

# ElSpec

Ilkka Virtanen  
Ionospheric Physics research unit  
University of Oulu, Finland

Version 0.1  
March 23, 2020

# License

## **ElSpec**

Copyright 2015-2018 Ilkka Virtanen and Björn Gustavsson

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	ElSpec . . . . .	4
1.2	System requirements . . . . .	4
1.3	Installation . . . . .	5
1.4	Help . . . . .	5
1.5	Reference . . . . .	5
<b>2</b>	<b>Electron energy spectrum estimation</b>	<b>6</b>
2.1	Electron precipitation and ion production . . . . .	6
2.2	ElSpec implementation . . . . .	7
2.2.1	Integration of the electron continuity equation . . . . .	7
2.2.2	Spectrum models . . . . .	8
2.2.3	Model selection . . . . .	8
2.2.4	Error estimation . . . . .	9
2.2.5	Derived parameters . . . . .	9
<b>3</b>	<b>ElSpec in practice</b>	<b>10</b>
3.1	The simplest example . . . . .	10
3.2	Input data . . . . .	10
3.2.1	Both ppdir and fitdir are defined . . . . .	11
3.2.2	Only fitdir is defined . . . . .	11
3.2.3	Only ppdir is defined . . . . .	11
3.3	Running the GUISDAP analysis . . . . .	12
3.4	Ion production models . . . . .	12
3.5	Ion recombination models . . . . .	12
3.6	Time resolution . . . . .	13
<b>4</b>	<b>Examples</b>	<b>14</b>
4.1	UHF beata with raw densities . . . . .	14
4.2	UHF beata with a steady state model . . . . .	17
4.3	A fast inversion with fitted data . . . . .	17

<b>A</b>	<b>Command reference</b>	<b>23</b>
A.1	ElSpec . . . . .	23
A.2	ElSpecPlot . . . . .	28
<b>B</b>	<b>GNU General Public License</b>	<b>31</b>

# Chapter 1

## Introduction

### 1.1 ElSpec

ElSpec is a MATLAB toolbox for estimating differential energy flux of precipitating electrons from EISCAT incoherent scatter radar observations of ionospheric electron density. The ElSpec input data are analysis results from the standard EISCAT data analysis tool, GUIDAP. The input data may be GUIDAP raw densities, GUIDAP fit results, or both. The software assumes that the radar is in magnetic field-aligned position. Several different models of ion production, recombination, and ion composition are available. Also time-resolution and energy grid points can be selected by the user, the former within the limits set by EISCAT experiment design.

ElSpec is an upgraded version of the electron energy spectrum estimation technique developed by Björn Gustavsson, which was used in (*Dahlgren et al.*, 2011). The upgraded technique and validation results are published in (*Virtanen et al.*, 2018)

### 1.2 System requirements

ElSpec has been run on both Mac and Linux with MATLAB2017b. Other reasonably recent MATLAB versions and other operating system will probably work as well.

ElSpec uses the MATLAB implementation of the NRLMSISE-00 model in the Aerospace toolbox, `atmosnrlmsise00`. This version of ElSpec cannot be used without the Aerospace toolbox.

## **1.3 Installation**

ElSpec is distributed as a MATLAB toolbox. Add full path to the 'ElSpec' directory in your MATLAB search path or, if you received the package as an .mltbx-file, install it using the MATLAB toolbox installer (a double-click is typically enough).

## **1.4 Help**

Standard MATLAB help messages are available at least for the user interface functions. Full command references for the main inversion and plotting routines are given in Appendix A of this document.

## **1.5 Reference**

In published scientific works using ElSpec, please refer to the paper:

Ilkka I. Virtanen, Björn Gustavsson, Anita Aikio, Antti Kero, Kazushi Asamura, and Yasunobu Ogawa, Electron energy spectrum and auroral power estimation from incoherent scatter radar measurements, *J. Geophys. Res. Space Physics*, 123, 6865-6887, 2018, doi:10.1029/2018JA025636.

## Chapter 2

# Electron energy spectrum estimation

The electron energy spectrum inversion is shortly outlined in this chapter. See (Virtanen *et al.*, 2018, and references therein) for details.

### 2.1 Electron precipitation and ion production

Energetic electrons precipitating to the ionosphere cause collisional ionization. If plasma convection and photoionization can be neglected, the electron gas will follow the continuity equation

$$\frac{\partial N}{\partial t} = Q + \alpha N^2, \quad (2.1)$$

where  $N$  is the electron number density,  $t$  is time and  $Q$  is electron production rate per unit volume.

The ion production rate depends on composition of the neutral atmosphere and the differential number flux of the precipitating electrons. If the neutral atmosphere is known,  $Q$  can be calculated for arbitrary electron fluxes and altitudes. Similarly, if the ion composition is known, the effective recombination coefficient  $\alpha$  can be calculated.

Incoherent scatter radars can observe electron density as function of time and altitude along a geomagnetic field-line. At night the assumptions of negligible photoionization and plasma convection are reasonable in the E region. Because the ion production rate can be expressed as a function of the differential number flux  $I$ ,  $Q = Q(I(t))$ , the differential number flux can be inverted from the incoherent scatter radar data.

## 2.2 ElSpec implementation

Key points of the ElSpec inversion are *analytic integration of the electron continuity equation*, *fits with multiple models* of the spectrum shape, and *model selection* by means of the Akaike information criterion. These are shortly outlined below.

### 2.2.1 Integration of the electron continuity equation

Input data to ElSpec are vectors of electron density at discrete height-gates  $h_i$ , sampled at times  $t_j$ ,

$$\mathbf{N}_j = \begin{pmatrix} N_{j,1} \\ N_{j,2} \\ \vdots \\ N_{j,H} \end{pmatrix}, \quad (2.2)$$

where  $H$  is the number of height gates.

Ion production rate in each gate are calculated as matrix product

$$\mathbf{Q}_j = \mathbf{A}\mathbf{I}_j, \quad (2.3)$$

where  $\mathbf{Q}_j$  is a vector of ion production rates, elements of the matrix  $\mathbf{A}$  are calculated using an ion production model, and  $\mathbf{I}_j$  is the differential number flux. In a similar manner, the effective recombination coefficients in each height gate are collected in a vector

$$\boldsymbol{\alpha}_j = \begin{pmatrix} \alpha_{j,1} \\ \alpha_{j,2} \\ \vdots \\ \alpha_{j,H} \end{pmatrix}. \quad (2.4)$$

A key point in the ElSpec implementation is that the time-derivative of electron density  $\partial N/\partial t$  is not explicitly calculated from data, but the software makes an iterative search to find the differential number flux  $\mathbf{I}$  that leads to best match between the observed electron density and the analytic solution from Equation (2.1). Electron density as function of time, as solved from (2.1), is

$$n(t) = \frac{n(t_{j-1}) + \sqrt{Q_j/\alpha_j} \tanh\left(\sqrt{\alpha_j Q_j} t\right)}{1 + \sqrt{\alpha_k Q_j} n(t_{j-1}) \tanh\left(\sqrt{\alpha_j Q_j} t\right)}, \quad t_{j-1} \leq t \leq t_j. \quad (2.5)$$

