1 Directory structure

bin - executables and matlab driver scripts
fortran - Fortran raytracer source code
gcpm - GCPM (a plasmasphere model)
iri2007 - IRI (an ionosphere model, required by GCPM)
lapack-3.2.1 - Lapack and BLAS, used for solving Ax=b
manual - Documentation
matlab - Matlab raytracer
othersrc - Original distribution files for 3rd party sources
tricubic-for - Tricubic interpolator
tsyganenko - Tsyganenko magnetosphere model
xform - coordinate transformation tools (single precision)
xform_double - coordinate transformation tools (double precision)

2 Sample driver script

The bin/ directory contains a Matlab script "test_vlf_run.m". This sets up a few sample raytracer runs. It also illustrates the use and parsing of the basic tools to extract model parameters and plot them. Basic operation:

Set up global parameters

- **dt0** - Initial timestep in seconds
- **dtmax** - Maximum allowable timestep in seconds
- **root** - Which root of the appleton-hartree equation (2=whistler in the magnetosphere)
- **fixedstep** - Whether to use fixed steps (1) or adaptive timesteps (0). Fixed steps have NO way to recover if you pop outside the resonance cone, so adaptive is always recommended
- **maxerr** - Error bound for adaptive timestepping
- **maxsteps** - Maximum number of steps
- **outputper** - Output only every "outputper" timesteps (reduces output file size). Defaults to 1 (output every timestep).
- **minalt** - Minimum altitude (referenced from the center of the earth)
- **modelnum** - Model (1=ngo, 2=GCPM (SLOW!!), 3=interpolated (gridded data), 4=interpolated (scattered data))

Choose a plasmasphere model (modelnum)

- **1**: Ngo model, the classic radial diffusion model used by the original raytracer. This requires the same input file (e.g. "newray.in") as used by the original raytracer to set up the plasmasphere parameters (ducts, knee distance, etc.). newray.in here is ONLY used to set up the plasmasphere parameters for this model. The new raytracer uses a different format for inputting rays.

- **2**: GCPM model. This is a complete plasmasphere model but is slow in practice. It takes as input a Kp index (see below).
- **3**: Interpolated model. This takes as input a gridded set of number densities (format: log base e of the number density in m^{-3}. E.g., use "gcpm_dens_model_buildgrid" to generate a grid for use with this model.
- **4**: Scattered interpolated model. This takes as input a scattered set of number densities (format: log base e of the number density in m^{-3}. E.g., use "gcpm_dens_model_buildgrid_random" to generate a data set for use with this model.

Set up plasmasphere model parameters

- **Model 1: Ngo Parameters**
  - **ngo_configfile**: newray input filename
  - **yearday**: year and day, e.g., 1999098
  - **milliseconds_day**: milliseconds of day
  - **use_tsyganenko**: use Tsyganenko magnetic field model? (1=use, 0=do not use)
  - **use_igrf**: use IGRF magnetic field model or dipole model? (1=IGRF, 0=dipole)
  - **tsyganenko_Pdyn**: Dynamic solar wind pressure in nP, used by the Tsyganenko model - 0.5 and 10 nPa
  - **tsyganenko_Dst**: Dst in nT, used by the Tsyganenko model - between -100 and +20 in nT
  - **tsyganenko_ByIMF**: IMF y component in nT, used by the Tsyganenko model - between -10 and +10 nT -
  - **tsyganenko_BzIMF**: IMF z component in nT, used by the Tsyganenko model - between -10 and +10 nT
  - **tsyganenko_W1**: W1, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W2**: W2, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W3**: W3, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W4**: W4, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W5**: W5, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W6**: W6, a model coefficient used by the Tsyganenko model
- **Model 2: GCPM Parameters**
  - **gcpm_kp**: kp index
  - **yearday**: year and day, e.g., 1999098
  - **milliseconds_day**: milliseconds of day
  - **use_tsyganenko**: use Tsyganenko magnetic field model? (1=use, 0=do not use)
  - **use_igrf**: use IGRF magnetic field model or dipole model? (1=IGRF, 0=dipole)
  - **tsyganenko_Pdyn**: Dynamic solar wind pressure in nP, used by the Tsyganenko model - 0.5 and 10 nPa
  - **tsyganenko_Dst**: Dst in nT, used by the Tsyganenko model - between -100 and +20 in nT
  - **tsyganenko_ByIMF**: IMF y component in nT, used by the Tsyganenko model - between -10 and +10 nT -

