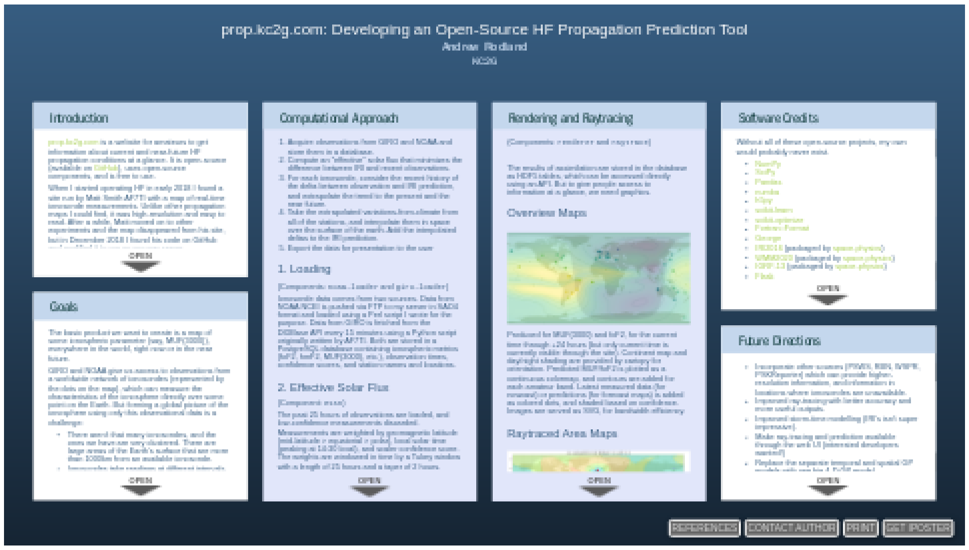
prop.kc2g.com: Developing an Open-Source HF Propagation Prediction Tool



Andrew Rodland

KC2G

PRESENTED AT:

INTRODUCTION

prop.kc2g.com (https://prop.kc2g.com) is a website for amateurs to get information about current and near-future HF propagation conditions at a glance. It is open-source (available on GitHub (https://github.com/arodland/prop/)), uses open-source components, and is free to use.

When I started operating HF in early 2018 I found a site run by Matt Smith AF7TI with a map of real-time ionosonde measurements. Unlike other propagation maps I could find, it was high-resolution and easy to read. After a while, Matt moved on to other experiments and the map disappeared from his site, but in December 2018 I found his code on GitHub and modified it to run on my own server.

Once I had the code running I started to find opportunities for improvement. The site as it stands today is the result of more than two years of research and development in my spare time, and has a small but growing number of regular users.

GOALS

The basic product we want to create is a map of some ionospheric parameter (say, MUF(3000)), everywhere in the world, right now or in the near future.

GIRO and NOAA give us access to observations from a worldwide network of ionosondes (represented by the dots on the map), which can measure the characteristics of the ionosphere directly over some point on the Earth. But forming a global picture of the ionosphere using only this observational data is a challenge:

- There aren't that many ionosondes, and the ones we have are very clustered. There are large areas of the Earth's surface that are more than 1000km from an available ionosonde.
- Ionosondes take readings at different intervals, and have different amounts of delay in reporting their results.
- Ionosonde measurements are subject to noise caused by radio interference or ionospheric conditions such as Sporadic-E.

Besides observational data, we also have access to the International Reference Ionosphere (IRI-2016) computer model. IRI is an empirical model drawing on decades of observation and research, and it can predict ionospheric parameters for any time and any place on Earth.

The drawback of IRI is that it is a large-scale *climate* model. Given the same geomagnetic latitude and longitude, the same position of the sun in the sky, and the same level of solar activity (SSN or SFI), it will report the same average conditions every time. The real ionosphere is subject to short-term and small-scale variations that IRI can't capture.

In combining observational data with the IRI model to produce a global map we want to satisfy these goals:

1. Predicted values that are near (in time and space) to actual observations should be near the observed values.
2. Predicted values that are far from observations should be *consistent* with observed values.
3. Predicted values should follow plausible contours in time and space — no featureless voids in the middle of the ocean, and no sudden discontinuities.