Here  $n(t)$  is the electron density and  $n(t_{j-1})$  is the density at beginning of the current time step. Because the radar observations are averages over a time step,



the model values, which should match with the observed densities, are integrated from (2.5),

$$\mathbf{n}_k = \frac{1}{\alpha_k \Delta t} \left( \log \left( \frac{\mathbf{n}(t_{k-1}) \sqrt{\alpha_k / \mathbf{Q}_k} \tanh(\sqrt{\alpha_k \mathbf{Q}_k} \Delta t) + 1}{\tanh(\sqrt{\alpha_k \mathbf{Q}_k} \Delta t) + 1} \right) + \sqrt{\alpha_k \mathbf{Q}_k} \Delta t \right). \quad (2.6)$$

## 2.2.2 Spectrum models

The inverse problem in electron energy spectrum estimation is ill-posed in general, because large energy interval needs to be covered with rather high resolution, but typically only some tens of electron density observations are available for each time step. Regularizing information is thus needed, and this information is included in the form of parametric spectrum shape models in ElSpec.

There is a variety of models one could potentially use. For example, Maxwellian and  $\kappa$  distributions, as well as narrow- and wide-band accelerated spectra are observed by satellites above and within the ionosphere. The original implementation in *Dahlgren et al.* (2011) used an exponential of a spline expansion, with both the spline node locations and their amplitudes determined in the inversion.

ElSpec uses a slightly different polynomial model, because the model of *Dahlgren et al.* (2011) reduces to a polynomial when number of spline nodes is small, which is almost always the case, and the polynomials are technically somewhat easier to control. The polynomial model is of the form

$$I = \mathbf{E} \exp \left( \sum_{l=0}^L P_L(l) \mathbf{E}^l \right), \quad (2.7)$$

where  $P_L$  is a vector of  $L + 1$  polynomial coefficients, which are the unknowns in the inversion. The vector  $\mathbf{E}$  is the selected energy grid. The model can produce exact Maxwellian fluxes, as well as slightly distorted ones with  $L = 1$ , and more complex shapes, such as  $\kappa$  distributions, with higher  $L$ .

## 2.2.3 Model selection

On each time-step, ElSpec runs a separate fit for each spectrum shape model with  $L = 1, 2, \dots, N_{max}$ , where  $N_{max}$  is typically between 5 and 10. The optimal one, which is then selected as the final estimate of the differential number flux, is then selected by means of the Akaike information criterion (e.g. *Burnham and Anderson*, 2002).

### 2.2.4 Error estimation

Error estimation in ElSpec is somewhat complicated, because the error in the modeled electron density in beginning of the integration must be taken into account. An error estimation based on an assumption of Gaussian error distributions and numerical derivatives of linearized theory in vicinity of the iteration convergence point is implemented.

### 2.2.5 Derived parameters

Several derived parameters and their error estimates are calculated from the differential number flux estimates  $I_j$  and their covariances.

**Differential energy flux** is calculated as a product of the differential number flux and the known energy grid values.

**Upward field aligned current** carried by the precipitating electrons is integrated from the differential number flux. However, this estimate might not reflect the true field-aligned current, because ElSpec cannot see either upward moving electrons or low-energy electrons (typically below 1 keV), whose contribution to the net current may be significant.

**Net energy flux** of the precipitation is integrated from the differential energy flux. These estimates are accurate, because contribution of the low-energy electrons is very small. The net energy flux estimates have been found to be very accurate even if the shape of the differential energy flux is not.

# Chapter 3

## ElSpec in practice

The user interface to the ElSpec inversion is the function `ElSpec`, which can be called with various optional parameters. The parameters, their default values, and examples of their use are listed in Appendix A. This chapter explains most important parts of the analysis in more detail.

### 3.1 The simplest example

The simplest, but typically suboptimal way to use ElSpec is to use standard fitted GUISDAP analysis results as input. Because the default settings are for EISCAT UHF beata experiment, more or less reasonable results will be produced if one cd's to a GUISDAP output directory of any field-aligned (CP1) UHF beata experiment and runs the command

```
>> ElSpecOut = ElSpec('fitdir', '.');
```

The software will print its settings, including the output file name. The output file contains a MATLAB struct 'ElSpecOut', where all settings and the inversion results are stored. The same structure is returned by the function when the analysis is complete. The results can be visualized with the function 'ElSpecPlot'. In the simplest case a plot is produced with the command

```
>> ElSpecPlot(ElSpecOut);
```

### 3.2 Input data

ElSpec can use two kinds of EISCAT data as input, 'normal' GUISDAP fit results and high-resolution 'raw densities' from two specific experiments, UHF beata and arc1. The GUISDAP options for calculating the sufficient raw densities are given

in Section 3.3. The different input options are explained in the following three subsections.

### **3.2.1 Both `ppdir` and `fitdir` are defined**

If both `ppdir` and `fitdir` are given in the `ElSpec` function call, the electron density data are read from raw electron densities in `ppdir`. Electron temperatures are extracted from the fit results in `fitdir`.

In this case the `tres` argument is effective and has two alternatives. `dump` means that the raw densities from EISCAT data dumps are integrated together. Standard deviation of the electron density is calculated as an ensemble average over all raw density profiles in a data dump. `best` means that the original, best available time resolution is maintained. The same standard deviations are copied for all raw density profiles in each EISCAT data dump.

This combination is currently available only for EISCAT UHF 'beata' and 'arc1' experiments.

### **3.2.2 Only `fitdir` is defined**

If only `fitdir` is defined but `ppdir` is not, `ElSpec` reads both electron densities and electron temperatures from the same GUISDAP fit files. This option works with all EISCAT experiments, because the fit results are in a standard format.

Notice that temperature estimation is typically not possible with very high time resolutions. If GUISDAP is set to fit only electron densities, the temperatures in GUISDAP output files will be from the IRI model.

### **3.2.3 Only `ppdir` is defined**

If only `ppdir` is defined, `ElSpec` uses the raw electron densities in `ppdir` and takes the electron temperature from a model. Depending on the selected ion recombination model, this model is either IRI or MSIS. In the latter case, electron temperature is assumed to be equal to the neutral temperature.

The electron temperature is taken from the IRI model if it is needed for the recombination model. Recombination models (`recombmodel`) using IRI are 'Rees', 'SheehanGr' and 'SheehanEx' (the last one is probably not useful in practice).

This option is available only for EISCAT UHF 'beata' and 'arc1' experiments.

### 3.3 Running the GUIDAP analysis

GUIDAP default options are sufficient for calculating the normal fit results in `fitdir`. However, one should carefully calibrate the data, because the energy spectrum is a nonlinear function of the density, and calibration errors may have significant, nontrivial effects to the spectrum estimates.