  - **tsyganenko_BzIMF**: IMF z component in nT, used by the Tsyganenko model - between -10 and +10 nT
  - **tsyganenko_W1**: W1, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W2**: W2, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W3**: W3, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W4**: W4, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W5**: W5, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W6**: W6, a model coefficient used by the Tsyganenko model

- **Model 3: Interpolated parameters**
  - **interp_interpfile**: grid filename
  - **yearday**: year and day, e.g., 1999098
  - **milliseconds_day**: milliseconds of day
  - **use_tsyganenko**: use Tsyganenko magnetic field model? (1=use, 0=do not use)
  - **use_igrf**: use IGRF magnetic field model or dipole model? (1=IGRF, 0=dipole)
  - **tsyganenko_Pdyn**: Dynamic solar wind pressure in nP, used by the Tsyganenko model - 0.5 and 10 nPa
  - **tsyganenko_Dst**: Dst in nT, used by the Tsyganenko model - between -100 and +20 in nT
  - **tsyganenko_ByIMF**: IMF y component in nT, used by the Tsyganenko model - between -10 and +10 nT -
  - **tsyganenko_BzIMF**: IMF z component in nT, used by the Tsyganenko model - between -10 and +10 nT
  - **tsyganenko_W1**: W1, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W2**: W2, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W3**: W3, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W4**: W4, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W5**: W5, a model coefficient used by the Tsyganenko model
  - **tsyganenko_W6**: W6, a model coefficient used by the Tsyganenko model

- **Model 4: Scattered interpolator parameters**
  - **interp_interpfile**: grid filename
  - **yearday**: year and day, e.g., 1999098
  - **milliseconds_day**: milliseconds of day
  - **use_tsyganenko**: use Tsyganenko magnetic field model? (1=use, 0=do not use)
  - **use_igrf**: use IGRF magnetic field model or dipole model? (1=IGRF, 0=dipole)
  - **tsyganenko_Pdyn**: Dynamic solar wind pressure in nP, used by the Tsyganenko model - 0.5 and 10 nPa
  - **tsyganenko_Dst**: Dst in nT, used by the Tsyganenko model - between -100 and +20 in nT
  - **tsyganenko_ByIMF**: IMF y component in nT, used by the Tsyganenko model - between -10 and +10 nT -
  - **tsyganenko_BzIMF**: IMF z component in nT, used by the Tsyganenko model - between -10 and +10 nT
  - **tsyganenko_W1**: W1, a model coefficient used by the Tsyganenko model

- **tsyganenko_W2**: W2, a model coefficient used by the Tsyganenko model
- **tsyganenko_W3**: W3, a model coefficient used by the Tsyganenko model
- **tsyganenko_W4**: W4, a model coefficient used by the Tsyganenko model
- **tsyganenko_W5**: W5, a model coefficient used by the Tsyganenko model
- **tsyganenko_W6**: W6, a model coefficient used by the Tsyganenko model
- **scattered_interp_window_scale**: This is the factor above the maximum sample spacing of the moving least squares sphere (the points that are actually considered for interpolation). Above 1.1 or so is usually safe, but set to 1.5 or even larger for robustness.
- **scattered_interp_order**: The order of the monomials used for interpolation (2 is usually fine).
- **scattered_interp_exact**: Whether to use (1) or not (0) exact interpolation. Using inexact (0) often generates smoother and more accurate results around discontinuities or high gradients, at the expense of not being an exact interpolant.
- **scattered_interp_local_window_scale**: During evaluation, the minimum average spacing in a neighborhood around the interpolation point is used to shrink or grow the weighting window. This gives better results on strongly inhomogeneous grids. Setting to some small value like 2.0 will resolve sharp features better. Setting very large, like 10, will smooth out sharp features but will generally be smoother and suffer from fewer artifacts.