COMPUTATIONAL APPROACH

1. Acquire observations from GIRO and NOAA and store them in a database.
2. Compute an "effective" solar flux that minimizes the difference between IRI and recent observations.
3. For each ionosonde, consider the recent history of the delta between observation and IRI prediction, and extrapolate the trend to the present and the near future.
4. Take the extrapolated variations-from-climate from all of the stations, and interpolate them in space over the surface of the earth. Add the interpolated deltas to the IRI prediction.
5. Export the data for presentation to the user

1. Loading

(Components: `noaa-loader` and `giro-loader`)

Ionosonde data comes from two sources. Data from NOAA NCEI is pushed via FTP to my server in SAO4 format and loaded using a Perl script I wrote for the purpose. Data from GIRO is fetched from the DIDBase API every 15 minutes using a Python script originally written by AF7TI. Both are stored in a PostgreSQL database containing ionospheric metrics (foF2, hmF2, MUF(3000), etc.), observation times, confidence scores, and station names and locations.

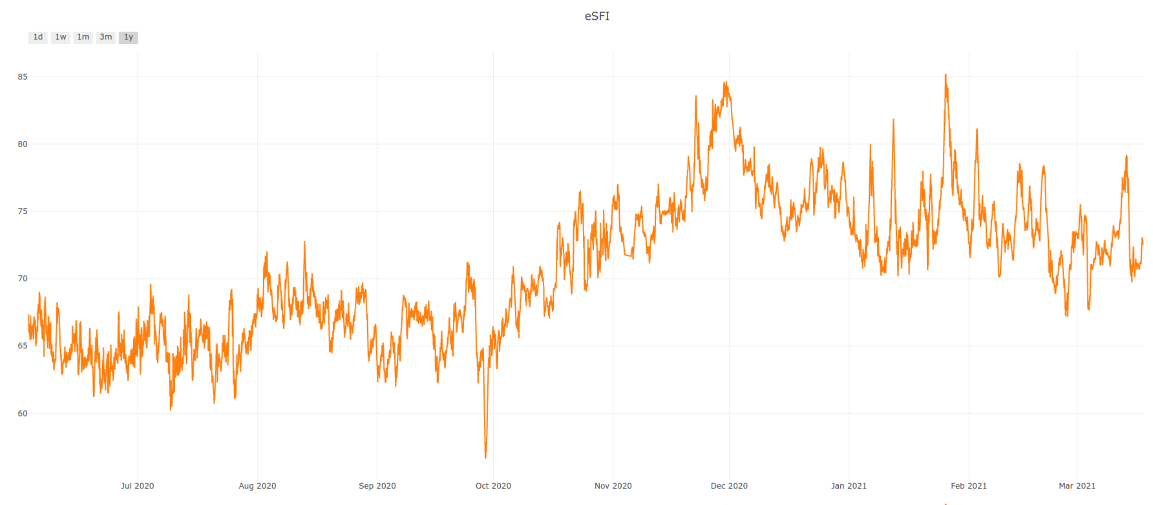
2. Effective Solar Flux

(Component: `essn`)

The past 25 hours of observations are loaded, and low-confidence measurements discarded. Measurements are weighted by geomagnetic latitude (mid-latitude > equatorial > polar), local solar time (peaking at 14:30 local), and scaler confidence score. The weights are windowed in time by a Tukey window with a length of 25 hours and a taper of 2 hours.

These weighted observations are then used to build a loss function for a black-box optimizer (`scipy.optimize.minimize_scalar`) which finds the value of the SSN parameter that minimizes a weighted sum of deltas of foF2, hmF2, and MUF(3000).

This procedure is similar to the Northwest Research Associates SSNe (Secan & Wilkinson 1997), but modified to smooth the weights, and by using a black-box optimizer to invert IRI instead of directly inverting the URSI foF2 function. The length and shape of the 25-hour window were chosen to minimize diurnal effects, which are a noted issue for NWRA.

