity (land, ocean; μW, IR; ice, snow, water, etc)
  - Reflectivity (diffuse and direct)
- Radiative Transfer. **`RTSolution`** functions.
  - Fixed multi-stream models
  - SOI model

INTERNAL CRTM Data Structures

- Not visible via the User Interface
- Developers modify the structure contents as needed
- Some components are mandatory and must be supplied; others are algorithm specific.

Type Name	Description
`CRTM_AtmAbsorption_type`	Gaseous absorption optical depths and related parameters.
`CRTM_AtmScatter_type`	Scattering parameters such as single scatter albedo, asymmetry factor, optical depths, etc.
`CRTM_SfcOptics_type`	Surface optical properties such as emissivity and reflectivity.

Example: Definition of `AtmScatter` Structure

```
TYPE, PUBLIC :: CRTM_AtmScatter_type
  ! -- Dimension values
  INTEGER :: n_Layers           = 0 ! K dimension
  INTEGER :: Max_Legendre_Terms = 0 ! Ic dimension
  INTEGER :: n_Legendre_Terms   = 0 ! IcUse dimension
  INTEGER :: Max_Phase_Elements = 0 ! Ip dimension
  INTEGER :: n_Phase_Elements   = 0 ! IpUse dimension

  ! - Algorithm-specific members
  REAL( fp_kind ), DIMENSION( :, :, : ), POINTER :: Phase_Coefficient => NULL()

  ! - Mandatory members
  REAL( fp_kind ), DIMENSION( : ), POINTER :: Optical_Depth         => NULL() ! K
  REAL( fp_kind ), DIMENSION( : ), POINTER :: Single_Scatter_Albedo => NULL() ! K
  REAL( fp_kind ), DIMENSION( : ), POINTER :: Asymmetry_Factor      => NULL() ! K
  REAL( fp_kind ), DIMENSION( : ), POINTER :: Delta_Truncation      => NULL() ! K
END TYPE CRTM_AtmScatter_type
```

INTERNAL CRTM Functions

- “Stub” functions are provided for each component. Empty shell routines with the required structure arguments.
- Each component contained within its own module (or module hierarchy)
- Developers modify the contents of the application function. Can also include other dependent module subprograms, or other modules.
- Interfaces don’t change (eventually!), so impact of a particular component update (e.g. the gaseous absorption algorithm – replace OPTRAN with OSS) on the code is minimised.
- This is an ideal characterisation, as there may be dependencies between components (e.g. scaling of polychromatic gas absorption optical depths in the radiative transfer functions.)

Changes still to be made

- Modifications for analytic Jacobians
  - CRTM design was based on forward → tangent-linear → adjoint → K-matrix approach.
  - For analytic Jacobians, the K-matrix structures need to be passed in/out of the forward model call.
- Example: Computation of gaseous absorption.
  - Current forward model interface,

```
iErr = CRTM_Compute_AtmAbsorption( Channel_Index, &
                                   AtmAbsorption )
```

  - Interface update,

```
iErr = CRTM_Compute_AtmAbsorption( Channel_Index, &
                                   AtmAbsorption, &
                                   AtmAbsorption_K = AtmAbsorption_K )
```

- K-Matrix structure is declared as an optional output argument for the analytic Jacobians.

Shared Data

The CRTM Shared Data

- Shared Data is the precomputed data that is loaded during the model initialisation. The shared data is loaded into a public data structure that can then be used by application modules.
- Shared data is not visible via the User Interface.
- Needed for:
  - Gaseous absorption functions require regression coefficients (e.g. OPTRAN) or optical depth lookup tables (e.g. OSS)
  - Surface optics functions requiring coefficients (e.g. IRSSEM)
  - Scattering functions may require the same (e.g. current aerosol absorption/scattering uses channel based coefficients, but will transition to spectra)
- Accessed by **`USE`** of the required shared data module.
- Getting data into the system is one of the more difficult parts of CRTM development (IMO).

Current Shared Data CRTM Data Structures

Type Name	Description
`SpcCoeff_type`	Channel frequencies, polarisation, Planck function coefficients, etc.
`TauCoeff_type`	Coefficient data used in the `AtmAbsorption` functions.
`AerosolCoeff_type`	Coefficient data used in the `AerosolScatter` functions.
`ScatterCoeff_type`	Coefficient data used in the `CloudScatter` functions.

- Will need the same for surface optics functions to compute surface emissivities/reflectivities.

