

Data processing software for RHESSI and STIX instruments

Presented by: Liaisian Abdrakhmanova

Duration: 6 months

Date of the defence: September 12th 2019



Msc. Optics, Vision, Image, Multimedia
International Biometrics

Company's address/Department: LESIA, Paris Observatory, 5 place Jules Janssen, 92195 MEUDON

Supervisor at the company: Abdallah HAMINI, Project Manager, abdallah.hamini@obspm.fr

Contact UPEC (Head of the Master program): Prof. Amine NAIT-ALI, naitali@u-pec.fr

Contact UPEC administration: admin.biometrics@u-pec.fr

TABLE OF CONTENTS

Introduction.....	5
Chapter 1. Reconstruction of the distribution of electrons accelerated during the solar flares based on X-rays	7
1.1 Overview	7
1.2 RHESSI instrument	9
1.3 Spectral data analysis package OSPEX	10
1.4 Input data.....	11
Chapter 2. Observations and data analysis	12
2.1 Elementary plasma processes responsible for X-ray emission	12
2.2 Forward fitting approach	13
Chapter 3. OSPEX software development tools.....	15
3.1 Python programming language.....	15
3.2 Astropy library	15
3.3 What is Chianti and ChiantiPy	15
3.4 SunXpex package	16
3.5 Graphical User Interface	16
Chapter 4. OSPEX software realization	21
4.1 Reading the spectrum FITs files in Python.....	22
4.2 Software graphical user interface	28
4.3 Spectroscopic analysis.....	32
4.3.1 Spectrum evaluation with ChiantiPy and SunXpex packages	32

4.3.2 Fitting process	35
Conclusion	40
References	42

Abstract

The Reuven Ramaty High Energy Solar Spectroscopic Imager (RHESSI) is a National Aeronautics and Space Administration(NASA) Small Explorer (SMEX) mission to study the acceleration and transport of high-energy electrons and nuclei in solar flares. It provides X-ray data of thermal and non-thermal processes in the solar atmosphere.

The measurement by RHESSI of the X –ray producing electrons accelerated during eruptions gives information about the "timing" of the electron beam acceleration in the Sun, the location of this acceleration on the Sun's surface, and the intensity and spectrum of the accelerated electrons.

In general, the use of the RHESSI instrument requires the preparation of software for analyzing the data in order to generate spectra, reconstruct images from visibilities and etc.

OSPEX (Object Spectral Executive) is an object-oriented interface for X-ray spectral analysis of solar data written in IDL.

Our goal is to create the software similar to OSPEX which contains the same benefits or even more using Python programming language.

The work started as an internship project at LESIA, Paris Observatory in March 2019.

Introduction

Solar flares are magnetic explosive processes that occur in the solar atmosphere, resulting in effective particle acceleration and plasma heating. These phenomena cover all layers of the solar atmosphere: the photosphere, the chromosphere and the solar corona, and generate all types of electromagnetic radiation, from radio waves to X-rays and gamma rays. Flare plasma diagnostics is usually carried out by studying extreme ultraviolet radiation, while information about the non-thermal component of the plasma, the distribution of high-energy accelerated electrons, can be obtained from X-ray data. This range of wavelengths is sensitive to both thermal and non-thermal processes that occur in the area of primary energy release.

Typically, a person cannot view a solar flare by simply staring at the Sun. Flares are in fact difficult to see because the Sun is already so bright. Instead, specialized scientific instruments are used to detect the light emitted during a flare. Radio and optical emission from flares can be observed with telescopes on Earth. Energetic emissions such as X-rays and gamma-rays require telescopes located in space, since these emissions, thankfully, do not penetrate Earth's atmosphere.

Instruments on many solar spacecraft have recorded many flares in X-rays over the last fifty years. Scientists in laboratories work with these data which form the basis of our current understanding of a solar flare.

The Paris Observatory is the foremost astronomical observatory of France and one of the largest astronomical centers in the world. The Paris Observatory covers all areas of astronomy and astrophysics from the solar system, stars and their environment to galaxies, the origin of the universe, the measurement of time and space, and etc. Numerous observation programs are run, in particular in conjunction with NASA and the European Space Agency (ESA).

The Laboratory of Space Studies and Instrumentation in Astrophysics (LESIA) is one of the five Scientific Departments of the Paris Observatory. LESIA's scientific activities are classified in five main themes: Stellar Physics, High Angular Resolution and Astrophysics, Planetology, Plasma Physics and Solar Physics.

The significant improvement in the spatial, temporal and spectral resolution of modern ground-based and space-based solar telescopes allows to perform a detailed study of the solar flares [1] [2] [3] [4].

It is believed that solar flares are the result of reconnection of the magnetic field in the solar corona, where a large fraction of the magnetic energy passes into accelerated particles. Despite the fact that a qualitative picture of flares driven by magnetic energy release in coronal magnetic loops is generally accepted, detailed particle acceleration and propagation processes remain an unsolved problem of astrophysics and plasma physics.

The thesis uses data from modern spacecraft, namely:

- The Reuven Ramaty High Energy Solar Spectroscopic Imager (RHESSI), which detects X-rays with high spatial ($\sim 2.3''$) and spectral (~ 1 -10 keV) resolution, and thus allows to determine the source and form of distribution of electrons in the energy range from 3 keV to ~ 17 MeV using germanium detectors [5].

In addition to this, further plans are to include observed data from the X-ray Spectrometer (STIX) aboard Solar Orbiter (launch date in February 2020) [6].

This work is mainly devoted to the restoration of the energy distributions of electrons that generate X-rays during solar flares.

To achieve the goal, the following tasks were consistently set and solved:

- Consider as an example the current software for solar X-ray spectral data visualization named Object Spectral Executive (OSPEX). It has been developed in Interactive Data Language (IDL).
- Evaluate the results of the application.
- Develop a new software using Python programming language for the RHESSI data processing, which contains all benefits of OSPEX and even more.
- Integrate the X-ray spectrum processing procedures into a Python graphical user interface (GUI).

The thesis consists of introduction, 4 chapters, conclusion, list of references. The total volume of the thesis is 43 pages, including 29 figures and 3 tables. References include 28 items.

The introduction indicates the relevance of the study, the purpose of the work, theoretical and practical significance, a summary of the thesis.

Chapter 1 describes the reconstruction of the energy spectra of electrons accelerated in solar flares from X-ray data. This chapter provides a brief description of the RHESSI satellite. The chapter explains the classification of data obtained from the instrument and the actual OSPEX software that allows to perform spectral data analysis.

Chapter 2 explains the processes involved in the X-ray production in flares. It is dedicated to the restoration of the energy distributions of electrons based on X-rays. This chapter provides a theoretical explanation of the elementary plasma processes responsible for X-rays. In addition to this, it explains how spectral data analysis is performed in our software. This chapter also describes the fitting process that is used in the application.

Chapter 3 is devoted to the structure, the main tools and a GUI of the software. This chapter describes in details the libraries and methods used to create the OSPEX package.

Chapter 4 includes program implementation information. This chapter describes the process and the results of the work achieved in the thesis, illustrates the plots, proposes a new program interface.

In conclusion, we conclude the whole work results and determine the future work. We propose the link with our project provided on GitHub repository.