In order to calculate the raw electron density estimates in `ppdir`, one should use the GUIDAP settings

```
analysis_ppshortlags=1  
analysis_pponly=1  
a_satch.do=0
```

The last one disables the GUIDAP satellite echo detection, which may sometimes cause problems with very high time resolutions. The analysis can be run also with the satellite check, but ElSpec cannot read in data files with data points missing due to satellite echo detections, and such files have to be manually removed. In addition, the analysis time-resolution should be matched with the EISCAT dump length, which is 4 s for UHF arc1 and 5 s for UHF beata. With these options the analysis does not produce the standard fitted parameters at all, but stores only the raw densities with highest available resolutions. ElSpec provides input routines for reading the experiment-specific GUIDAP outputs for UHF beata and arc1.

Power calibration is most practical for the standard fit results. The 'magic constant' estimated with the standard fit can (and should) be used also for the raw density fits.

### 3.4 Ion production models

ElSpec allows the user to select between two alternative ion production models. The first implementation used in *Dahlgren et al.* (2011) used an ion production model by (*Sergienko and Ivanov*, 1993), which can be selected with the `ionomodel` parameter. The default model is from a more recent work by (*Fang et al.*, 2010).

### 3.5 Ion recombination models

The effective recombination coefficient  $\alpha$  is a major issue in the energy spectrum inversion. The topic is problematic, because the coefficient depends on two poorly known things, the ion composition and the recombination coefficients of individual ion species.

The default choice is to use recombination speeds for ions in vibrational ground state from laboratory measurements (*Sheehan and St-Maurice*, 2004) and assume

that only  $\text{NO}^+$  and  $\text{O}_2^+$  ions, whose abundances are taken from IRI, are present. The recombination coefficients are different from e.g. the widely used values by *Rees* (1989), who did not provide different values for the vibrationally excited states. Although there is no practical means to monitor the vibrational state of the molecular ions, the excited states are short-lived in comparison to the ion recombination times, and vast majority of the ions should be in ground state.

The IRI compositions are known to be suboptimal, because the collisional ionization produces large amounts of  $\text{O}_2^+$ , whose contribution varies rapidly along changes in the precipitation flux. The incorrect compositions have been found to cause up to 30 % error in the net power flux estimates. In order to provide simpler models, which may be equally good in practice, the package contains options for assuming pure  $\text{NO}^+$  or pure  $\text{O}_2^+$  ionospheres.

Altogether nine different options for the ion recombination model are currently available. These are listed in Appendix A.

### 3.6 Time resolution

ElSpec contains a build-in system for controlling the time-resolution according to ionospheric conditions. When electron density is high and recombination fast, the time resolution will automatically drop to the shortest one allowed by the data. When the recombination is slow the user can allow ElSpec to fit to longer periods of data. The time resolution is determined for each time step and altitude separately, meaning that e.g. low densities at the upper E region may be analyzed with a coarser time resolution than the E region peak.

The adaptation to the recombination time is built in the system and cannot be directly accessed by the user. The adaptation is based on scaling the electron density variances according to the recombination time-scale, as explained in *Virtanen et al.* (2018).

However, the user can control the maximum amount of data used in each fit with the `ninteg` parameter. If this parameter is set to one, the adaptation is disabled and data from exactly one time step is always used.

# Chapter 4

## Examples

This chapter provides some practical examples of using ElSpec.

### 4.1 UHF beata with raw densities

Data path	/data/raw/2015/beata_cp1_2.0u_SP	Data path	/data/raw/2015/beata_cp1_2.0u_SP
Start time	2015 02 16 23 00 00	Start time	2015 02 16 23 00 00
Stop time	2015 02 17 05 00 00	Stop time	2015 02 17 05 00 00
Dsp expr	beata Vs2	Dsp expr	beata Vs2
Site	T	Site	T
Result path	/kaappi/EISCAT/RESULTS/res_2015.	Result path	/kaappi/EISCAT/RESULTS/res_2015.
Real time	RT	Real time	RT
Integration t	60	Integration t	5
Disp figures	0 0 0 0 0	Disp figures	0 0 0 0 0
Special	fit_altitude(1:5,1:2)=[0 Inf ; 80 Inf ; 103 Inf ; 0 0 ; 0 Inf]	Special	analysis_ppshortlags=1 analysis_pponly=1
GO	Reset Save Quit	GO	Reset Save Quit

Figure 4.1: Left: A GUIDAP GUI window with settings for a fit, which will produce the electron temperature estimates for ElSpec. Right: A GUIDAP GUI window with setting for a fit of the raw electron densities.

A default ElSpec use case is inversion of EISCAT UHF beata data, using both GUIDAP fits of all plasma parameters and raw electron densities.

First of all, a 'normal' GUISDAP fit with a rather coarse time resolution is needed. 60 s is typically enough for reasonable temperature estimation. The default height limits for temperature estimation are typically a bit high, and better electron temperature and density estimates can be gained if ion temperature is fitted above 80 km and the electron-to-ion temperature ratio above 103 km. The limits can be set with the GUISDAP `fit_altitude` option. An example of a GUISDAP GUI is given on the left panel of Figure 4.1.

After running the analysis, one should carefully calibrate the data, using either plasma lines or comparisons with the EISCAT dynasonde. For example, the dynasonde calibration can be done with the standard GUISDAP function `calib_ne`. Notice that ElSpec is typically used in conditions, where E region may have strong ionization but the F region is empty, so it may be better to calibrate against the E region peak densities. Also, the dynasonde calibration should be made with a time period where the ionosphere is more or less stable. Another option is to use the plasma line data in beata, which is often available because the electron precipitation heats the electron gas. The plasma lines have the benefit that they are from the very same volume with the ion line data.

The calibration will suggest a 'Magic constant'. If the value is different from that used in the original analysis, the analysis should be run again with the correct magic constant. The calibration should be checked again after the second analysis run, but is typically OK at this point. If not, one more analysis run might be needed.

When the data are correctly calibrated, high-resolution raw densities should be calculated from the same data. This is done by means of setting

```
analysis_ppshortlags=1
analysis_pponly=1
a_satch.do=0
```

in GUISDAP, and selecting a time resolution that matches with the EISCAT data dump length. In UHF beata the resolution is 5 s. An example of a GUISDAP GUI with sufficient settings is given on the right in Figure 4.1.

After the GUISDAP analysis, we will have the 60 s fitted parameters and the 5 s raw densities in separate directories. In this case the directories are 2015-02-16\_beata\_60s@uhfa and 2015-02-16\_beata\_5\_rawNe@uhfa. Because the ElSpec defaults are for the UHF beata experiment, we can start the inversion by means of providing only the data directories,

```
>> out=ElSpec('ppdir','2015-02-16_beata_5_rawNe@uhfa','fitdir','2015-02-16_beata_60@uhfa');
```

ElSpec will then print the license information<sup>1</sup> and its settings. The listing is worth checking, especially when analyzing other EISCAT experiments than the

---

<sup>1</sup>Only on the first call in each MATLAB session



default beata, to be sure that all default parameters match with the experiment at hand. ElSpec reads all input data at once, which may take some time. When all data are read the result file name is printed, and information about each time step is printed while the inversion proceeds. An example of the output is given below.

```
>> out = ElSpec('ppdir','2015-02-16_beata_5_rawNe@uhfa','fitdir','2015-02-16_beata_60@uhfa')
```

Copyright 2015-2018, Ilkka Virtanen and Bjorn Gustavsson  
This is free software, licensed under GNU GPL **version 2** or later.