3 Computing damping rate

The hot damping rate is computed entirely as a postprocessing step using matlab scripts. A sample damping calculation and some plots are available in the script "test_vlf_damping.m".

4 Building

In the top-level directory, there is a makefile. Run "make clean" then "make" in that directory to rebuild everything. Compilation requires a Fortran 90 compiler (possibly with some fortran 95 features). It is tested with g95 and gfortran. Some of the third-party packages require static variables; in g95 this is "-fstatic"; in gfortran, "-fno-automatic". Check and modify the Makefiles accordingly.

Under Windows, the easiest way to compile this package is to download MinGW and the MinGW g95 package. After installing, modify your PATH and then run "make clean" then "make".

5 Tools

dumpmodel

This tool dumps a plasmasphere/magnetic field model to an ASCII text file. Run the tool without arguments to see usage. A typical run would be:

```
dumpmodel --minx=0 --maxx=3.1855e+007 --miny=0 --maxy=0 \
  --minz=-1.59275e+007 --maxz=1.59275e+007 --nx=200 --ny=1 \
```

```
--nz=200 --filename=dumpout.txt --modelnum=1 --yearday=2001001 \
--milliseconds_day=0 --use_tsyganenko=0 --use_igrf=0 \
--tsyganenko_Pdyn=4 --tsyganenko_Dst=0 --tsyganenko_ByIMF=0 \
--tsyganenko_BzIMF=-5 --ngo_configfile=newray.in
```

This will output a textfile named "dumpout.txt". The first line of the file is a header:

```
nspec nx ny nz
```

In order: the number of species, number of samples in the x direction, number of samples in the y direction, and number of samples in the z direction, respectively. The second line is:

```
minx maxx miny maxy minz maxz
```

In order: minimum extent in the x direction, maximum extent in the x direction, minimum extent in the y direction, maximum extent in the y direction, minimum extent in the z direction, and maximum extent in the z direction.

The remainder of the file is the data.

The Matlab function **readdump.m** reads the output from this tool.

gcpm_dens_model_buildgrid

This tool is similar to the model dump tool, but only outputs the plasmasphere parameters and optionally their derivatives, for use by Model 3, the interpolated model. GCPM is prohibitively slow so for a large number of runs, it is more efficient to generate a large grid once over the region of interest, and then on later runs, interpolate over this grid. Note that the format of this file is in log base e of the electron number density in m^{-3}!

Run the program without arguments to get the usage. A typical run would be:

```
gcpm_dens_model_buildgrid --minx-1e7 --maxx=1e7 --miny=0 --maxy=0 \
  --minz=-1e7 --maxz=1e7 --nx=40 --ny=1 --nz=40 --compder=0 \
  --filename=out.txt --gcpm_kp=4.0 --yearday=2001001 --milliseconds_day=0
```

One file has already been generated for testing:

gcpm_kp4_2001001_L10_80x80x80_noderiv.txt - kp=4, yearday=2001001, range=+-10 earth radii in every direction, 80x80x80 cells, no derivatives calculated at the faces.

The Matlab function **readinterpolationgrid.m** reads the output from this tool.

gcpm_dens_model_buildgrid_random

This tool uses stratified random sampling to sample GCPM on a random grid for use with Model 4, the scattered interpolated model. GCPM is prohibitively slow so for a large number of runs, it is more efficient to generate a large grid once over the region of interest, and then on later runs, interpolate over this grid. Note that the format of this file is in log base e of the electron number density in m^{-3}!

