FITACF (version 1) description

Pasha Ponomarenko and Colin Waters, 2005-2006

Chapter 1

Introduction

This work was necessitated by two factors:

- Since its inception in late 1980s FITACF code remains pretty much a "black box" for the wast majority of the SuperDARN data users.
- During last decade extensive but unsuccessful attempts were made to explain large spectral width $W \geq 200$ m/s values regularly observed by SuperDARN radars at high latitudes.

At SuperDARN'04 in Saskatoon the authors initiated a discussion concerning the above topics by presenting a poster "Possible causes of large spectral width in SuperDARN echoes from high latitudes". This effort resulted in creation of the "FITACF workgroup" charged with documenting FITACF algorithms. However, the group activity was not very productive so far. We attribute this to strong de-centralization of the group, when each member was asked to "de-cipher" a separate portion of the code. From the very nature of the FITACF package, it requires more general(ising) approach considering the multitude of the tasks performed.

On our part, we started from plotting a flow-chart, which allows to generally understand tasks performed by different procedures and connections between them. Then we focused on the procedures, which are involved in estimating W. Obviously, during this process we **ought** to learn about details of FITACF algorithms and their implementation. After documenting how exactly major FITACF algorithms have been implemented, we searched for possible factors artificially increasing spectral width estimates, found several of them, and studied experimentally their relative contributions to W. As a result, several modifications to FITACF have been proposed, which considerably improved physical validity and accuracy of the output parameter estimates, including spectral width, power and Doppler velocity. This results allowed us to propose a viable physical explanation to the large spectral width values, but this topic lies outside the scope of the report.

Chapter 2

Flow-chart

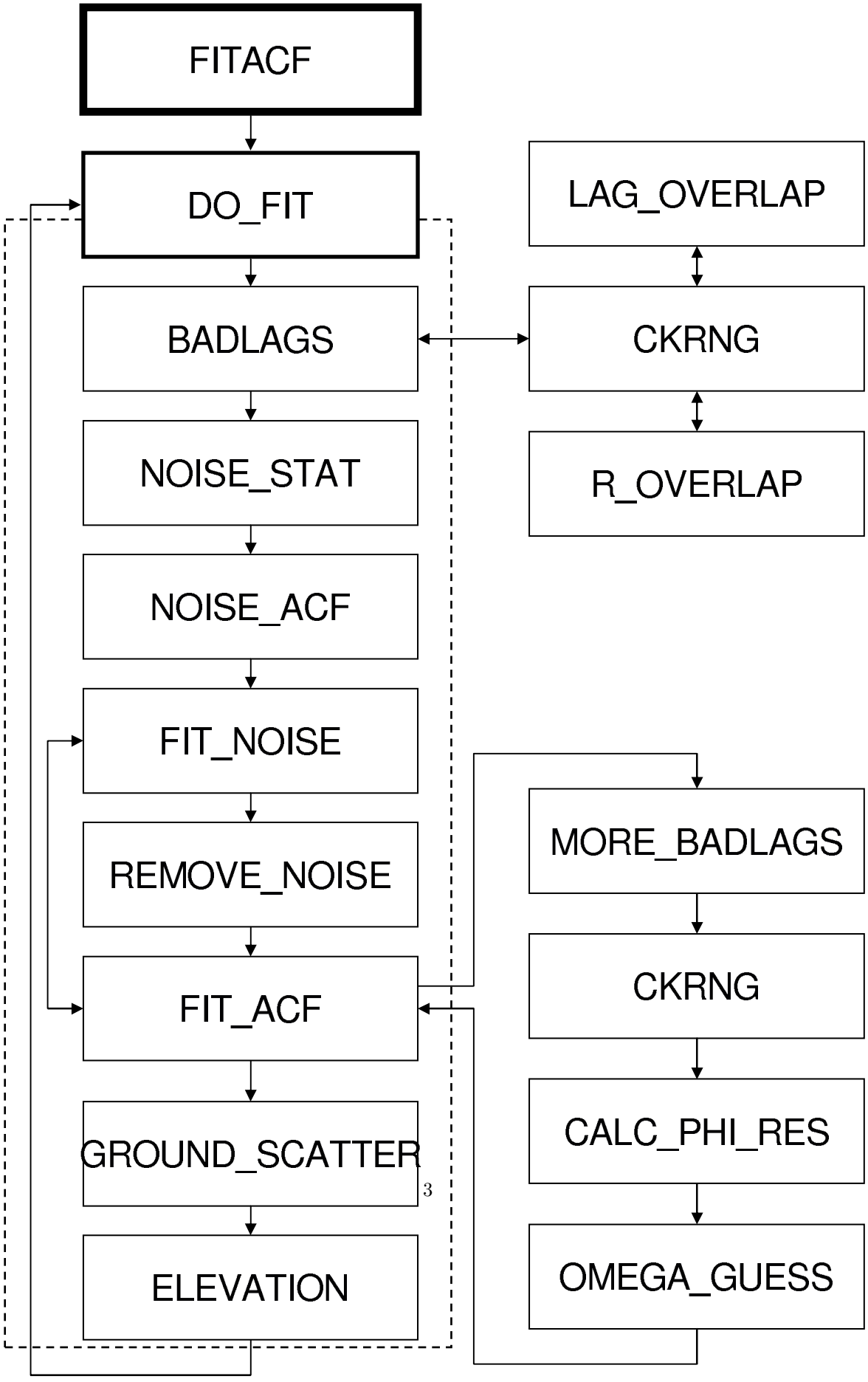
This flow-chart was plotted based on our analysis of the FITACF package (version 1.09) written in C and provided by Simon Shepherd (JHUAPL).