This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
See the GNU General Public License for more details

ElSpec **input** arguments

```
ppdir: 2015-02-16_beata_5_rawNe@uhfa
fitdir: 2015-02-16_beata_60@uhfa
experiment: beata
radar: uhf
version: 1
hmin [km]: 80.0
hmax [km]: 150.0
btime:
etime:
ionomodel: Fang
recombmodel: SheehanGr
integtype: integrate
E [keV]: 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03
0.03 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.06 0.06 0.06 0.06 0.06
0.06 0.07 0.07 0.07 0.07 0.07 0.07 0.08 0.08 0.08 0.08
0.09 0.09 0.09 0.09 0.10 0.10 0.10 0.11 0.11 0.11 0.11
0.12 0.12 0.13 0.13 0.13 0.13 0.14 0.14 0.15 0.15 0.16
0.16 0.16 0.17 0.18 0.18 0.19 0.19 0.20 0.20 0.21 0.21
0.22 0.22 0.23 0.24 0.25 0.25 0.26 0.27 0.28 0.29 0.29
0.30 0.31 0.32 0.32 0.34 0.35 0.36 0.37 0.38 0.39 0.39
0.40 0.42 0.43 0.44 0.46 0.47 0.48 0.50 0.52 0.53 0.53
0.55 0.57 0.58 0.60 0.62 0.64 0.66 0.68 0.70 0.72 0.72
0.75 0.77 0.79 0.82 0.84 0.87 0.90 0.93 0.95 0.98 0.98
1.02 1.05 1.08 1.11 1.15 1.18 1.22 1.26 1.30 1.34 1.34
1.38 1.43 1.47 1.52 1.56 1.61 1.66 1.71 1.77 1.82 1.82
1.88 1.94 2.00 2.06 2.13 2.19 2.26 2.33 2.41 2.48 2.48
2.56 2.64 2.72 2.81 2.89 2.98 3.08 3.17 3.27 3.38 3.38
3.48 3.59 3.70 3.82 3.94 4.06 4.19 4.32 4.45 4.59 4.59
4.74 4.89 5.04 5.20 5.36 5.53 5.70 5.88 6.06 6.25 6.25
6.45 6.65 6.86 7.07 7.29 7.52 7.76 8.00 8.25 8.51 8.51
8.77 9.05 9.33 9.62 9.92 10.23 10.55 10.88 11.22 11.58 11.58
11.94 12.31 12.70 13.09 13.50 13.93 14.36 14.81 15.27 15.75 15.75
16.24 16.75 17.28 17.82 18.37 18.95 19.54 20.15 20.78 21.43 21.43
22.10 22.80 23.51 24.24 25.00 25.79 26.59 27.42 28.28 29.17 29.17
30.08 31.02 31.99 32.99 34.02 35.09 36.18 37.32 38.48 39.69 39.69
40.93 42.21 43.53 44.89 46.30 47.75 49.24 50.78 52.37 54.01 54.01
55.70 57.44 59.23 61.09 63.00 64.97 67.00 69.10 71.26 73.49 73.49
75.79 78.16 80.60 83.13 85.73 88.41 91.17 94.03 96.97 100.00 100.00
maxorder: 5
plotres: 1
tres: dump
Emin [eV]: 1000
ninteg: 6
nstep: 1
Will write results in:
ElSpec_20150216T230135-20150216T235955_beata_uhf_Fang_SheehanGr_integrate_6_1_dump_20180221T123708.mat
16-Feb-2015 23:01:37 4 order, chisqr: 0.8, FAC: 0.08 ( 0.01)  $\mu\text{A}/\text{m}^2$ , Power: 0.19 ( 0.02)  $\text{mW}/\text{m}^2$ 
16-Feb-2015 23:01:42 1 order, chisqr: 1.5, FAC: 0.13 ( 0.02)  $\mu\text{A}/\text{m}^2$ , Power: 0.29 ( 0.04)  $\text{mW}/\text{m}^2$ 
16-Feb-2015 23:01:47 1 order, chisqr: 1.2, FAC: 0.12 ( 0.02)  $\mu\text{A}/\text{m}^2$ , Power: 0.28 ( 0.04)  $\text{mW}/\text{m}^2$ 
16-Feb-2015 23:01:52 1 order, chisqr: 1.5, FAC: 0.09 ( 0.02)  $\mu\text{A}/\text{m}^2$ , Power: 0.22 ( 0.04)  $\text{mW}/\text{m}^2$ 
16-Feb-2015 23:01:57 1 order, chisqr: 1.2, FAC: 0.08 ( 0.01)  $\mu\text{A}/\text{m}^2$ , Power: 0.21 ( 0.04)  $\text{mW}/\text{m}^2$ 
16-Feb-2015 23:02:02 1 order, chisqr: 1.7, FAC: 0.08 ( 0.02)  $\mu\text{A}/\text{m}^2$ , Power: 0.20 ( 0.04)  $\text{mW}/\text{m}^2$ 
```

With the default settings ElSpec will regularly update a plot of the inversions results, and will leave a plot of the final results open when the analysis is finished. The default axis limits used in this plot may not be the optimal ones. In this case

we adjust the axis limits with

```
>> ElSpecPlot(out,'plim',[0 10],'faclim',[0 5],'elim',[1 20]);
```

The plot produced with this command is shown in Figure 4.2. The top panel is the observed raw electron density, followed by the density modeled in the inversion, the inverted differential energy flux, the upward field-aligned current, and the net energy flux. The bottom panel is  $\chi^2$  of the fit.

Figure 4.2 shows a relatively good match between the observed and modeled densities, but also reveals one key issue in the inversion. Clearly visible mismatch between the model and the observation is seen during sudden decreases in electron density around 23:11, 23:15, 23:28, etc., where the ElSpec recombination model does not allow the electrons to recombine as fast as the density decreases in the observed data. These issues are possibly related to ion convection – the plasma may have moved out of the radar beam instead of recombining within the beam.

## 4.2 UHF beata with a steady state model

The steady state assumption, i.e. the assumption that ion production and recombination are in balance during an analysis time step, is widely used in electron energy spectrum inversion. This option can be selected with the `integtype` option of ElSpec. The same beata data can be analyzed with the steady state assumption using the command

```
out=ElSpec('ppdir','2015-02-16_beata_5_rawNe@uhfa','fitdir', ...  
           '2015-02-16_beata_60@uhfa','integtype','equilibrium');
```

Result of this inversion shown in Figure 4.3. A clear difference to the result with time dependent model in Figure 4.2 is that the issue with rapidly decreasing electron densities is not visible. The steady state model finds a precipitation, which would lead to the observed density profile if the energy spectrum remained unchanged for a long period of time. Individual time steps are thus completely independent and the analysis has no problem to adjust to arbitrarily large changes in the electron density. However, the steady state assumption is implicitly wrong and the smoother result is by no means better than the time dependent model – the simplistic steady state model simply hides the issue.