Chapter 1. Reconstruction of the distribution of electrons accelerated during the solar flares based on X-rays

1.1 Overview

Currently, the “standard” (two-dimensional) model of a solar flare [7] [8] [9] is actively involved in interpreting flare events, although there are a number of other models [2]. In particular, it is considered [10] that the release of primary energy occurs as a result of magnetic reconnection, which subsequently leads to the release of the plasmoid and the acceleration of charged particles in the region of the top of the flare loop (see Figure 1, built on basis of the model from [11]). Part of the accelerated particles, going up, leaves the solar corona through open magnetic lines of force and can propagate in the interplanetary medium towards the Earth. Another part of energetic electrons, propagating along magnetic field lines, spills out at the base of the loop, causing the generation of hard X-ray radiation and the heating of the chromosphere [12]. “Evaporating” hot plasma with a temperature of $(5-30) \times 10^6$ K fills the coronal part of the magnetic loop and is displayed in the ultraviolet and soft X-rays.

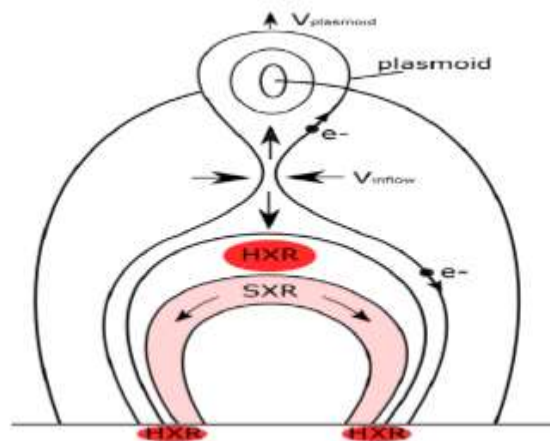


Figure 1. “Standard” (two-dimensional) model of a solar flare. The figure shows schematically the geometry of the flare region and the characteristic processes generated during the flare process[11]

The duration of solar flares generally does not exceed a few minutes, but in some cases it can reach several hours. In different ranges of the energy spectrum this value may vary. As a rule, a solar flare is divided into three phases: the growth phase, the maximum (impulsive phase), and the intensity decrease.

The accelerated electrons are non-thermal, possessing kinetic energies several times the thermal energy of the surrounding plasma. They travel along the field lines and interact with the ambient solar atmosphere, releasing part of their energy as bremsstrahlung X-ray photons. The remaining energy contributes to heating of the atmosphere, driving the plasma temperature to tens of millions of degrees – well above the pre-flare values.

Thermal electrons interacting in the hot plasma also emit bremsstrahlung X-rays. We can directly observe the X-ray emission from both populations of electrons.

The non-thermal electron bremsstrahlung observed in solar flares has a characteristic power-law spectrum. The hard X-ray flux observed in many flares implies that the total energy deposited by non-thermal electrons is of the same order as the total energy output of the flare.

This suggests a direct connection between flare-energy release and particle acceleration. Many flare models also predict that non-thermal X-ray emission should be linearly polarized due to beaming of the electrons.

Thermal electron bremsstrahlung has a characteristic quasi-exponential spectrum, in contrast to the non-thermal spectrum. From the shape and amplitude of the thermal continuum, along with an estimate of the source volume, we can calculate the plasma temperature and density, and thereby obtain the total thermal energy of the plasma. This, in turn, allows us to learn how energy is deposited into the solar atmosphere by non-thermal electrons.

The high flare temperatures also ionize coronal atoms and excite atomic transitions that are characterized by spectral line emission. In particular, transitions of highly ionized iron (Fe) and nickel (Ni) generate numerous lines that comprise two “complexes” centered at ~ 6.7 keV and ~ 8 keV. Observations of the Fe and Fe/Ni line complexes can thus provide confirmation of the continuum temperature measurement, and can also allow us to probe the Fe abundance in the solar corona [13].

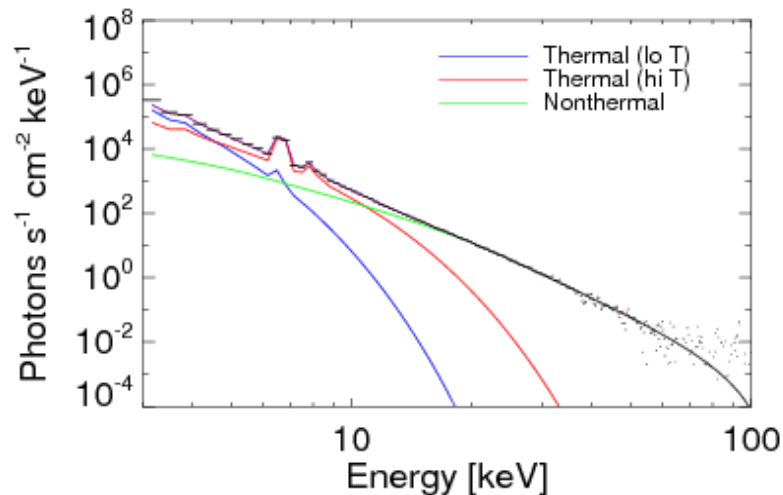


Figure 15. Photon spectrum for February 13th 2011, 17:32:36(UT), contains two thermal components (red, blue) and a non-thermal component (green)
(Source: <http://inspirehep.net/record/1675833/plots>)

In the last decade, numerous results of hard X-ray emission during flares on the Sun with high spatial, temporal and energy resolution were obtained, allowing a detailed study of the structure of hard X-ray flares [1] [14] [2] [15].

1.2 RHESSI instrument

The RHESSI mission consists of a single spin-stabilized spacecraft in a low-altitude orbit inclined 38 degrees to the Earth's equator. The only instrument on board is an imaging spectrometer with the ability to provide X-ray images and spectra. It uses two new complementary technologies: fine grids to modulate the solar radiation, and germanium detectors to measure the energy of each photon very precisely.

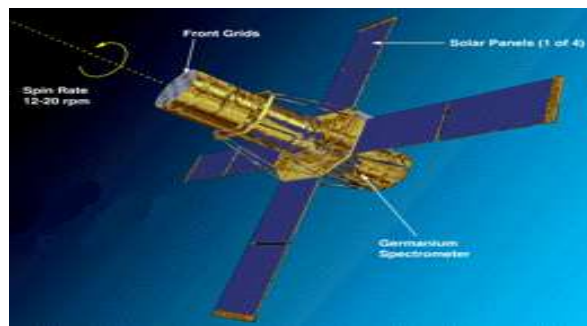


Figure 2. RHESSI spacecraft[16]

Context observations from ground-based observatories and a theory program are also integral parts of the RHESSI mission. Ground-based optical and radio telescopes provide complementary data on the magnetic fields, electric currents, hot plasma, and the energetic electrons in the flaring regions where the X-ray and gamma-ray emissions are generated [16].