Testing Software

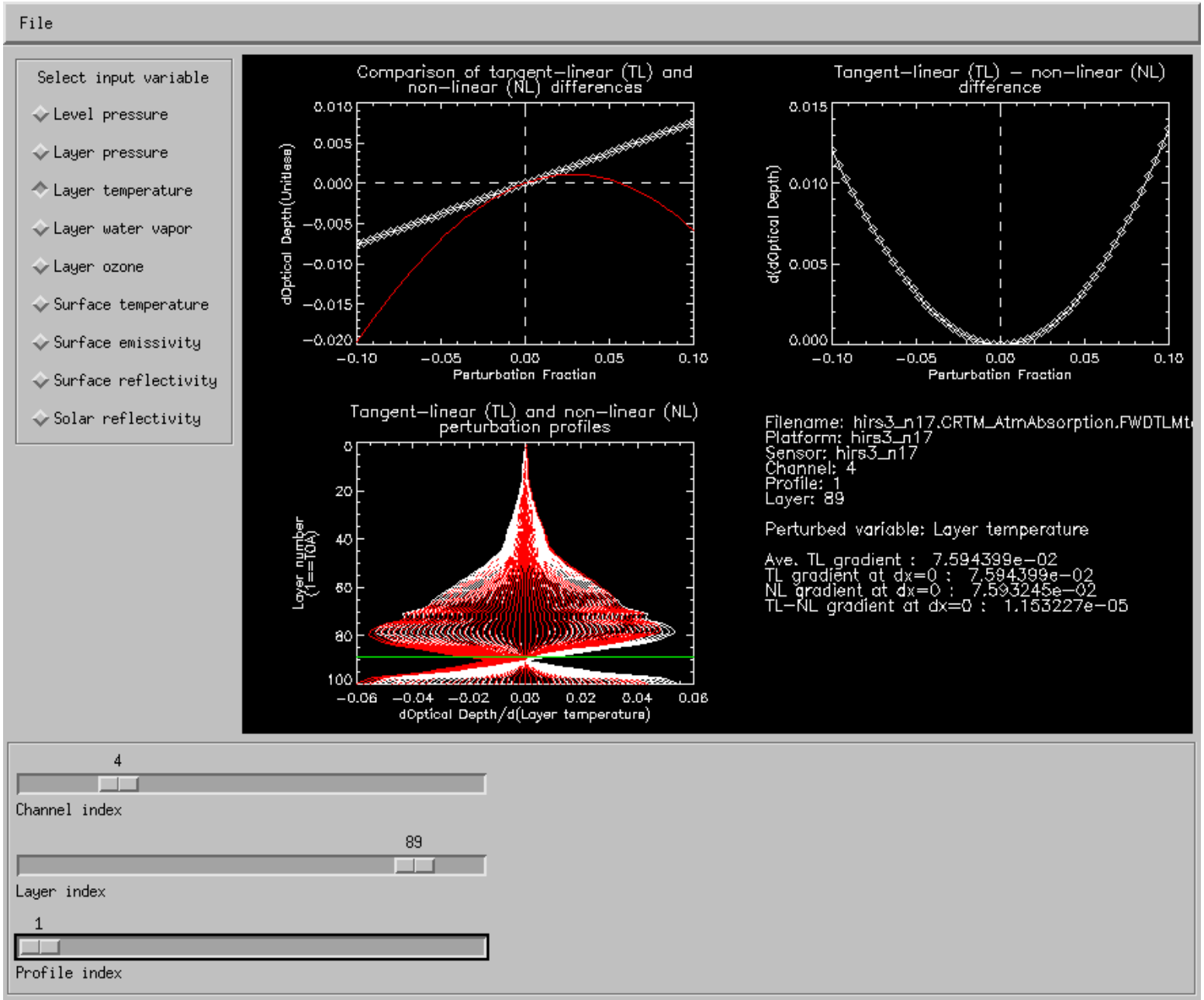
Testing Outline

- Data
  - Simple data initially.
    - Atmosphere data from ECMWF 52 profile set.
    - Cloud and Surface data from Quanhua Liu.
    - Aerosol data from Clark Weaver.
  - Collocated surface, atmosphere and satellite observations.
- Test instruments.
  - *Infrared*: AIRS (hyperspectral) and HIRS (broadband)
  - *Microwave*: AMSU (crosstrack scanner) and WindSat (polarimetric conical scanner)
- Anticipate three phases of testing:
  - Code testing using above simple datafiles.
  - Comparison with theoretical results and satellite observations.
  - Impact of CRTM on forecast skill.