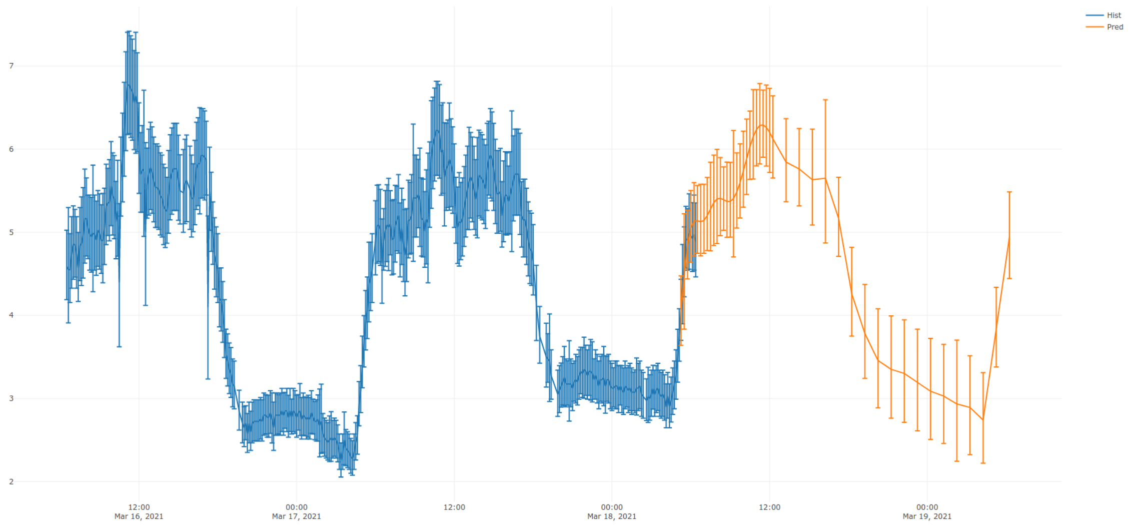


3. Temporal Smoothing & Extrapolation

(Component: `pred`)

The past 14 days of observations are loaded for each station. For each observation, the IRI model is used to produce a prediction for the observed date and time and the station location, using the effective solar flux computed in step 2. Deltas of the form `log(observed) - log(predicted)` are produced for the parameters foF2, hmF2, and MUF(3000), and scaler confidence scores are used to synthesize a log-standard-deviation for each observation. The `george` package is used to create a Gaussian Process Regression model from this data, which is queried for the current time as well as 1, 2, ... 24 hours into the future. Each prediction is in the form of mean ± stdev of a log-transformed value, and is stored in the database.

The Gaussian Process covariance kernel contains diurnal and semidiurnal periodic components, as well as RBF and rational-quadratic terms, and was derived using log-marginal-likelihood maximization on a sample of available data. This allows the model to capture daily periodic variations (using previous days to predict the next day) as well as trends and enforce a locally smooth result. The stdev values capture the model's "degree of confidence", which decreases as it predicts further into the future, or when less data is available or the data is noisier.



4. Spatial Smoothing & Assimilation to IRI

(Component: `assimilate`)

The extrapolated deltas from step 3 for a given target time are loaded, and the ionosonde locations are converted from latitude/longitude to 3D Cartesian coordinates. The deltas, their associated standard deviations, and coordinates are passed to `george` to create another Gaussian Process Regression model, this one using a 3D isotropic kernel which is the sum of an RBF kernel and a Matérn(v=5/2) kernel, giving a spatially smooth output. This model is queried on a 1°x1° grid and the deltas are applied to the IRI model to create a final prediction.

In the case of the MUF(3000) metric, the above procedure is applied to the parameter M(3000) = MUF(3000) / foF2, and then the M(3000) and foF2 outputs are multiplied to get the MUF(3000) output.

All interpolation is done in log-space, which eliminates the need to deal with the differing scale factors of the different parameters, and means that the model will never produce an unphysical negative result.