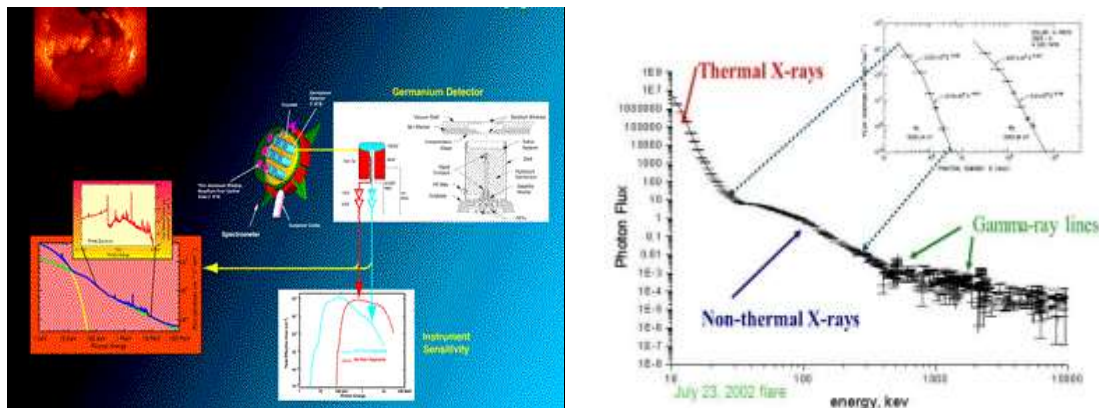


Figure 3 . HESSI Spectroscopy[16]

Spectral analysis is an inverse problem that begins with a spectrum of counts per spectrometer channel and seeks to recover the spectrum of photons per energy interval initially incident on the spacecraft. The RHESSI data analysis software is written to be part of the SolarSoft system [17]. It allows users to display, analyze, and store HESSI image, spectrum, lightcurve, and observing summary data, as well as view data from ancillary sources.

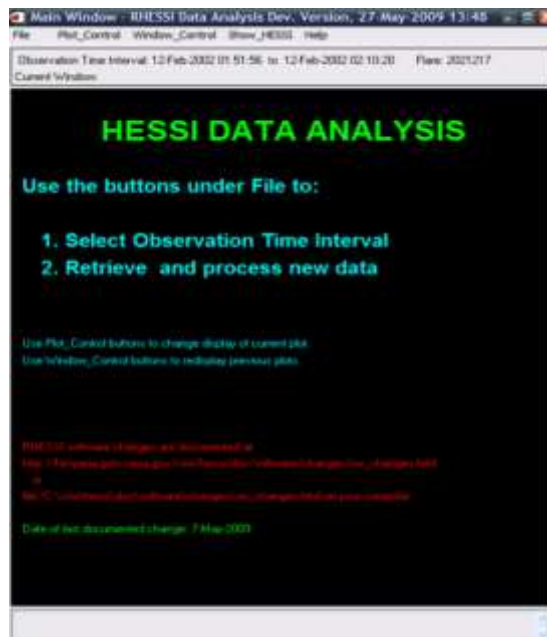


Figure 4. The RHESSI data analysis software[17]

All RHESSI science data are available to the science community within hours of transmission from the satellite to the ground.

The Observing Summary files are smaller daily FITS files (~2 MB / day) that contain various rates pre-binned to coarse energy and time resolution.

Spectral analysis in particular is accomplished with the SPEX spectral – inversion code(see below) and uses larger data files than the Observing Summary files. These data are generated by HESSI software.

1.3 Spectral data analysis package OSPEX

OSPEX is an object-oriented interface for X-ray spectral analysis of solar data [18]. Through OSPEX, the user reads and displays the input data, selects and subtracts background, selects time intervals of interest, selects a combination of photon flux model components to describe the data, and fits those components to the spectrum in each time interval selected. During the fitting process, the response matrix is used to convert the photon model to the model counts to

compare with the input count data. The resulting time-ordered fit parameters are stored and can be displayed and analyzed with OSPEX.



Figure 5. Main window of OSPEX software[18]

1.4 Input data

RHESSI covered more than a full 11-year solar cycle, operating from 2002 to 2018 and recording more than 120,000 X-ray events, 42 with gamma-radiation above 300 keV and 27 with gamma-ray emission. During this time, RHESSI was the only observatory that could observe the spectroscopy of energetic electrons that carry such a large part of the energy released in flares.

OSPEX currently handles lot of types of file extensions. In our work we analyze RHESSI spectrum which is stored in FITS format. FITS is one of the most common ways to store spectra. In most cases the flux values are stored as the pixel values in a 1-, 2- or 3-D arrays. In Chapter 3 we will describe how to read and extract the information from FITS files.

Chapter 2. Observations and data analysis

2.1 Elementary plasma processes responsible for X-ray emission

Diagnostics of X-ray emission is one of the most direct methods by which energetic electrons are studied in solar flares. The main mechanisms responsible for plasma radiation in the X-ray region of the spectrum are free-free (bremsstrahlung) and free-bound transitions.

Free-free transitions are understood as the deceleration of energetic electrons on particles of the solar plasma, accompanied by the emission of quanta, where the dominant interactions are mainly electron-ion.

The Figure 6 shows a comparison of the contribution of the two radiation mechanisms (bremsstrahlung, free-bound) to the continuum for isothermal spectra based on numerical calculations of the atomic database CHIANTI for a temperature of 10 MK [19] [20]. The CHIANTI database will be described in more details in Chapter 3.

Free-bound transitions are the process of capturing a free electron by an ion or atom, followed by emission of a quantum, but the second option is less likely [21].

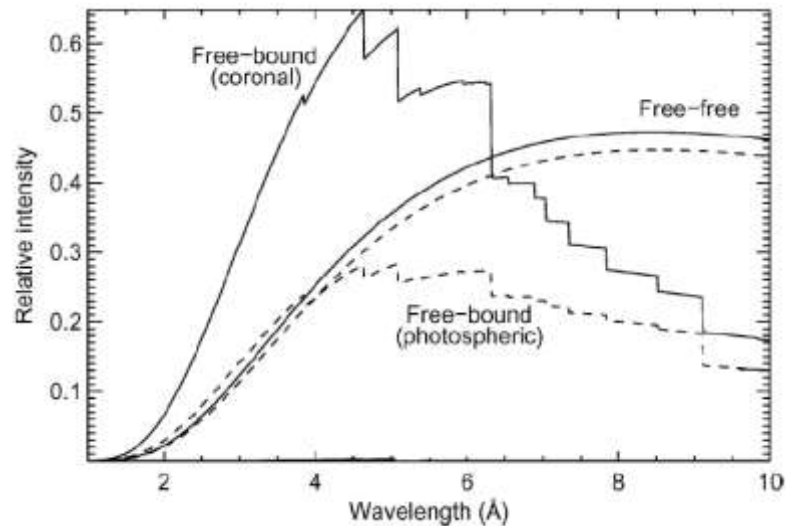


Figure 6. Comparison of the contribution of the two mechanisms to the continuum for isothermal spectra calculated using CHIANTI database at 10 MK. Spectra for free-bound, free-free transitions for coronal (solid lines) and photospheric (dashed lines) abundances of elements. The curves are normalized to the peak intensity of the total radiation for coronal abundances. The sharp edges in the spectra of free-bonded transitions represent transitions of ions to different states[19]

2.2 Forward fitting approach

Although we use a spectrometer to measure the spectrum of a source, what the spectrometer measures is not the real spectrum, but rather counts (C) within specific instrument channels(I). This observed count spectrum is related to the original spectrum of the source ($f(E)$) by:

$$C(I) = \int f(E)R(I,E)dE$$

Where $R(I, E)$ is the instrumental response and is proportional to the probability that an incoming photon of energy E will produce a count in channel I . Ideally, then, we would like to determine the actual spectrum of a source, $f(E)$, by inverting this equation, thus deriving $f(E)$ for a given set of $C(I)$. Regrettably, this is not possible in general, as such inversions tend to be non-unique and unstable to small changes in $C(I)$.