## 4.3 A fast inversion with fitted data

A simple way to create quicklook plots that give a rough idea of the energy spectrum characteristics is to run the inversion with standard GUISDAP fit results alone. Such an inversion is very fast to run and does not require additional GUISDAP runs on top of a routine parameter fit.

The same data that was used in the two previous examples can be inverted in this way using the command

```
out=ElSpec('fitdir','2015-02-16_beata_60@uhfa','integtype','equilibrium','ninteg',1);
```

Figure 4.4 shows this inversion results. All fine details are missing, because they are averaged out in the 60 s GUISDAP fit. However, the main characteristics of the spectra are roughly still there. Also the estimates of the field-aligned current and the net power flux are reasonably close to the values from the higher resolution inversions. A clear difference to the previous examples are the  $\chi^2$  estimates, which are much larger. This is possibly caused by inaccuracies in GUISDAP error estimation, but the actual cause has not been studied in detail. Also the default height resolution of the GUISDAP fit is slightly too coarse for ElSpec, and may have affected the results.

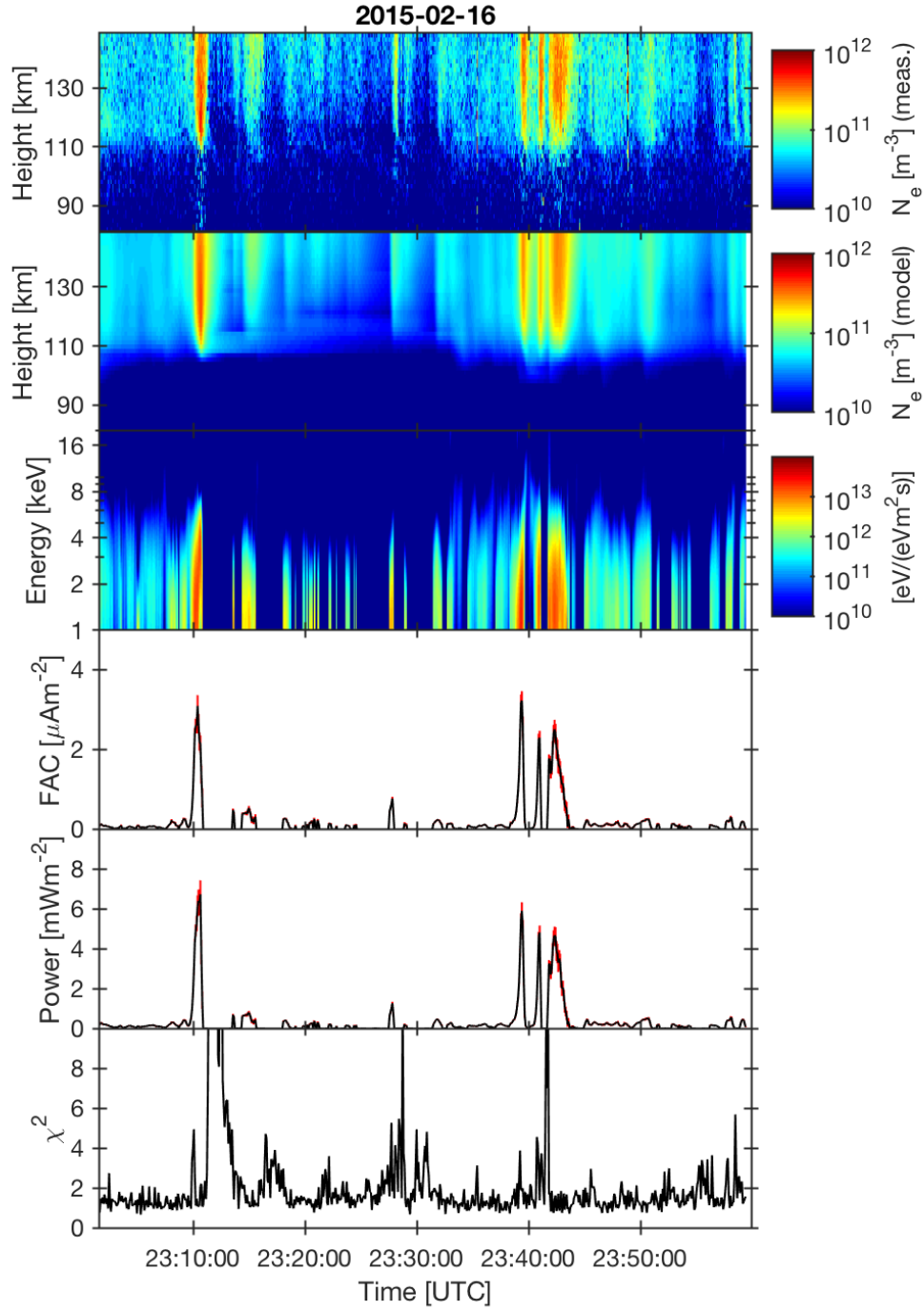


Figure 4.2: EISpec inversion results with the time dependent model and 5 s time resolution.