The use of nonzero "standard deviations" for the input data (on-diagonal elements of the covariance matrix) means that this is not a strict interpolation — the output isn't required to exactly equal the extrapolated station values at station locations, but varies according to the weight of nearby evidence. As that evidence decreases, the error signal regresses to zero, leaving the IRI base, satisfying the goals of consistency and plausibility.

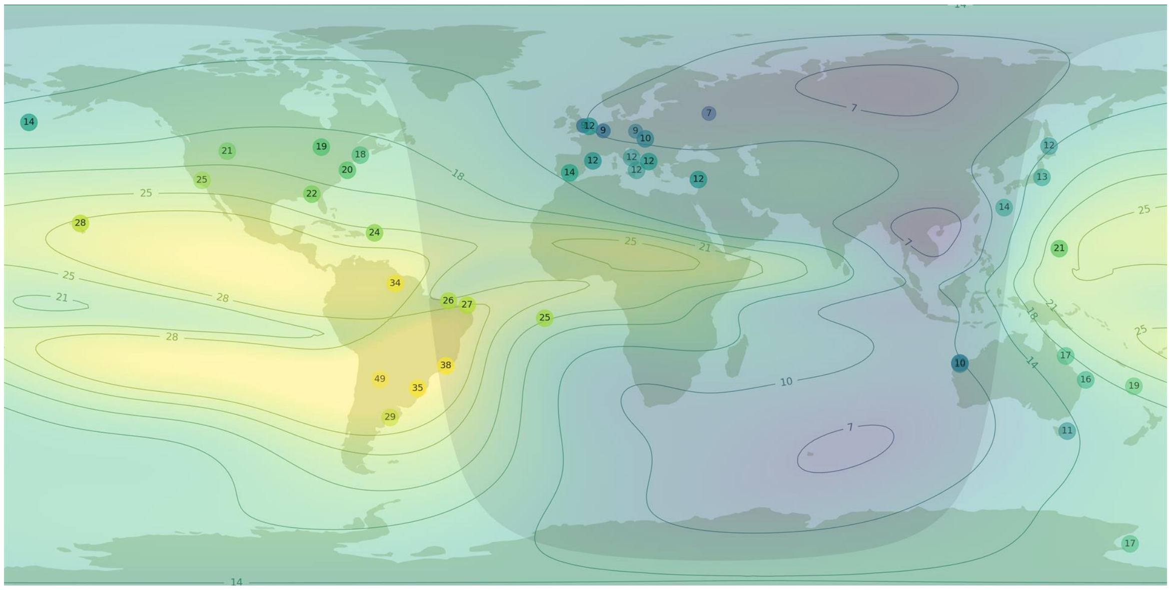
The transformation to 3D and the use of "chordal distance" rather than great-circle distance is supported by (Guinness & Fuentes 2016).

RENDERING AND RAYTRACING

(Components: `renderer` and `raytrace`)