The usual alternative is to choose a model spectrum, $f(E)$, that can be described in terms of a few parameters (i.e., $f(E, p1, p2, \dots)$), and match, or “fit” it to the data obtained by the spectrometer. For each $f(E)$, a predicted count spectrum ($C_p(I)$) is calculated and compared to the observed data ($C(I)$). Then a “fit statistic” is computed from the comparison and used to judge whether the model spectrum “fits” the data obtained by the spectrometer.

The model parameters then are varied to find the parameter values that give the most desirable fit statistic. These values are referred to as the best-fit parameters. The model spectrum, $f_b(E)$, made up of the best-fit parameters is considered to be the best-fit model.

The most common fit statistic in use for determining the “best-fit” model is χ^2 , defined as follows:

$$\chi^2 = \sum \frac{(C(I) - C_p(I))^2}{(\sigma(I))^2}$$

where $\sigma(I)$ is the error for channel I (e.g., if $C(I)$ are counts then $\sigma(I)$ is usually estimated by $\sqrt{C(I)}$ [22]. It originates from the discrete nature of electrons and associated with the particle nature of light. This is called Poisson noise [23].

Once a “best-fit” model is obtained, one must ask two questions:

- How confident can one be that the observed $C(I)$ can have been produced by the best-fit model $f_b(E)$? The answer to this question is known as the “goodness-of-fit” of the model. The χ^2 statistic provides a well-known-goodness-of-fit criterion for a given number of degrees of freedom (ν , which is calculated as the number of channels minus the number of model parameters) and for a given confidence level. If χ^2 exceeds a critical value (tabulated in many statistics texts) one can conclude that $f_b(E)$ is not an adequate model for $C(I)$. As a general rule, one wants the “reduced χ^2 ” ($\sim \chi^2/\nu$) to be approximately equal to one (i.e. $\chi^2 \sim \nu$). A reduced χ^2 that is much greater than one indicates a poor fit, while a reduced χ^2 that is much less than one indicates that the errors on the data have been over-estimated. Even if the best-fit model ($f_b(E)$) does pass the “goodness-of-fit” test, one still

cannot say that $f_b(E)$ is the only acceptable model. For example, if the data used in the fit are not particularly good, one may be able to find many different models for which adequate fits can be found. In such a case, the choice of the correct model to fit is a matter of scientific judgment.

- For a given best-fit parameter (p_1), what is the range of values within which one can be confident the true value of the parameter lies? The answer to this question is the “confidence interval” for the parameter. The confidence interval for a given parameter is computed by varying the parameter value until the χ^2 increases by a particular amount above the minimum, or “best-fit” value. The amount that the χ^2 is allowed to increase (also referred to as the critical $\Delta\chi^2$) depends on the confidence level one requires, and on the number of parameters whose confidence space is being calculated. The critical for common cases are given in the following table:

Confidence	Parameters		
	1	2	3
0.68	1.00	2.30	3.50
0.90	2.71	4.61	6.25
0.99	6.63	9.21	11.30

Table 1. Confidence of fitting process based on the model parameters

Considering the forward fitting method in our work, it is assumed that the form of the energy distribution of electrons is known, and the parameters that determine this form are searched. On the other hand, the parameters found have a physical meaning (spectral index, total number of accelerated electrons, total energy), which makes it possible to diagnose a radiation source, to consider the dynamics of flare processes and to describe certain processes in a flare plasma. The approximation of observational data by a model function is, as described above, in the selection of function parameters for which the minimum value of χ^2 is found.

In our computing, we apply the Levenberg–Marquardt algorithm is used to solve non-linear least squares problems [24].

Chapter 3. OSPEX software development tools

3.1 Python programming language

One of the most popular programming languages today is Python, the use of which makes it possible to perform various tasks. The language includes many packages and modules that ensure its versatility.

Considering a number of advantages and the fact that Python is the best package for data processing and working with graphs, it was decided to develop our application in this language.

3.2 Astropy library

All modern astronomical research makes use of software in some way. Astronomy as a field has thus long supported the development of software tools for astronomical tasks, such as scripts that enable individual scientific research, software packages for small collaborations, and data reduction pipelines for survey operations. Some software packages are supported by large institutions and are intended for a wide range of users. These packages therefore typically provide some level of documentation and user support or training. Other packages are developed by individual researchers or research groups and are then typically used by smaller groups for more domain-specific purposes. For both packages meant for wider distribution and for scripts specific to particular research projects, a library that addresses common astronomical tasks simplifies the software development process. The users of such a library then also benefit from a community and ecosystem built around a shared foundation. The Astropy project has grown to become this community for Python astronomy software, and the astropy core package is a feature-rich Python library [25].

The Astropy project aims to provide an open-source and open-development core package (astropy) and an ecosystem of affiliated packages that support astronomical functionality in the Python programming language. The astropy core package is now a feature-rich library of sufficiently general tools and classes that supports the development of more specialized code. An example of such functionality is reading and writing FITS files. Another example of such a common task is in dealing with representations of and transformations between astronomical coordinate systems.

Since the Astropy project aims to develop and provide high-quality code and documentation in accordance with the best practices in software development, we chose it as one of the tools for creating OSPEX software.

3.3 What is Chianti and ChiantiPy

The CHIANTI package consists of a critically evaluated set of atomic data (energy levels, wavelengths, radiative transition probabilities and excitation data) for a large number of ions of astrophysical interest. It also includes a number of ancillary data and a suite of IDL programs to calculate optically thin synthetic spectra to perform spectral analysis and plasma diagnostics.

Plasma emission codes have long been used to study UV and X-ray spectral lines emitted from solar or stellar atmospheres. A comparison of the theoretical line intensities with the observed intensities allows a determination of the physical parameters such as abundances for the plasma.

The CHIANTI database has been used extensively by the astrophysical and solar communities to analyse emission line spectra from astrophysical sources [20].

While IDL has been the lingua franca of solar physics for over twenty years, Python is gaining momentum in the community and is the language of choice for many younger researchers. This is largely due to the fact that IDL is no longer free programming language. Users must purchase a license to use IDL.

In addition to this, Python has a success in general astronomy. For example, availability of Astropy library, the advent of SunPy which is a stable and well-supported Python package for solar data analysis.

Given the growing popularity of Python in the solar community and the importance of CHIANTI to solar observers and modelers alike, a well-supported Python interface to this database is critical. The ChiantiPy project, started by Ken Dere, provides a Python package for interacting with the CHIANTI database and an alternative to the IDL tools [26]. ChiantiPy is not a direct translation of its IDL counterpart, but instead provides an intuitive object oriented interface to the database (compared to the more functional approach in IDL). Given these facts, it was decided to consider the ChiantiPy package as a spectrum estimation tool in our software.

3.4 SunXpex package

SunXpex is a package for solar X-ray spectroscopy. This project was proposed by Dan Ryan in 2019(NASA) [27]. The package is based upon the Astropy package template. SunXpex provides the module for the classification of elements by their abundances responsible for X-ray emission. In addition to this, there is a module that allows to plot Chianti lines. Continuum plotting part is also under development. Thus, this package is an integral fitting part for thermal emission and tested in our work.

3.5 Graphical User Interface

Several frameworks exist that offer the ability to create slick GUI with Python.

GUI means all those windows, buttons, text fields for input and etc., that can be seen on the screen, opening any application. Through them we interact with the program and manage it. All these interface elements together is called widgets.

Currently, almost all applications that are created for the end user have a GUI. The OSPEX application is also designed to interact with the user.