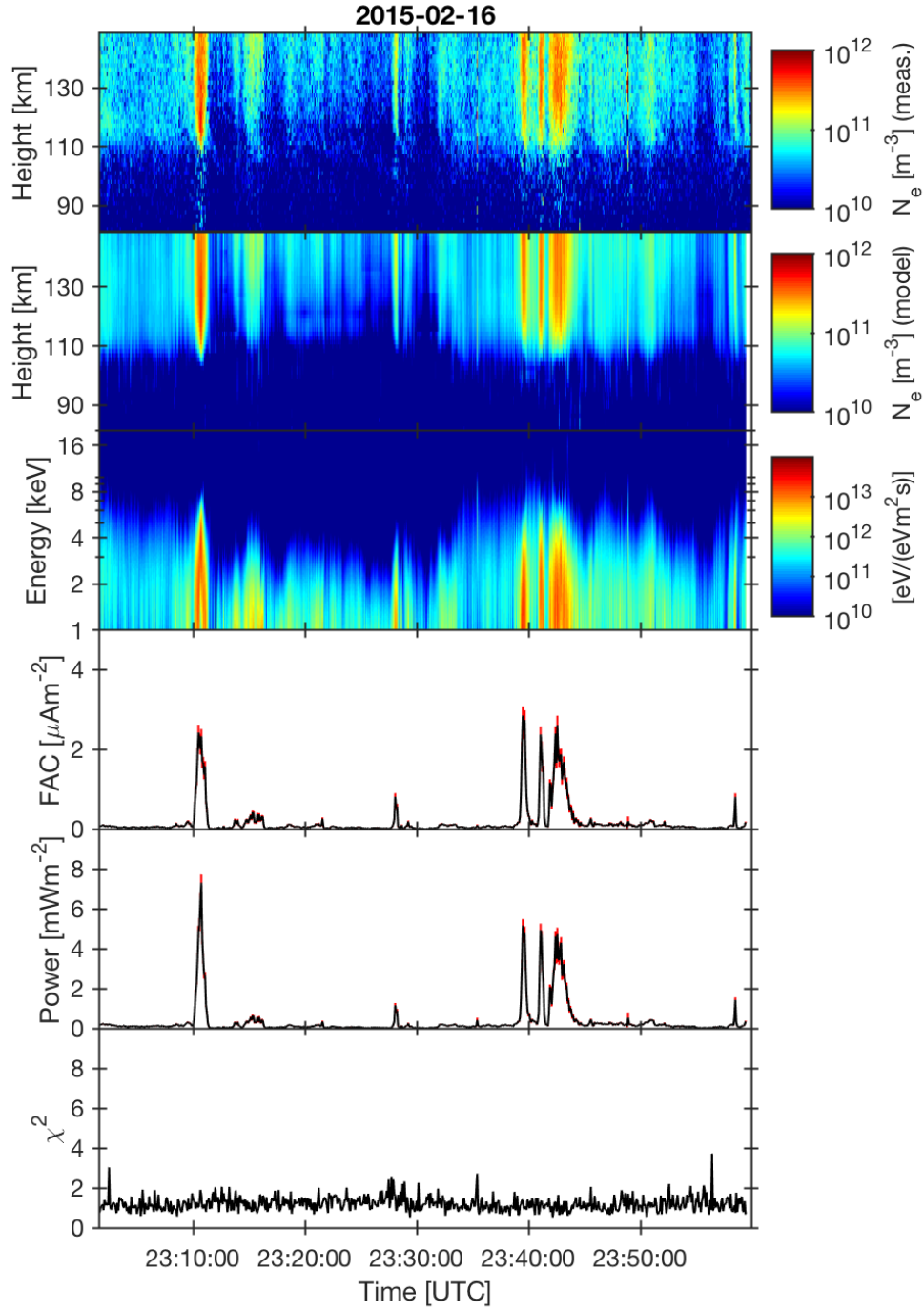


Figure 4.3: ElSpec inversion results with the steady state model and 5 s time resolution.

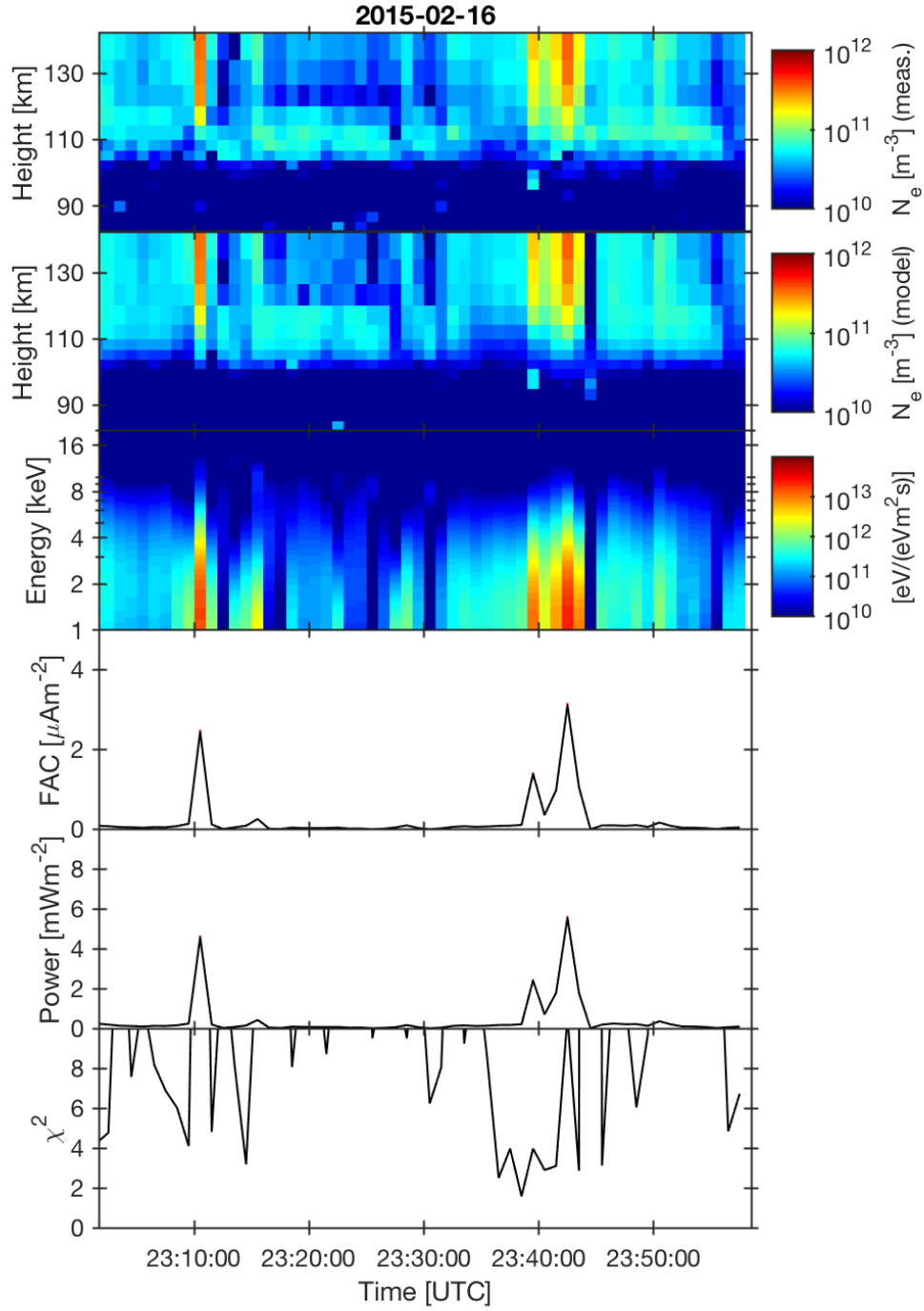


Figure 4.4: ElSpec inversion results with the steady state model and 60 s time resolution using only the standard GUISDAP fit results.

# Bibliography

- Burnham, K. P., and D. R. Anderson (2002), *Model Selection and Multimodel Inference*, 2nd ed., Springer.
- Dahlgren, H., B. Gustavsson, B. S. Lanchester, N. Ivchenko, U. Brändström, D. K. Whiter, T. Sergienko, I. Sandahl, and G. Marklund (2011), Energy and flux variations across thin auroral arcs, *Ann. Geophys.*, 29, 1699–1712, doi: 10.5194/angeo-29-1699-2011.
- del Pozo, C. F., J. K. Hargreaves, and A. D. Aylward (1997), Ion composition and effective ion recombination rate in the nighttime auroral lower ionosphere, *Journal of Atmospheric and Solar-Terrestrial Physics*, 59(15), 1919–1943, doi: 10.1016/S1364-6826(97)00033-3.
- Fang, X., C. E. Randall, D. Lummerzheim, W. Wang, G. Lu, S. C. Solomon, and R. A. Frahm (2010), Parameterization of monoenergetic electron impact ionization, *Geophys. Res. Lett.*, 37, L22,106, doi:10.1029/2010GL045406.
- Picone, J. M., A. E. Hedin, D. B. Prob, and A. C. Aikin (2002), NRLMSISE-00 empirical mode of the atmosphere: Statistical comparison and scientific issues, *J. Geophys. Res.*, 107(A12), 1468, doi:10.1029/2002JA009430.
- Rees, M. (1989), *Physics and chemistry of the upper atmosphere*, Cambridge University Press.
- Sergienko, T. I., and V. E. Ivanov (1993), A new approach to calculate the excitation of atmospheric gases by auroral electron impact, *Annales Geophysicae*, 11, 717–727.
- Sheehan, C. H., and J.-P. St-Maurice (2004), Dissociative recombination of  $\text{N}_2^+$ ,  $\text{O}_2^+$ , and  $\text{NO}^+$ : Rate coefficients for ground state and vibrationally excited ions, *J. Geophys. Res.*, 109, A03,302, doi:10.1029/2003JA010132.
- Virtanen, I. I., B. Gustavsson, A. T. Aikio, K. Asamura, and Y. Ogawa (2018), Incoherent scatter observations of electron precipitation energy flux in the auroral ionosphere, manuscript to be submitted to J. Geophys. Res. Space Physics.