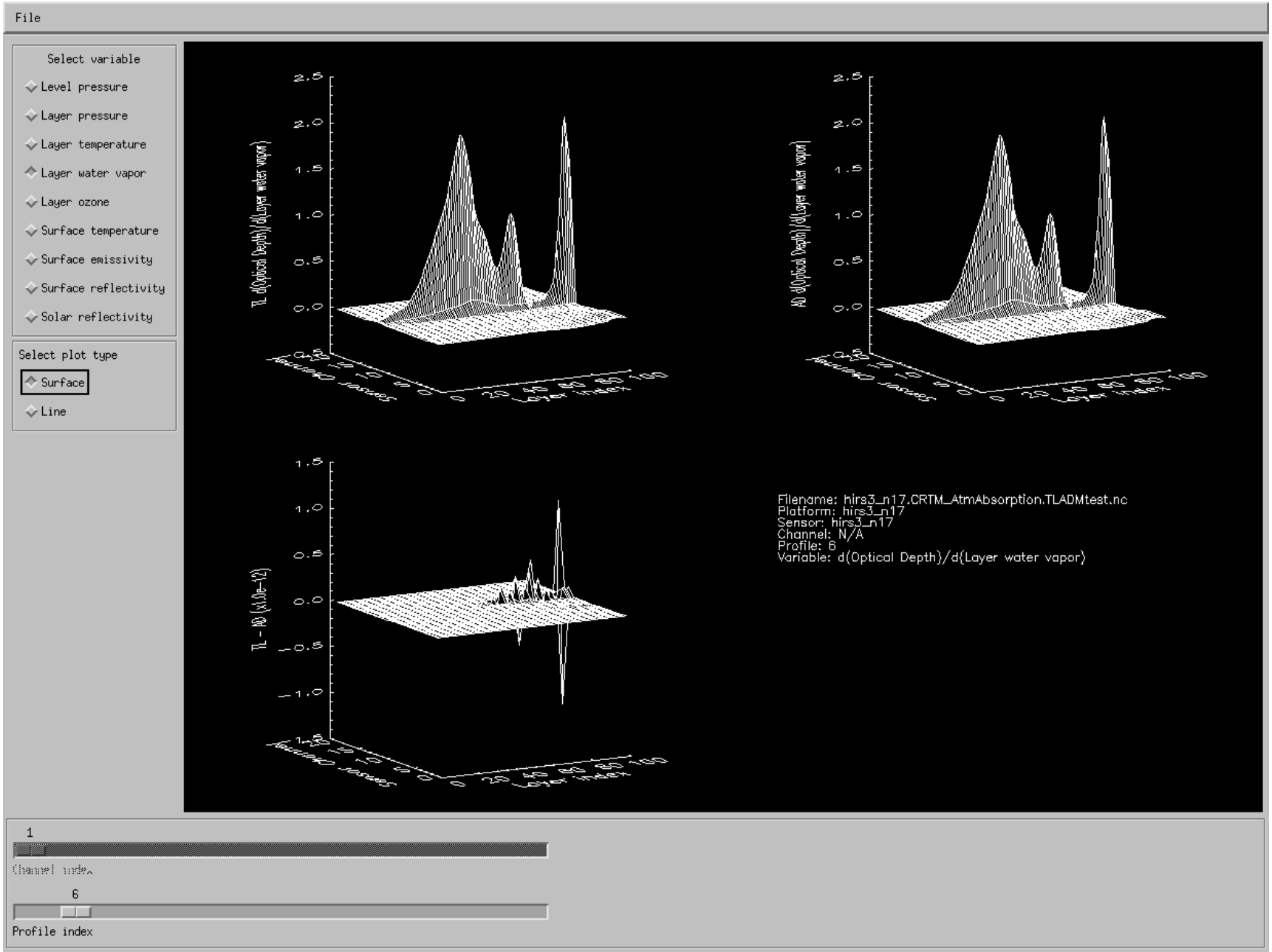
Testing Software

- Want to test each CRTM component (gaseous absorption, scattering, etc) in each model (Forward, K-matrix, etc) for consistency, as well as the end-to-end test.
- Have some initial attempts based on pCRTM and OPTRAN integration into CRTM.
  - Forward/Tangent-linear check.
  - Tangent-linear/Adjoint check
  - Adjoint/K-matrix check.
- Each test type has a defined structure that is filled and dumped to file.
- Some rudimentary visualisation tools using IDL.
- The same structure definitions and file I/O is also used in the pCRTM tests – the IDL code can view these also.