Run the program without arguments to get the usage. A typical run would be:

```
gcpm_dens_model_buildgrid_random --minx=-6.371e7 --maxx=6.371e7 \
  --miny=-6.371e7 --maxy=6.371e7 --minz=-6.371e7 --maxz=6.371e7 \
  --n_zero_altitude=5000 --n_iri_pad=20000 --n_initial_radial=0 \
  --n_initial_uniform=200000 --initial_tol=1.0 --max_recursion=80 \
  --adaptive_nmax=600000 --filename=out.txt --gcpm_kp=4.0 \
  --yearday=2001001 --milliseconds_day=0
```

One file has already been generated for testing:

gcpm_kp4_2001001_L10_random_5000_20000_0_200000_600000.txt - kp=4, yearday=2001001, range=+-10 earth radii in every direction, about 825000 total samples, 20000 extra samples in the ionopshere, 5000 extra samples at the earth's surface, 200000 uniform random samples, and 600000 samples used for refinement.

Explanation of parameters:

- **n_zero_altitude**
  Number of initial sample points at zero altitude. This number is for the WHOLE earth.
- **n_iri_pad**
  Number of initial points to sample at IRI altitudes (up to 2000 km). The gradients here are strong so lots of sampling is required to keep the errors low. This number is for the WHOLE earth.
- **n_initial_radial**
  Number of points to sample everywhere else on radially-weighted random points
- **n_initial_uniform**
  Number of points to sample everywhere else, uniform distribution over the bounds given.
- **initial_tol**
  Adaptive oversampling starting tolerance (1.0 is suitable for GCPM).
- **max_recursion**
  Adaptive oversampling recursion depth (set to around 80 just to be safe).
- **adaptive_nmax**
  Maximum number of adaptive samples that are allowed. The tolerance will be successively halved until this number of samples is exceeded.
- **filename**
  output filename
- **inputfile**
  (optional) existing output file to read in any points in this file will be made part of the initial grid and will also get pushed to the output.
- **gcpm_kp**
  kp index (GCPM parameter)
- **yearday**
  year and day, e.g., 1999098 (GCPM parameter)
- **milliseconds_day**
  milliseconds of day (GCPM parameter)