There are many GUI libraries. Our application is oriented to work with data and visualize them graphically. Referring to these conditions, it was outlined the main requirements for libraries that are necessary when creating our application:

1. Cross-platform.

2. System integration.

4. Convenience of plotting and transfer of the entered parameters to the plot.

5. Create multiple windows / frames.

Initially, it is conducted a general monitoring to determine the most popular libraries for the current year. As of March 2019, the most popular toolkits for creating GUIs are as follows:

Toolkits	Rating
KIVY	1
PYQT	2
TKINTER	3
PYSIDE	4
PYSIMPLEGUI	5

Table 2. Most popular GUI libraries in 2019

This information was collected referring to user ratings. But as it turned out in the further analysis, the last two of them are not cross-platform and it was decided to exclude them from the list.

Currently, Python has 5 cross-platform tools that allow us to write "full-fledged" applications under Windows, Linux and Mac operation systems. They are the following :

№	Toolkits
1	TKINTER
2	PYQT
3	PYGTK
4	WXPYTHON
5	KIVY

Table 3. 5 cross-platform tools for Python

We can observe that three of these libraries are included in popularity and compliance for our case in both lists. Further, it will be logical to consider only them and conduct a detailed analysis.

After analyzing the main advantages and disadvantages of each of them, it can be concluded that Tkinter is most commonly used method. It is a standard interface shipped with Python. Python with Tkinter outputs the fastest and easiest way to create the GUI applications.

Advantages:

- ✓ Free for commercial use.
- ✓ Included in the standard Python library.
- ✓ Building executables is less complicated.

Compared to other GUI libraries, building executables for TkInter applications are simpler because TkInter is included in Python and has no other dependencies. This results in less complicated packaging requirements and smaller binary size.

- ✓ Easy to learn and get productive.

TkInter is a small library with a gentle learning curve compared to what's out there. It has a straightforward API and is often the go-to choice for building quick GUIs for Python scripts.

- ✓ Many learning resources available.

Tkinter is one of the most popular framework for Python. It has an active community with many third-party code examples and tutorials available.

Drawbacks:



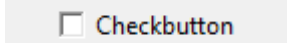
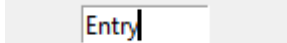
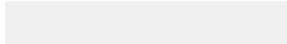
- No advanced widgets.



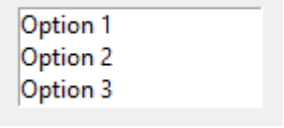
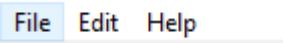

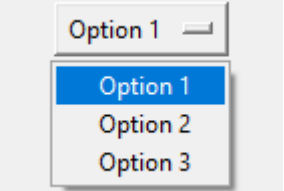

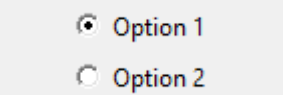
TkInter does not come with advanced widgets out of the box (e.g. date picker)

- No reliable UI builder available.

There is no tool in the same league as Qt Designer (PyQt) for TkInter.

Below we covered the basic widgets in order to determine whether this library has all the components that are contained in the OSPEX software toolbar [28].

Widget	Description	Example
Button	Clickable area with text that calls an associated function whenever the user clicks in the area.	
Canvas	General purpose widget that can display simple graphics or provide an area to implement a custom widget.	
Checkbutton	Allows a user to read and select between two distinct values (e.g. on/off). Also known as a "checkbox."	
Entry	Area that allows the user to enter one line of text. For entering multiple lines, use the <i>Text</i> widget.	
Frame	A container for other widgets. A Frame can be useful for grouping other widgets together in a complex layout.	

<u>Label</u>	Used to display text or an image that may not be edited by the user.	
<u>LabelFrame</u>	A rectangular area that contains other widgets, but unlike a <i>Frame</i> , a border and label may be added to help group widgets together.	
<u>Listbox</u>	Displays a list of text alternatives. The user can choose (highlight) one or more options.	
<u>Menu</u>	Used to create menus and submenus within an interface. Can be used to create the always-shown "menu bar" popular in many GUI applications.	
<u>Menubutton</u>	Obsolete as of Tk 8.0. Use <i>Menu</i> widget instead.	See <i>Menu</i>
<u>Message</u>	Used to display static text, like <i>Label</i> , but allows for multiple lines, text wrapping, and maintaining aspect ratios.	
<u>OptionMenu</u>	Drop-down (or pop-up) menu that allows users to select one option from several options.	
<u>PanedWindow</u>	Container for one or more widgets split into multiple "panes." These panes can be resized by the user by dragging the separator line(s) (known as "sashes").	
<u>Radiobutton</u>	Several Radiobuttons can be used together to allow the	




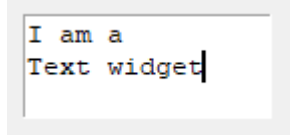
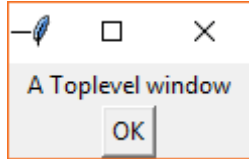
	user to select one option out of a group of options.	
Scale	User can select a numerical value by moving a slider.	
Scrollbar	Paired with a <i>Canvas</i> , <i>Entry</i> , <i>Listbox</i> , or <i>Text</i> widget to allow for scrolling within that widget.	
Spinbox	Allows the user to select only one option out of a list of options.	
Text	Area used to display and edit multiple lines of text. Can be used as a full text editor by the user.	
Toplevel	A container for other widgets (much like the <i>Frame</i> widget) that appears in its own window. It can be useful for creating other application windows or pop-up notifications.	

Table 4. Main Tkinter widgets[28]

The information from the table allows us to conclude that Tkinter is a suitable library, with the exception of some shortcomings.

PyQt is more modern and has better widgets and all around infrastructure to go with it. But as a big disadvantage of this library we can consider the fact that sometimes there is not enough documentation to learn and apply all the features of this toolkit. In particular, this concerns the plotting. In addition to this fact, from my own experience it can be concluded that in PyQt5 not all buttons are active by default. In order to activate them, we need to generate a separate code. Regarding our requirements for easy system integration, the PyQt library does not always cope with it.

Kivy is ideal for development of mobile applications and games.