# Appendix A

## Command reference

This appendix lists the optional arguments of the functions `ElSpec` and `ElSpecPlot`.

### A.1 `ElSpec`

All arguments to `ElSpec` are given using the standard MATLAB name-value pair notation, e.g. `ElSpec('inputdir', 'mydatadir')`, etc.

`ppdir` path to the raw electron densities.

Default: `[]`

Example: `ElSpec('ppdir', '1900-01-02_beata_5_rawNe@uhfa')`

NOTICE: either `ppdir` or `fitdir` must be defined.

`fitdir` path to the 'standard' GUISDAP fit results.

Default: `[]`

Example: `ElSpec('ppdir', '1900-01-02_beata_60@uhfa')`

NOTICE: either `ppdir` or `fitdir` must be defined.

`experiment` Name of the EISCAT experiment. The argument is used for selecting correct input routine for the raw densities, and is written in the output filename.

Default: `'beata'`

Example: `ElSpec('experiment', 'arc1')`

`radar` Name of the radar. The argument is used for selecting correct input routine for the raw densities, and is written in the output filename.

Default: `'uhf'`



Example: `ElSpec('radar','esr')`

**version** EISCAT experiment version. The argument is passed for the raw density input routines, but is currently not effective.  
Default: 1

**hmin** Lower altitude limit in kilometers. Must be from between 80 and 150 km.  
Default: 80  
Example: `ElSpec('hmin',95)`

**hmax** Upper altitude limit in kilometers. Must be from between 80 and 150 km.  
Default: 150  
Example: `ElSpec('hmax',130)`

**btime** Analysis start time as a vector [yyyy mm dd HH MM SS]. If not given, the analysis begins from the first available data point.  
Default: []  
Example: `ElSpec('btime',[1900 01 02 13 14 15])`

**etime** Analysis end time as a vector [yyyy mm dd HH MM SS]. If not given, the analysis runs until end of data.  
Default: []  
Example: `ElSpec('etime',[1900 01 02 18 19 20])`

**ionomodel** Ion production model, either 'Fang' or 'Sergienko'. The former is the model by *Fang et al.* (2010) and the latter is the model by *Sergienko and Ivanov* (1993). For both models, the neutral atmosphere is taken from the MATLAB aerospace toolbox implementation of NRLMSISE-00 model (*Picone et al.*, 2002).  
Default: 'Fang'  
Example: `ElSpec('ionomodel','Sergienko')`

**recombmodel** Ion recombination model, the models differ in ion composition modeling and in effective recombination coefficient for  $\text{NO}^+$  and  $\text{O}_2^+$ . The available options are:

Rees Effective recombination coefficients from *Rees* (1989), ion composition from IRI.

ReesNO+ Effective recombination coefficients from *Rees* (1989), pure NO<sup>+</sup>.

ReesO2+ Effective recombination coefficients from *Rees* (1989), pure O<sub>2</sub><sup>+</sup>.

SheehanGr Effective recombination coefficients for vibrational ground states from *Sheehan and St-Maurice* (2004), ion composition from IRI.

SheehanGrNO+ Effective recombination coefficients for vibrational ground states from *Sheehan and St-Maurice* (2004), pure NO<sup>+</sup>.

SheehanGrO2+ Effective recombination coefficients for vibrational ground states from *Sheehan and St-Maurice* (2004), pure O<sub>2</sub><sup>+</sup>.

SheehanEx Effective recombination coefficients for vibrationally excited ions from *Sheehan and St-Maurice* (2004), ion composition from IRI.

delPozo1 Effective recombination coefficient as function of altitude, without ion composition. Equation (13) from *del Pozo et al.* (1997). The functional form is  $\alpha_{eff} = 2.5 \cdot 10^{-12} \exp(-0.0242h) + 1.63 \cdot 10^5 \exp(-0.524h)$ .

delPozo2 Effective recombination coefficient as function of altitude, without ion composition. A simpler equation from (page 1934 *del Pozo et al.*, 1997).  $\alpha_{eff} = 2.5 \cdot 10^{-12} \exp(-0.0195h)$ .

Default: 'SheehanGr'

Example: `ElSpec('recombmodel', 'SheehanGrNO+')`

NOTICE: If `fitdir` is not defined, `recombmodel` has an effect on the electron temperature modeling.  $T_e$  is taken from IRI if `recombmodel` uses IRI, otherwise the temperature will be neutral temperature from MSIS.

integtype Technique for integrating the electron continuity equation. 'integrate', 'endNe' or 'equilibrium'. 'integrate' is the actual integration, 'endNe' calculates electron density at end of a time step instead of the average over a time step. 'equilibrium' assumes photochemical equilibrium, i.e. ignores the electron density time-derivative.

Default: 'integrate'

Example: `ElSpec('integtype', 'equilibrium')`

NOTICE: If 'integrate' is selected, the electron density at end of each time step is calculated with the 'endNe' option in a separate function call.

egrid Energy grid points [eV].

Default: `logspace(1,5,300)`

Example: `ElSpec('egrid',logspace(log10(800),log10(8e4),200))`

NOTICE: ElSpec cannot invert energies much above 100 keV (altitudes below about 80 km), because the ion chemistry becomes too complicated, and because the polynomial in the spectrum models may lead to problematic oscillations at higher energies.

**maxorder** Maximum order of the polynomial model.

Default: 5

Example: `ElSpec('maxorder',10)`

NOTICE: the minimum order is always 1, we do not fit flat (constant) spectra.

**plotres** Logical, should the inversion results be plotted with ElSpecPlot during and after the inversion process.

Default: true

Example: `ElSpec('plotres',0)`

**tres** Time resolution for reading the raw electron density data in `ppdir`, either 'dump' or 'best'. 'dump' integrates all raw density profiles from an EISCAT data dump in one profile, while 'best' keeps the profiles separate. The final time resolution depends on the experiment, 'dump' is 5 s for UHF beata and 4 s for UHF arc1. 'best' is approximately 0.35 s for UHF beata and 0.44 s for UHF arc1.