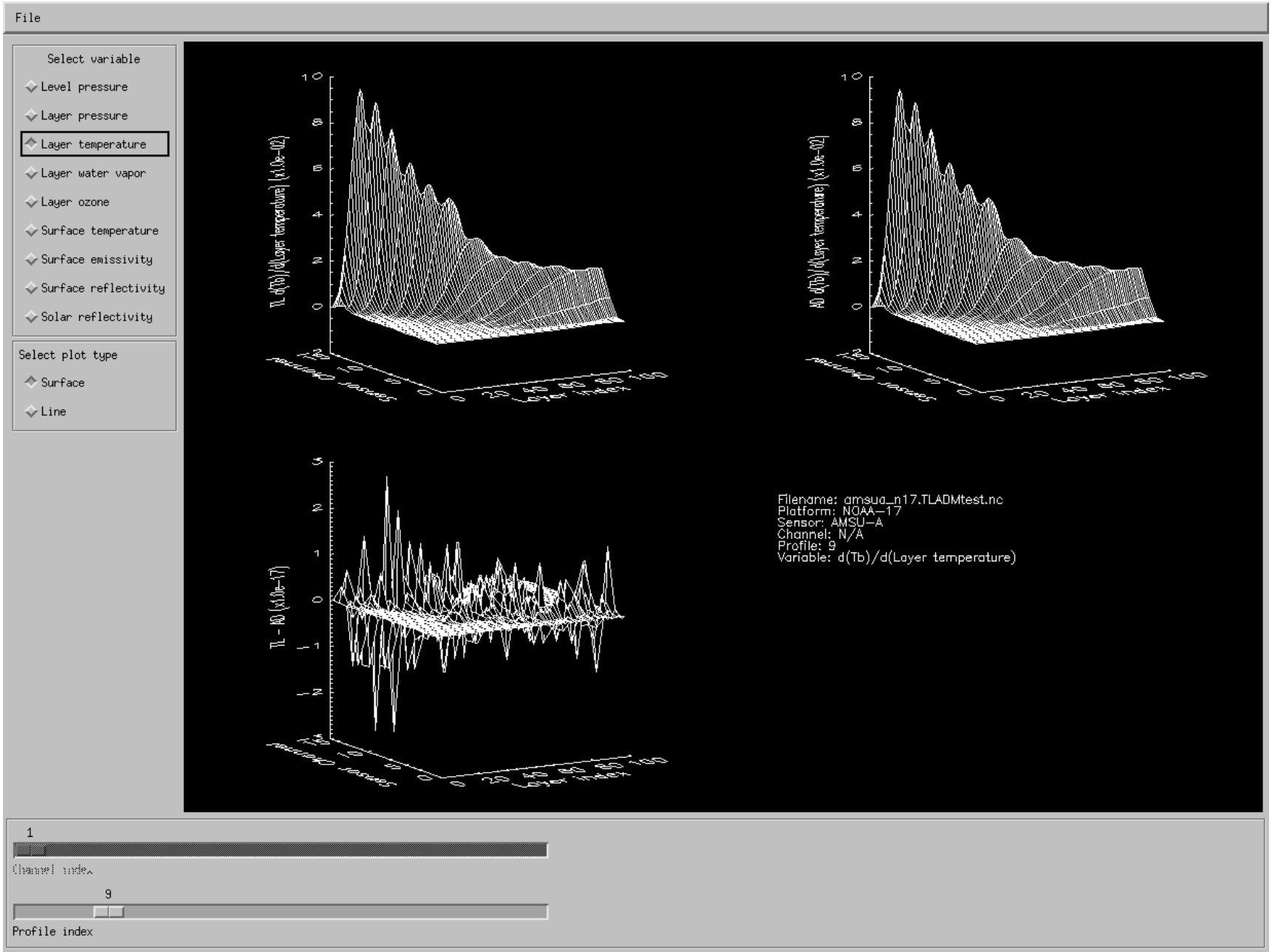
FWD/TL test results for `AtmAbsorption`



TL/AD test results for `AtmAbsorption`



TL/AD test results for the pCRTM



Utilities/Feedback

Utilities/Feedback

- What generic utilities are needed for the CRTM?
  - Example: Planck functions.
  - Others?
    - HG phase function computation routines?
    - Matrix operations/linear algebra libraries?
    - Other source functions (e.g. for NLTE)?
- Feedback
  - What do developers need?
  - How is the current setup working?
    - Easy to obtain and share software?
    - Easy to obtain and share data?
    - Sufficient documentation? Is the website sufficient? Deficient?
  - What do we need to do to make development, testing, and integration of CRTM components easier?
- Looking at SourceForge as a home for CRTM developers.