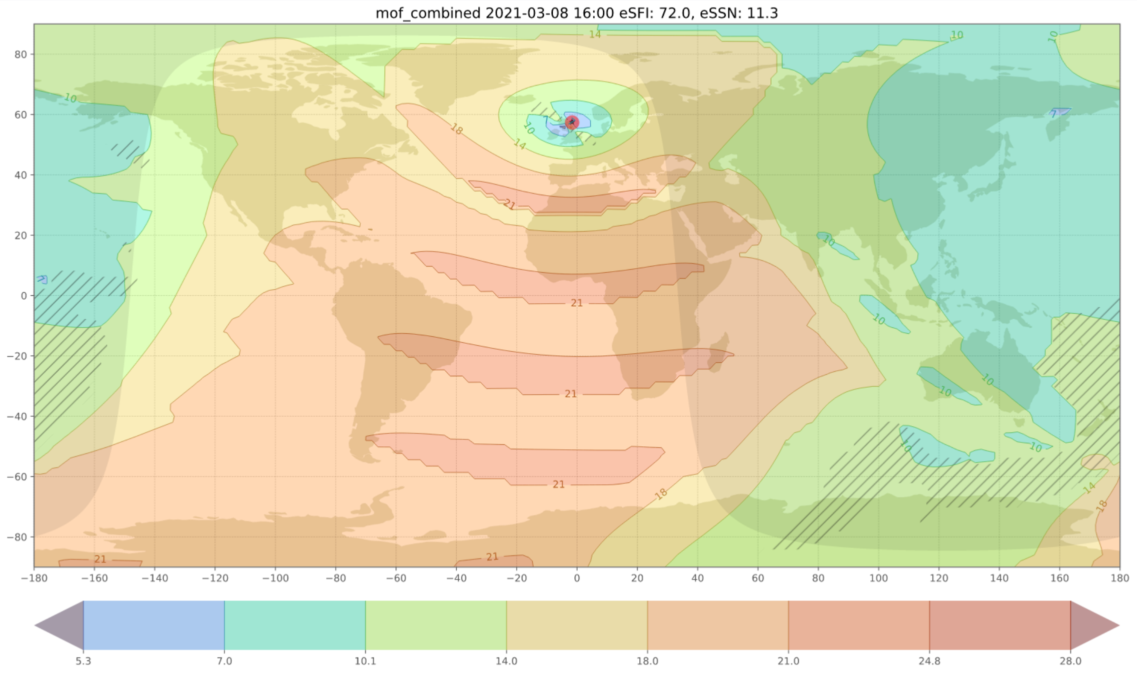
The results of assimilation are stored in the database as HDF5 tables, which can be accessed directly using an API. But to give people access to information at a glance, we need graphics.

Overview Maps



Produced for MUF(3000) and foF2, for the current time through +24 hours (but only current time is currently visible through the site). Continent map and day/night shading are provided by cartopy for orientation. Predicted MUF/foF2 is plotted as a continuous colormap, and contours are added for each amateur band. Latest measured data (for nowcast) or predictions (for forecast maps) is added as colored dots, and shaded based on confidence. Images are served as SVG, for bandwidth efficiency.

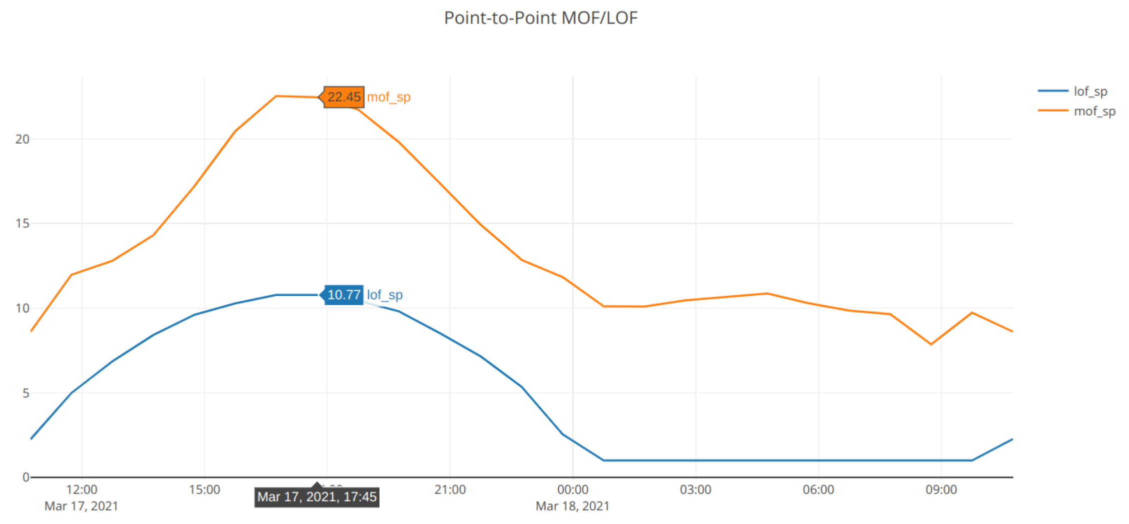
Raytraced Area Maps



Currently in development. For a given station location we can compute MUF and LUF to everywhere in the world and present it as a map. The engine is based on MOFDOS/MOFWIN by Dave Robbins K1TTT, I rewrote its raytracing core using numpy and numba, and replaced its ionospheric model with queries against my assimilated data. MUF is a relatively straightforward geometric control-point algorithm, and LUF uses integrated D-layer loss (Lucas et al 1993). No attempt is currently made to predict received signal power or SNR, to account for other forms of loss, or to handle anything other than F-hops.