It shows that some procedures (e.g. CKRNG and FIT_ACF) are used twice.

The main philosophy of programm is following. The data block which is being processed at once consists of a single record read from DAT file. I.e., it represents complex ACFs measured for all the range gates recorded at certain time by a certain beam (gate number X 1).

The FITACF package

- creates corresponding arrays and reads data into them,
- marks ACF lags contaminated by cross-range and pulse-overlap interference
- determines reference noise parameters
- checks for presence of the coherent interference (broadcasting stations etc) and, if any present, attempts to remove this component from ACFs
- marks "bad lags" based on the ACF power shape assumptions (non-increasing power)
- fits model curves to ACF power and phase to estimate output parameters and respective errors
- marks ACFs with ground/sea scatter
- calculates elevation from interferometer data (XACF)
- writes output parameters into a FIT file.


FITACF
DO_FIT
BADLAGS
NOISE_STAT
NOISE_ACF
FIT_NOISE
REMOVE_NOISE
FIT_ACF
GROUND_SCATTER
ELEVATION
LAG_OVERLAP
CKRNG
R_OVERLAP
MORE_BADLAGS
CKRNG
CALC_PHI_RES
OMEGA_GUESS

Chapter 3

Detailed description of FITACF procedures

- *BADLAGS* produces identified bad receiver samples overlapped by transmitted pulses (unchanged for a given pulse sequence) and, via *R_OVERLAP*, cross-range interference table (identical for all gates). *CKRNG* uses these tables to find bad lags for a given range set of ACFs.

  Functions *BADLAGS* and *CKRNG* are stored in the same file called *BADLAGS.c*. Functions *LAG_OVERLAP* and *R_OVERLAP* are stored in *LAG_OVERLAP.C*.