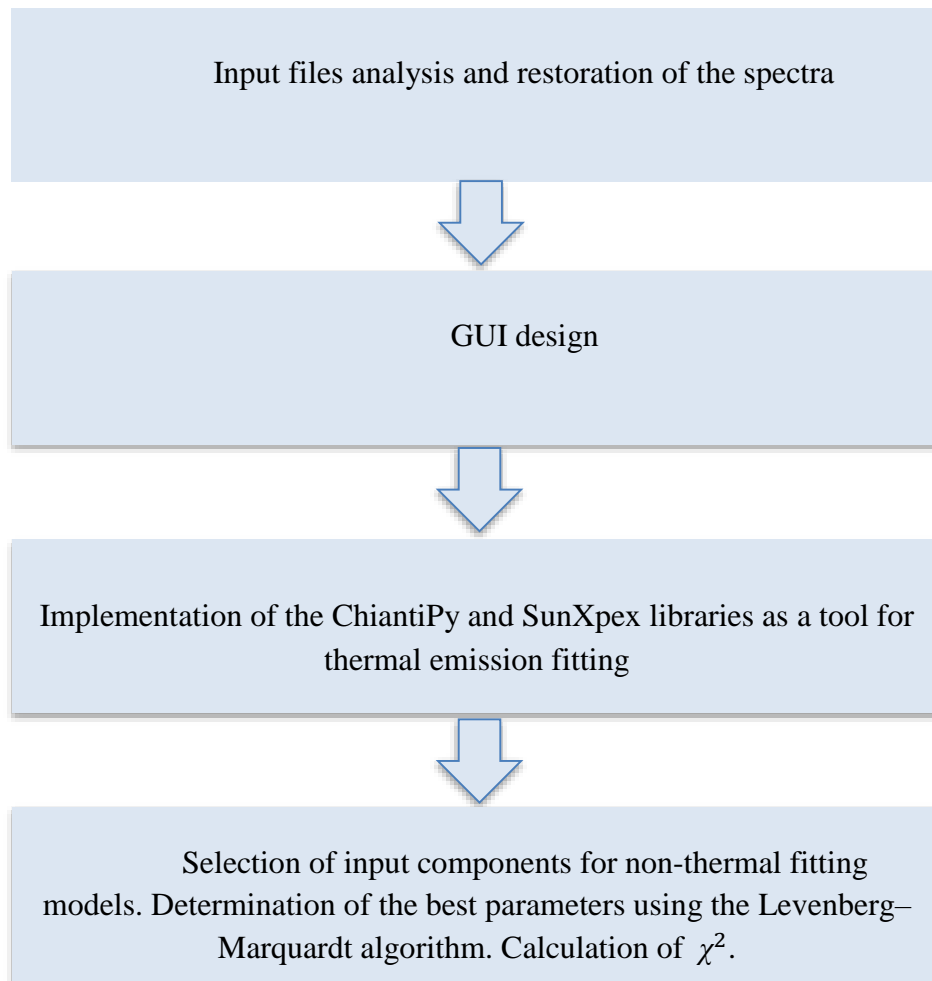
PyQt and Kivy can be used for big applications and Tkinter for small ones. Although the PyQt library is one of the most powerful GUI libraries, It was decided to create the GUI using Tkinter. This decision is based on the fact that OSPEX software contains standard interface with basic widgets.

Chapter 4. OSPEX software realization

As mentioned above, our goal is to create a Python application that includes the capabilities of the OSPEX package written in IDL. The following list shows the main features of this software:

1. Select input data files;
2. Define background and analysis intervals and select fit function components;
3. Fit data;
4. View fit results;
5. Save session and results.

In order to develop all the features of the original OSPEX software, a sufficient amount of time is required. During the internship, the following steps were realized:



These steps allowed us to create a “Select Input” part, perform a new GUI using Python library tool and implement forward fitting approach for One Dimensional Powel Law and One Dimensional Broken Power Law functions.

Next sections will describe in detail the software implementation and illustrate the results.

4.1 Reading the spectrum FITs files in Python

Let’s have a look at some of the basics required to analyze X-ray data. We will start off with opening up and investigating the basic data files.

As photons come in over the time and with different energies, a spectrum of counts is built up. The detection process, data acquisition, and data reduction processes differ across instruments, but most of them and also our case with RHESSI, as was mentioned in Chapter 1, produce the data files for observations in the FITS file format. It is a binary file with text headers that contains regulated keywords so that people and analysis tools can have a common format and understand exactly what is in the data.

We will now extract the counts spectrum from a FITS file that contain various rates pre-binned to energy and time resolution.

The Astropy collaboration has taken over the Python code to read FITS files and it is the tool we will utilize here.

We are going to work with some sample data. This file contains the event recorded on February 20th 2002 with a duration of 2400 seconds(s) started at 08:00:00 and ended at 08:40:00. This interval was chosen purposefully, since it has more significant information about the energy spectra.

Opening a FITS file with Astropy package’s module is relatively straightforward. When open the file, the returned object, called *hdulist*, behaves like a Python list and each element maps to a Header-Data Unit (HDU) in the FITS file. We can view more information about the FITS file:

Filename:

C:/Users/a_lesya007/PycharmProjects/PythonVersionOSPEX/hsi_spectrum_20020220_080000.fits

No. Name Ver Type Cards Dimensions Format

0 PRIMARY 1 PrimaryHDU 26 ()

1 RATE 1 BinTableHDU 106 600R x 7C ['77E', '77E', '77J', '77E', 'I', 'D', 'E']

2 ENEBAND 1 BinTableHDU 49 77R x 3C ['J', 'E', 'E']

3 HESSI Spectral Object Parameters 1 BinTableHDU 415 1R x 184C [I, I, I, I, I, B, B, 6A, B, 9B, D, D, 2E, 2E, I, B, I, E, I, 27B, 18B, I, I, I, I, I, I, I, E, I, I, 6E, I, E, I, I, 20A, 27B, 2E, D, I, 2D, I, I, B, B, B, I, E, I, E, E, E, E, 18D, 18E, 18E, 18E, 18E, 18I, 18I, E, I, 18I, 18J, E, I, E, E, E, E, I, 4A, B, 10B, I, 2D, 2D, J, 74A, I, 18B, B, B, B, 18B, 10B, B, E, E, I, 2E, E, I, D, 10B, 10B, E, B, B, B, 16A, 25A, 9A, 9A, I, I, 9E, E, I, E, I, 18B, 77E, 600E, I, 6A, 4J, I, D, E, E, I, 10800E, 2E, J, D, 9I, 36E, 2D, B, 2D, 2J, D, B, 7E, 91E, 78E, 81E, 8A, 2D, I, 54B, 18B, J, J, J, J, J, J, E, E, E, E, D, D, B, B, I, I, I, 9E, 9E, D, D, B, B, I, I, I, 9E, 9E, 54E, J, D, B, B, D, B, 78E, 2856A, 1040A]

Figure 7. FITs file information

We can see that there are four extensions(HDUs) in the file. We will primarily be interested in the RATE and ENEBAND extensions which contain the spectral data.

First, we will extract the spectrum extension. There are various ways to access data in a fits file that depend of speed, ease of use, etc. We will just access the extensions by keyword.

Extracted object (HDU) then has two important attributes: data, which behaves like an array, can be used to access the data, and header, which behaves like a dictionary, can be used to access the header information. First, we can take a look at the header.

```
XTENSION= 'BINTABLE'      / Written by IDL: Wed Jul 17 16:15:08 2019
BITPIX =      8 /
NAXIS =      2 /
NAXIS1 =    1246 / Number of bytes per row
NAXIS2 =    600 / Number of rows
PCOUNT =      0 / Random parameter count
GCOUNT =      1 / Group count
TFIELDS=      7 / Number of columns
COMMENT
COMMENT *** End of mandatory fields ***
COMMENT
DATE = '2019-07-17T16:15:08' / File creation date (YYYY-MM-DDThh:mm:ss UTC)
ORIGIN = 'RHESSI' / High Energy Solar Spectroscopic Imager
OBSERVER= 'ahamini' / Usually the name of the user who generated file
TELESCOP= 'RHESSI' / Name of the Telescope or Mission
INSTRUME= 'RHESSI' / Name of the instrument
OBJECT = 'Sun' / Object being observed
DATE_OBS= '2002-02-20T08:00:00.000' / nominal U.T. date when integration of this
DATE_END= '2002-02-20T08:40:00.000' / nominal U.T. date when integration of this
TIME_UNI=      1 /
ENERGY_L=    0.00000 /
ENERGY_H=    0.00000 /
TIMESYS= '1979-01-01T00:00:00' / Reference time in YYYY MM DD hh:mm:ss
TIMEUNIT= 'd' / Unit for TIMEZERO, TSTART1 and TSTOP1
GEOAREA =    277.130 /
DETUSED = 'SEGMENTS: 1F|3F|4F|5F|6F|8F|9F' /
SUMFLAG =      1 / Spectra summed over detectors
SUMCOINC=      0 /
COINCIDE=      0 / Anti-coincident spectra
RESPFILE= 'hsi_srm_20020220_080000.fits' /
```

Figure 8. Content of the Header

It contains a lot of information about the event for the observed period of time.