Default: 'dump'

Example: `ElSpec('tres','best')`

**emin** Minimum energy used for estimating the field-aligned current and net energy flux [eV].

Default: 1000

Example: `ElSpec('emin',500)`

NOTICE: The lowest energies are not reliable, because the low energy electrons cause ionization mainly at altitudes above the 150 km upper boundary (and would be unreliable even if upper altitudes were included, due to convection and long recombination times). The lowest detectable energy varies a bit, along variations in density profile of the neutral atmosphere. The default 1 keV limit should be rather safe.

**ieprior** *A priori* differential number fluxes in the same energy and time grid that is used in the inversion. Use an empty matrix to run without a prior. This is

a remnant from the inversion development work, which was left in place, because it might still have some use. The original idea was to first run the inversion with the 'equilibrium' integration, and use that result as a prior in the final analysis.

Default: []

Example: `ElSpec('ieprior', equilibriumresult.Ie)`

NOTICE: In the example above, `equilibriumresult` is an `ElSpec` output list from a similar inversion with 'integtype', 'equilibrium'.

`stdprior` *A priori* standard deviations in the same energy and time grid that is used in the inversion. Use an empty matrix to run without a prior. See also `ieprior` above.

Default: []

Example: `ElSpec('stdprior', equilibriumresult.IeStd)`

`ninteg` Number of time steps to integrate in each profile.

Default: 6

Example: `ElSpec('ninteg', 1)`

NOTICE: Only the first time slice is included with a full weight. Weights for the others depend on the electron recombination time scale.

NOTICE: If `tres` is changed, `ninteg` should be changed accordingly.

NOTICE: To run the inversion without the adaptive integration, set `ninteg` to 1.

`nstep` Number of electron density time slices to proceed in each inversion step. Changing this might be reasonable in some cases if `tres` is 'best', but should not be touched typically.

Default: 1

Example: `ElSpec('nstep', 2)`

`saveiecov` Logical, should the large number flux covariance matrices be saved? The matrices may be needed e.g. for recalculating the estimates of field-aligned current and net energy flux with different minimum energies `emin`. Correlations between nearby energy grid points are significant, and may be needed for some other purposes, too.

Default: false

Example: `ElSpec('saveiecov', 1)`

## A.2 ElSpecPlot

ElSpecPlot has one required argument, ElSpecOut, which is an output list from ElSpec. In addition, it has one optional argument, fignum, which may be given immediately after ElSpecOut in the function call. The other optional parameters are given in the standard MATLAB name - value pair format.

**ElSpecOut** An output list from ElSpec. This is a normal required argument.

Example: `out=ElSpec('fitdir','myFitDir'); ElSpecPlot(out)`

NOTICE: This is a normal MATLAB function argument, the call `ElSpecPlot('ElSpecOut',out)` will not work.

**fignum** Device number of the figure window, optional. If fignum is not given, a new figure window is created.

Example: `out=ElSpec('fitdir','myFitDir'); ElSpecPlot(out,2)`

NOTICE: Because fignum is not a MATLAB parameter but an optional argument, it must be given immediately after the ElSpecOut argument, and without its name.

**hlim** Height limits in the electron density plot [km].

Default: Smallest and largest height found from ElSpecOut.

Example: `ElSpecPlot(out,'hlim',[100 130])`

**nelim** Limits of the electron density color scale [ $\log_{10}(\text{m}^{-3})$ ].

Default: [10 12]

Example: `ElSpecPlot(out,'nelim',[9 11])`

**ielim** Limits for differential number flux color scale [ $\log_{10}((\text{eVm}^2\text{s})^{-1})$ ].

Default: [6 10]

Example: `ElSpecPlot(out,'ielim',[7 11])`

NOTICE: The differential number flux is not plotted by default, see 'fluxtype'.

**ieelim** Limits for differential energy flux color scale [ $\log_{10}(\text{eV}(\text{eVm}^2\text{s})^{-1})$ ].

Default: [10 14]

Example: `ElSpecPlot(out,'ieelim',[9 12])`

- elim** Energy axis limits in the flux plots [keV].  
 Default: [1 100]  
 Example: `ElSpecPlot(out,'elim',[1 40])`
- faclim** Current axis limits in the plot of field-aligned current [ $\mu\text{Am}^{-2}$ ].  
 Default: [0 10]  
 Example: `ElSpecPlot(out,'faclim',[0 30])`
- plim** Energy axis limits in the plot of net energy flux [ $\text{mWm}^{-2}$ ].  
 Default: [0 50]  
 Example: `ElSpecPlot(out,'plim',[0 200])`
- chisqrlim** Axis limits for the  $\chi^2$  plot.  
 Default: [0 10]  
 Example: `ElSpecPlot(out,'chisqrlim',[0 50])`
- btime** Start time as a vector [yyyy mm dd HH MM SS]. If not given, the plot begins from the first data point in `ElSpecOut`.  
 Default: []  
 Example: `ElSpecPlot(out,'btime',[1900 01 02 13 14 15])`
- etime** End time as a vector [yyyy mm dd HH MM SS]. If not given, the plot ends at the last data point in `ElSpecOut`.  
 Default: []  
 Example: `ElSpecPlot(out,'etime',[1900 01 02 18 19 20])`
- fluxtype** Type of the differential number flux plot 'number', 'energy' or 'both'. 'number' is the differential number flux, 'energy' is the differential energy flux, and 'both' plots both fluxes. If 'both' is selected, the net power flux plot is omitted.  
 Default: 'energy'  
 Example: `ElSpecPlot(out,'fluxtype','number')`
- neplot** Type of the electron density plot. 'log' uses a logarithmic density scale, 'linear' plots in linear scale.  
 Default: 'log'

Example: `ElSpecPlot(out, 'neplot', 'linear')`

NOTICE: `nelim` is given in logarithmic scale even if `'linear'` is selected.

`emin` Minimum energy included in field-aligned current and net power flux estimates [eV]. If the value is different from `ElSpecOut.emin`, the estimates of field-aligned current and net power flux are recalculated.

Default: 1000

Example: `ElSpecPlot(out, 'emin', 500)`

NOTICE: Error estimates of FAC and net power flux can be recalculated only if the full covariance matrix of the number flux is available (`saveiecov` is `true`) in `ElSpec`. If the covariance matrix is not available, the estimates are plotted without error bars.

# Appendix B

## GNU General Public License

GNU GENERAL PUBLIC LICENSE  
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,  
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This

General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.)

You can apply it to  
your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you



distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

#### GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License.

The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works.

But when you

distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or

collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code.

(This alternative is

allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable.

However, as a

special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt

otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed

through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number.

If the Program

specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN

OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  
THE ENTIRE RISK AS

TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  
SHOULD THE  
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,  
REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING  
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR  
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,  
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING  
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED  
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY  
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER  
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE  
POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest  
possible use to the public, the best way to achieve this is to make it  
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program.  
It is safest  
to attach them to the start of each source file to most effectively  
convey the exclusion of warranty; and each file should have at least  
the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify  
it under the terms of the GNU General Public License as published by  
the Free Software Foundation; either version 2 of the License, or  
(at your option) any later version.

This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the  
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along  
with this program; if not, write to the Free Software Foundation, Inc.,  
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.