6 Raytracer

The core raytracer tool is called “raytracer”. The input arguments are identical to those described in testrun.m. The command-line arguments are as follows:

```
Raytracer version 1.09
 Usage:
   program --param1=value1 --param2=value2 ...

 --dt0           initial timestep in seconds
 --dtmax         maximum timestep in seconds
 --tmax          maximum time for simulation
 --root          root number (2=whistler at VLF frequencies)
 --fixedstep     fixed timesteps (1) or adaptive (0)
 --maxerr        maximum error for adaptive timestepping
 --maxsteps      maximum number of timesteps
 --minalt        minimum altitude
 --inputraysfile ray input filename
 --outputfile    output filename
 --modelnum      (1) Ngo model
                 (2) GCPM ionosphere model
                 (3) Interpolated model (gridded data)
                 (4) Interpolated model (scattered data)
  Ngo Parameters (required if model 1 is chosen):
    --ngo_configfile     newray input filename
    --yearday            year and day, e.g., 1999098
    --milliseconds_day   milliseconds of day
    --use_tsyganenko     (1=use, 0=do not use)
    --use_igrf           (1=use, 0=do not use)
    --tsyganenko_Pdyn    between 0.5 and 10 nPa
    --tsyganenko_Dst     between -100 and +20 in nT
    --tsyganenko_ByIMF   between -10 and +10 nT
    --tsyganenko_BzIMF   between -10 and +10 nT
    --tsyganenko_W1      TS04_s W1
    --tsyganenko_W2      TS04_s W2
    --tsyganenko_W3      TS04_s W3
    --tsyganenko_W4      TS04_s W4
    --tsyganenko_W5      TS04_s W5
    --tsyganenko_W6      TS04_s W6
  GCPM Parameters (required if model 2 is chosen):
    --gcpm_kp            kp index
    --yearday            year and day, e.g., 1999098
    --milliseconds_day   milliseconds of day
    --use_tsyganenko     (1=use, 0=do not use)
    --use_igrf           (1=use, 0=do not use)
    --tsyganenko_Pdyn    between 0.5 and 10 nPa
    --tsyganenko_Dst     between -100 and +20 in nT
    --tsyganenko_ByIMF   between -10 and +10 nT
    --tsyganenko_BzIMF   between -10 and +10 nT
    --tsyganenko_W1      TS04_s W1
    --tsyganenko_W2      TS04_s W2
```

```
  --tsyganenko_W3      TS04_s W3
  --tsyganenko_W4      TS04_s W4
  --tsyganenko_W5      TS04_s W5
  --tsyganenko_W6      TS04_s W6
Interp parameters (required if model 3 is chosen):
  --interp_interpfile  grid filename
  --yearday            year and day, e.g., 1999098
  --milliseconds_day   milliseconds of day
  --use_tsyganenko     (1=use, 0=do not use)
  --use_igrf           (1=use, 0=do not use)
  --tsyganenko_Pdyn    between 0.5 and 10 nPa
  --tsyganenko_Dst     between -100 and +20 in nT
  --tsyganenko_ByIMF   between -10 and +10 nT
  --tsyganenko_BzIMF   between -10 and +10 nT
  --tsyganenko_W1      TS04_s W1
  --tsyganenko_W2      TS04_s W2
  --tsyganenko_W3      TS04_s W3
  --tsyganenko_W4      TS04_s W4
  --tsyganenko_W5      TS04_s W5
  --tsyganenko_W6      TS04_s W6
Scattered interp parameters (required if model 4 is chosen):
  --interp_interpfile  data filename
  --yearday            year and day, e.g., 1999098
  --milliseconds_day   milliseconds of day
  --use_tsyganenko     (1=use, 0=do not use)
  --use_igrf           (1=use, 0=do not use)
  --tsyganenko_Pdyn    between 0.5 and 10 nPa
  --tsyganenko_Dst     between -100 and +20 in nT
  --tsyganenko_ByIMF   between -10 and +10 nT
  --tsyganenko_BzIMF   between -10 and +10 nT
  --tsyganenko_W1      TS04_s W1
  --tsyganenko_W2      TS04_s W2
  --tsyganenko_W3      TS04_s W3
  --tsyganenko_W4      TS04_s W4
  --tsyganenko_W5      TS04_s W5
  --tsyganenko_W6      TS04_s W6
  --scattered_interp_window_scale
                       window radius scale factor above
                       maximum sample spacing
  --scattered_interp_order
                       monomial order
  --scattered_interp_exact
                       exact(1) or inexact(0)
  --scattered_interp_local_window_scale
                       scale factor above minimum average
                       sample spacing
```

The input file “inputraysfile” gives the ray starting positions, initial wavenormal directions, and frequency (in rad/s), one ray per line, e.g.:

```
1.0E007 0.0 0.0 0.0 0.0 1.0 6.28e4
```

This input file launches one ray at position (1e7,0,0) with an initial wavenormal direction of (0,0,1) and a frequency of 6.28e4 rad/s (approximately 10 kHz). By convention, every model uses SM (solar magnetic) coordinates.

The output file "outputfile" records the ray positions, the time since start, positions, wavenormals, relative group velocities, ray frequencies, and the magnetic field and plasma parameters along the ray path. The Matlab script "readrayoutput.m" parses this file.

7 Adapting a new model

The raytracer includes 4 default plasmasphere models (Ngo, GCPM, interpolated, scattered interpolated), two magnetic field models (dipole and IGRF), and one magnetic field correction model (Tsyganenko). The raytracer uses these models to compute the dispersion relation at arbitrary points in space.