We can now pay attention to the data. This tells us that it is a 1-d array.

The important items for our current tasks are the data types(dtype). These are the data stored in the extension. Let's grab some of the data.

The rate is an array which has the count rate data in each energy channel. In order to perform conversion between rates and counts, we multiply accumulation time by rate.

Now we need to know what calibrated energies to which the channels correspond. Let's go into the ENEBAND extension and extract the energy bounds of each channel.

In order to convert the rates to a photon flux, we should divide the rate by area and energy bin width.

When we get the OSPEX units for Rate, Counts and Flux, it becomes possible to define the functions to plot. OSPEX software provides the data visualization for Spectrum plotting, Time profile plotting and Spectrogram. In our realization it was achieved and illustrated. All figures below demonstrate the plots for Python version of OSPEX.

First figure shows the Spectrum plotting for Rate.

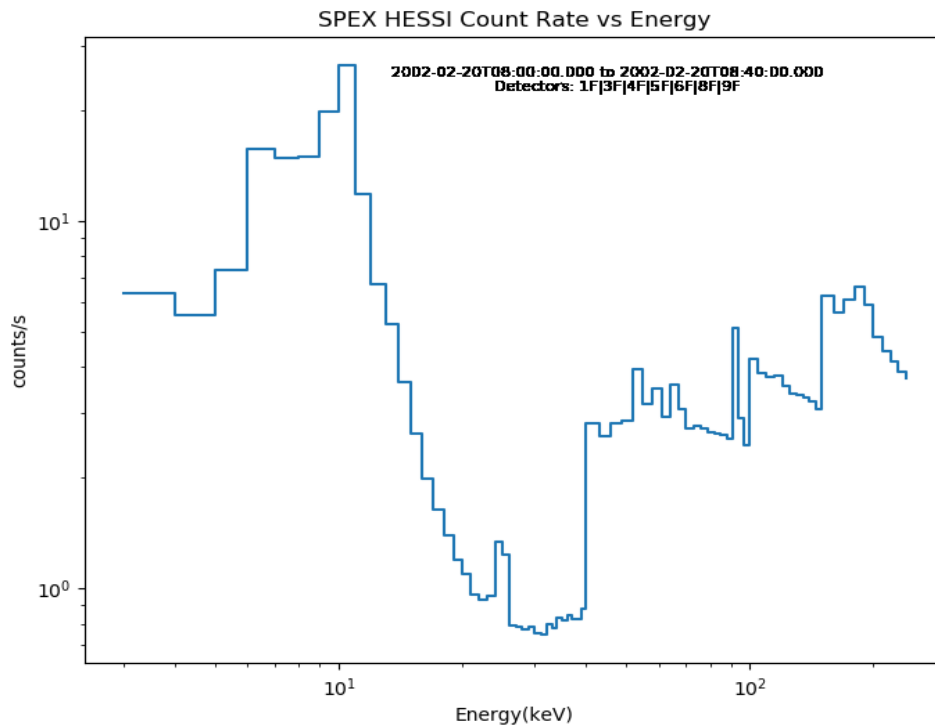


Figure 9. Spectrum plotting for Rate

There are a few things that we can take note. First, this is the energy scattering spectrum. It has the shape of the original photon spectrum convolved with both the effect of energy dispersion and the effective area of the detector. The area where we can observe the visible peak indicates the solar emission between 3 and 20 keV. An area starting at 40 keV is the background as well as a line around 20 keV. There's some information about detectors, start and end date of the considered event. This information directly loaded from the header file.

Next part of our work is to visualize the data in time profile. As a result, we propose the plot for Counts in Python version of OSPEX software. It contains the information about energy distribution for different channels relative to the time of the observed flare.

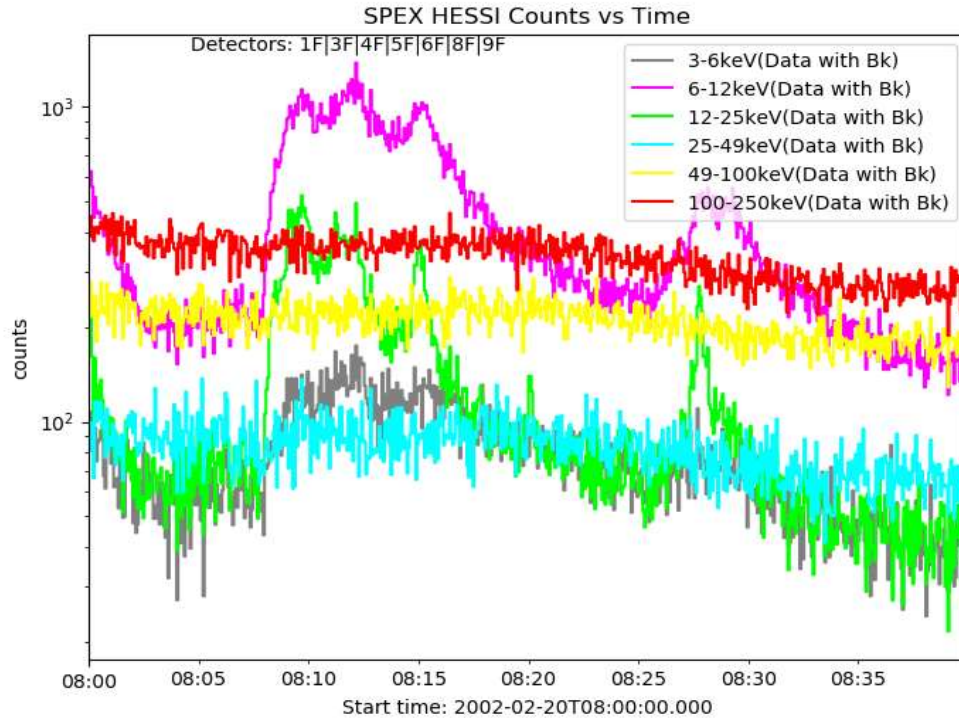


Figure 10. Time profile for Counts

Energy distribution for each channel is plotted in a specific color. This is done so that the user can easily determine the energy range of the lightcurve. This idea was reproduced from the original application written in IDL and implemented in our software using one of the Python libraries.

The following spectrogram, illustrated below, allows to perform the spectral analysis of the data.