Both short-path and long-path are computed, and a combined map is available that uses shading to indicate when the long path is superior (higher MUF or lower LUF) to the short path.

Point-to-Point Prediction



Using the ray-tracing engine and the predicted ionosphere model, it is also possible to produce a plot of the predicted MUF/LUF for two points over the next 24 hours, which allows choosing the best time to make a contact, or the best frequency for a given time.

SOFTWARE CREDITS

Without all of these open-source projects, my own would probably never exist.

- NumPy (https://numpy.org/)
- SciPy (https://www.scipy.org/)
- Pandas (https://pandas.pydata.org/)
- numba (https://numba.pydata.org/)
- h5py (https://www.h5py.org/)
- scikit-learn (https://scikit-learn.org/stable/)
- scikit-optimize (https://scikit-optimize.github.io/stable/)
- Fortran::Format (https://metacpan.org/pod/Fortran::Format)
- George (https://george.readthedocs.io/en/latest/)
- IRI2016 (http://irimodel.org/) (packaged by space-physics (https://github.com/space-physics/IRI2016))
- WMM2020 (https://www.ngdc.noaa.gov/geomag/WMM/) (packaged by space-physics (https://github.com/space-physics/wmm2020))
- IGRF-13 (https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html) (packaged by space-physics (https://github.com/space-physics/igrf))
- Flask (https://flask.palletsprojects.com/)
- SQLAlchemy (https://www.sqlalchemy.org/)
- uWSGI (https://uwsgi-docs.readthedocs.io/en/latest/)
- Mojolicious (https://mojolicious.org/)
- Minion (https://docs.mojolicious.org/Minion)
- Plack (https://metacpan.org/pod/Plack)
- Starman (https://metacpan.org/pod/Starman)
- matplotlib (https://matplotlib.org/)
- cartopy (https://scitools.org.uk/cartopy/)
- svgo (https://github.com/svg/svgo)
- Podman (https://podman.io/)
- PostgreSQL (https://www.postgresql.org/)

FUTURE DIRECTIONS

- Incorporate other sources (PSWS, RBN, WSPR, PSKReporter) which can provide higher-resolution information, and information in locations where ionosondes are unavailable.
- Improved ray-tracing with better accuracy and more useful outputs.
- Improved storm-time modelling (IRI's isn't super impressive).
- Make ray-tracing and prediction available through the web UI (interested developers wanted!)
- Replace the separate temporal and spatial GP models with one big 4-D GP model
  - Makes better use of data
  - Can capture a correlation between "conditions here, now" and "conditions 15 degrees west of here, an hour from now"
  - Challenge: very memory-intensive, especially for kernel optimization
- GPU optimization to handle more ray-tracing traffic.

REFERENCES

Secan, J. A., and Wilkinson, P. J. (1997), Statistical studies of an effective sunspot number, Radio Sci., 32(4), 1717– 1724, doi:10.1029/97RS01350 (http://dx.doi.org/10.1029/97RS01350).

Guinness, J. and Fuentes, M. (2016), Isotropic covariance functions on spheres: Some properties and modeling considerations. Journal of Multivariate Analysis, 143, 143-152. doi:10.1016/j.jmva.2015.08.018 (http://dx.doi.org/10.1016/j.jmva.2015.08.018).

Lucas, D. & Prinson, G. & Headrick, J. & Thomason, J.. (1993). RADARC HF ionospheric prediction program for OTH radar. Defense Technical Information Center Memorandum ADA269557.