Adding a new model requires adding a new *adapter* to the raytracer and recompiling. A sample adapter is included in "skeleton_dens_model_adapter.f95". The adapter, reduced to its bare minimum, is simply a Fortran 90 module that contains a single subroutine that returns the plasma parameters for a spatial position, and conforms to a specific interface:

```
! Implementation of the plasma parameters function.
! Inputs:
!   x - position vector in cartesian (SM) coordinates
! Outputs:
!  qs - vector of species charges
!  Ns - vector of species densities in m^-3
!  ms - vector of species masses in kg
! nus - vector of species collisions in s^-1
!  B0 - cartesian (SM) background magnetic field in Tesla
! In/out:
! funcPlasmaParamsData - arbitrary callback data
subroutine funcPlasmaParams(x, qs, Ns, ms, nus, B0, funcPlasmaParamsData)
```

This function, which you should implement, takes as input a 3-vector x (the position in SM coordinates in meters), optionally some arbitrary state data in the character array "funcPlasmaParamsData", and outputs the quantities qs, Ns, ms, nus, and B0 for that position (this function should also allocate space for qs, Ns, ms, and nus if they are not already allocated). qs, Ns, ms, and nus should all be identically-sized vectors, one for each plasma species, and return the charges "qs" in C, number densities "Ns" in m^{-3}, masses "ms" in kg, and collision frequency (currently unused!) in s^{-1}. B0 is a simple cartesian 3-vector giving the background magnetic field in Teslas.

Most 3rd-party models require some additional state variables besides position. These can be set in module-level variables (essentially scoped global variables), or, better, in a callback state variable "funcPlasmaParamsData". "funcPlasmaParamsdata" is a simple allocatable character array. The Fortran intrinsic function TRANSFER can be used to transfer any derived type into and out of this state variable, fulfilling the same role held by a "void *" argument in C.

For example, suppose we have one state variable "use_igrf" that we wish to set before starting the raytracer. "skeleton_dens_model_adapter" illustrates. First we define a derived type as a container:

```
type :: StateData
   ! Whether to use (1) IGRF or not use (0) and use dipole instead
   integer*4 :: use_igrf
end type StateData
```

We then define another derived type containing a pointer to this derived type. This is the data we will actually pass through the "funcPlasmaParamsData" parameter. Note that the pointer must be contained within a derived type so TRANSFER knows how to handle it properly:

```
! Pointer container type.  This is the data that is actually marshalled.
type :: StateDataP
   type(StateData), pointer :: p
end type StateDataP
```

Then, in the main fortran file "raytracer_driver.f95", we create an empty character array, an instance of the type and its pointer container and associate the instance to its pointer container:

```
! empty, allocatable character array
character, allocatable :: data(:)

! declare the instances
type(StateData),target :: my_state_data
type(StateDataP) :: my_state_datap

! associate the pointer
my_state_dataP%p => my_state_data
```

Set our state variable:

```
my_state_data%use_igrf = 1
```

Then marshall the pointer container into the data array:

```
! marshall the data pointer to our function
sz = size(transfer(my_state_datap, data))
allocate(data(sz))
data = transfer(my_state_dataP, data)
```

Then, in the function that we've implemented, "funcPlasmaParams", we need to unmarshall this data in order to use it:

```
! Create the derived type pointer container
type(StateDataP) :: datap

! Unmarshall the callback data
datap = transfer(funcPlasmaParamsData, datap)

! Now you can access the data as datap%p%use_igrf
```

Finally, run the raytracer, passing our function and state variables as an argument:

```
call raytracer_run( &
     pos, time, vprel, vgrel, n, &
     B0, qs, ms, Ns, nus, stopcond, &
     pos0, dir0, w, dt0, dtmax, maxerr, maxsteps, minalt, root, tmax, &
     fixedstep, del, funcPlasmaParams, data, raytracer_stopconditions)
```