  - Determining bad lags due to transmitter pulse overlap. Criterion: bad pulses appear within (pulse length + XR switch time) after beginning of the transmitter pulse.(*BADLAGS*)
  - Determining bad lags due to cross-range interference. Criterion: the maximum *lag0* power from the checked gate should exceed *(lag0 power)*(0.3*nave)* power from any interfering gate. (*LAG_OVERLAP (RANG_BADLAGS)*)

- Determining noise level (*mnpwr*): look for the lowest 10 values of *lag0* power and average to get the noise level. Ignore values that are exactly 0. If you can't find 10 usable values within the first 1/3 of the sorted power list, then just use whatever you got in that first 1/3. If you didn't get any usable values, then use the *NOISE* parameter from the "clear sky" scan.(*DO_FIT*)

- Determining the level which will be used as the cut-off power for *FIT_ACF* (*noise_pwr*): this is the average power at all non-zero lags of all acfs which have *lag0* power $< 1.6 * mnpwr + 1$ standard deviation from that average power level.(*NOISE_STAT*)

- Finding whether there is a coherent component in the noise (broadcasting stations etc.): the average level of the non-zero lags (see previous point) is compared with its variance *var*. If "signal" $> var * \sqrt{3}$, one might suspect that there is a coherent interference in the data. (*NOISE_STAT*).

- If there is some coherent component, determine an average "noise acf" (*NOISE_ACF*). This is being done through separate averaging of $\Re$ and $\Im$ parts of ACFs, which correspond to following selection criteria:

  - *lag0* power for all the ACFs $< 1.6 * mnpwr$

- $abs(\Re(lag0)) + abs(\Im(lag0)) > 0$
- $abs(\Re(lag0)) < 1.6 * mnpwr$ and $abs(\Im(lag0)) < 1.6 * mnpwr$(repeat?)
- After that the bad lags due to transmitter pulses are determined (again?).
-

Average non-zero-lag power *ave_noise_pwr* for the obtained noise ACF is calculated (*DO_FIT*).

- Compare the *ave_noise_pwr* (coherent) in the noise ACF with the *average noise_pwr* (incoherent) calculated from all the noise ranges independently. If the average power in the noise acf is greater than 1/2 of the independent average, then there is a significant coherent signal that should be removed. Otherwise, there is no coherent signal, so the noise removal is skipped. The fitted noise ACF is determined for both real and imaginary parts of the noise ACF (*FIT_NOISE*) and then extracted from original ACFs. In this process the envelope is estimated from the fit to the experimental ACF power and the carrier was calculated from the estimated velocity (phase slope).
- Removing the fitted noise ACF from all the ranges if necessary (*REMOVE_NOISE*).
- Calculating log signal/noise ratio $P\_0 = 10 * log10((lag0Power - skynoise)/skynoise)$ – one of the output parameters (*DO_FIT*).
- Fitting procedure (*FIT_ACF*): extracting *noise_pwr* from the ACF envelope.
  - If the resulting power < 0, then its value is set to 0.1.
  - If the resulting *lag0* power$*(1 - 1/\sqrt{nave}) < noise_pwr$, no fitting to be run, and the ACF is marked as "bad".
- Finding bad lags due to non-monotonous ACF power (*MORE_BADLAGS*) This is being used in *FIT_ACF* after extracting background noise level and noise ACF if any found. The MORE_BADLAGS routine is designed to adjust the ACF according to a single-component shape assumed by the fitting routines. Importantly, before this the fluctuation level, σ_R, and non-zero lag noise, $\langle R_n(\tau \neq 0)\rangle$, are extracted from $|R(\tau)|$, which may lead to negative values for ACF power. ACF lags are marked as "bad" according to the following rules:
  - ACF power is less than zero, $|R(\tau)| \leq 0$.
  - If two consecutive points in the ACF power are less than the zero, then the longer lags ("tail") are dismissed:

$$R(\tau_i) < \langle R_n(\tau \neq 0)\rangle \text{ and } R(\tau_{i+1}) < \langle R_n(\tau \neq 0)\rangle \rightarrow$$

$$R(\tau_j) \equiv \text{ "bad" for } j > i + 1.$$

  - If the ACF power increases with increasing lag outside the uncertainty limit determined by the fluctuation level and twice the non-zero lag noise, then this lag is declared "bad":

$$R(\tau_i) > R(\tau_{i-1}) + R(0)/\sqrt{2N_{avg}} + 2\langle R_n(\tau \neq 0)\rangle \rightarrow$$

$$R(\tau_i) \equiv \text{ "bad"}.$$

    This effectively removes positive "spikes" in the ACF power and caters for the decaying shape to comply with the single component fitting models. However, if the following lag is

"well-behaved", then the previous lag is considered as a negative "spike", and the current lag is relabelled as "good":

$$R(\tau_{i+1}) < R(\tau_i) + R(0)/\sqrt{2N_{avg}} + 2\langle R_n(\tau \neq 0)\rangle \rightarrow$$

$$R(\tau_{i-1}) \equiv \text{ "bad"},$$

$$R(\tau_i) \equiv \text{ "good"}.$$

- Checking the total number of good lags for this ACF. If it is $<$ *lag_lim* (usually equal to 0?????), then no fitting to be done.
- Extracting *lag0* power$*(1/sqrt(NAVE))$ from the ACF power and checking for new bad lags. If the resulting power < 0 then it is set to 1 (good for logarithms).
- Phase fit Lags with a power level below noise and fluctuation levels are set to 0.1 or 1 and effectively removed from the fitting process, where weighting is being done by power. "Bad power lags" are determined as following
  - Below ($P0 + noise\_lev$) - power set to 0.1 (229)
  - Below ($P0 + 2 * noise\_lev$) - *bad_pwr* is set to 1 (240-241)
- Calculation of the residual phase (264, *CALC_PHI_RES*): For badlag = 0 it calculates arctangent from a complex ACF taking into account quadrants.
- Local frequency (268, *OMEGA_GUESS*): It calculates estimate of the single-component frequency based on closely-separated ACF lags (separation 1 or 2 elementary lags). Only good lags are used for this purpose (*badlag* = 0, line 79 in *OMEGA_GUESS*) Pairs of ACF phase values are corrected by extracting or adding 2π if the phase difference is larger than π (85-86).
- Individual phase estimates are weighted by the average ACF power (uncorrected, i.e., straight from acf) and summed (91), as well as their mean square (92) used to calculated estimate error. Average frequency and error are calculated via dividing by the summed average power (98-110).
- Velocity estimate errors are calculated as a geometrical me4an of the fitting errors (least square residuals) and standard deviation of the residual phase (see above).
- Ground scatter.

  Until FITACF version 1.15 the empirical criteria in FITACF used to reject sea scatter echoes were

  $$|W| - W_{err} < 35\text{m/s}$$

  and

  $$|V| - V_{err} < 30\text{m/s},$$

  where W_{err} and V_{err} are the fitting errors for the spectral width and velocity, respectively. In version 1.15 Stefan Sundeen, Gerard Blanchard and Kile Baker imnplemented an improved algorithm based on statistical studies of velocity-width distribution and their combined error estimate.

  The analysis inluded all data from 12 days (with large amounts of both ionospheric scatter and ground scatter) distributed over all seasons of the year. Two radars were used in the analysis, Saskatoon and Kapuskasing.

The result of the initial statistical analysis showed that scatter was most likely ground scatter if

$$|V| < GS_V^{max} - (GS_V^{max}/GS_W^{max}) \cdot |W|,$$

where

$$GS_V^{max} = 30 \text{ m/s}$$

and

$$GS_W^{max} = 90 \text{ m/s}.$$

Let $g(|V|, |W|) = (|V| - GS_V^{max} - (GS_V^{max}/GS_W^{max}) \cdot |W|)$

Then, if was assume the errors in V and W are independent, we can estimate the error in the function g to be:

$$g_{err}/g = \sqrt{(V_{err}/|V|)^2 + (W_{err}/|W|)^2}$$

Then the final condition for flagging data as ground scatter is:

$$\text{if } g - g_{err} \leq 0 \text{ then flag as ground scatter.}$$