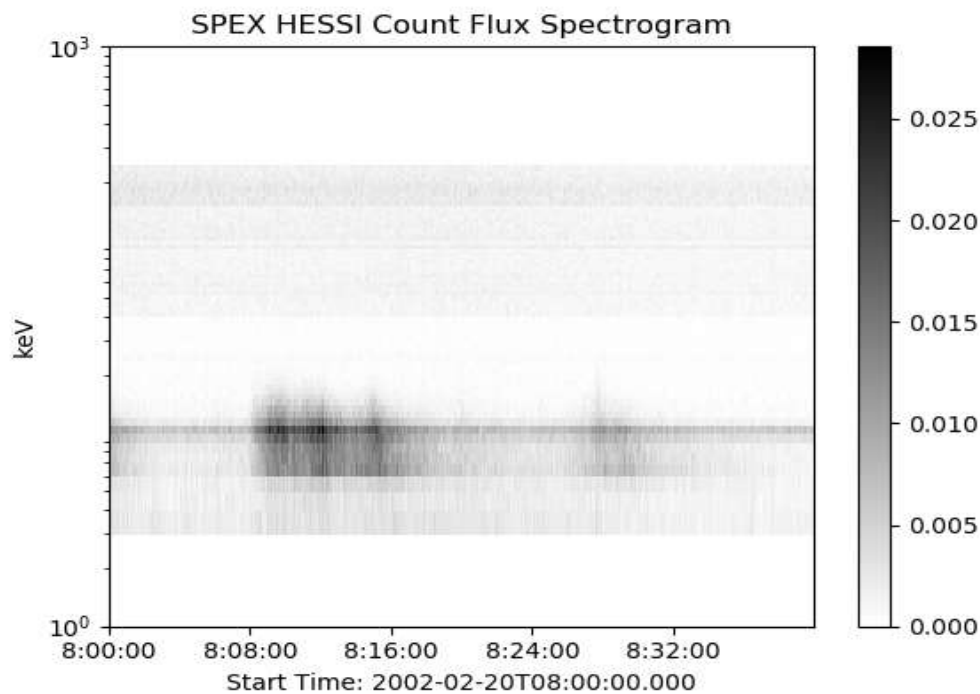


Figure 11. Spectrogram illustration for Flux

This is a function of counts as a function of energy and time. We can observe that the darker sections of the spectrogram in time intervals from about 08:00 to 08:30 show the intensity of the peaks of the observed event.

Now that we have extracted our count rate data, how do we get back to the emitted photon spectrum? We can guess the solution and test it examining instrument response file.

Most X-ray analysis instrument responses are stored in FITS files. The way they are stored depends on what is suitable for the instrument, but the end product is the same. Matrix describes the energy dispersion and effective area as a function of both photon energy and energy channel.

Let's open up a response file.

Filename: /home/stage/Bureau/InputData/SRM_test/hsi_srm_20020220_080000.fits

No.	Name	Ver	Type	Cards	Dimensions	Format
-----	------	-----	------	-------	------------	--------

0	PRIMARY	1	PrimaryHDU	26	()	
---	---------	---	------------	----	----	--

1	SPECRESP MATRIX	1	BinTableHDU	65	90R x 6C	[E, E, I, J, J, 77E]
---	-----------------	---	-------------	----	----------	----------------------

2	EBOUNDS	1	BinTableHDU	53	77R x 3C	[J, E, E]
---	---------	---	-------------	----	----------	-----------

3	SRM Object Parameters	1	BinTableHDU	114	1R x 35C	[I, 18B, B, B, B, 18B, 10B, B, E, E, I, 2E, E, I, D, 10B, 10B, E, B, B, B, 16A, 25A, 9A, 9A, I, I, 7E, 91E, 78E, 81E, 8A, 2D, 638A, 180A]
---	-----------------------	---	-------------	-----	----------	-------------------------------------------------------------------------------------------------------------------------------------------

Figure 12. Response file content

EBOUNDS extension is similar to the one found in the spectrum file. But we should always check with the instrument documentation if we see differences between the two.

The second extension is the SPECRESP MATRIX which contains the dispersion matrix.

The algorithm for reading these matrices is the same as reading a spectrum file.

We didn't use yet the instrument response file in Python version of OSPEX software. But we can illustrate the modeled RHESSI front-segment response to input photons at the energy = 50 keV if interested readers want to look at how it is done.

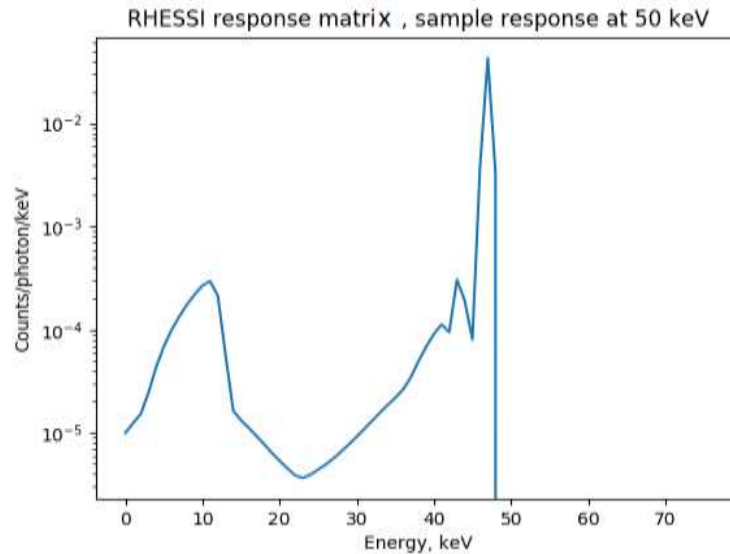


Figure 13. RHESSI response matrix: sample response at 50 keV

Converting from photons to counts is achieved through a matrix multiplication of the response matrix with the photon spectrum.

4.2 Software graphical user interface

The idea and content of the GUI were taken from the original version of the OSPEX software. The Tkinter library allowed us to implement the benefits, menu features and key widgets for managing the program in Python.

Main window contains the general information about the application options.

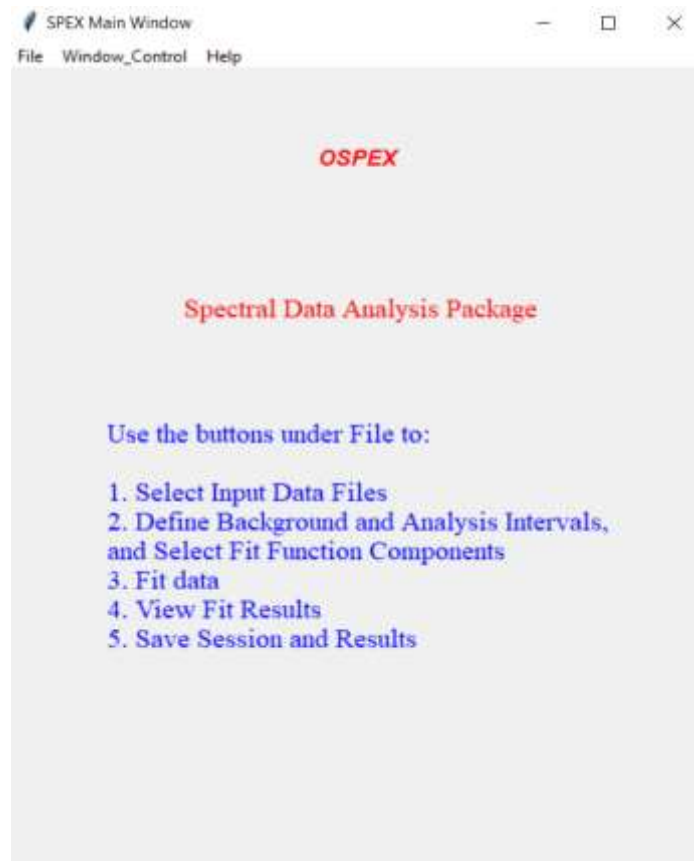


Figure 14. Python version of GUI

As we can see from the Figure, using application, we select the input data files, define the background and perform the fitting process.

At the top of the window is the menu bar which allows user to choose desired items. Menu “File” consist of the options to select input file, subtract the background and do fit. In our work we don’t deal with background.

Using “Help” menu we can find the information about creators of the application and their contacts.

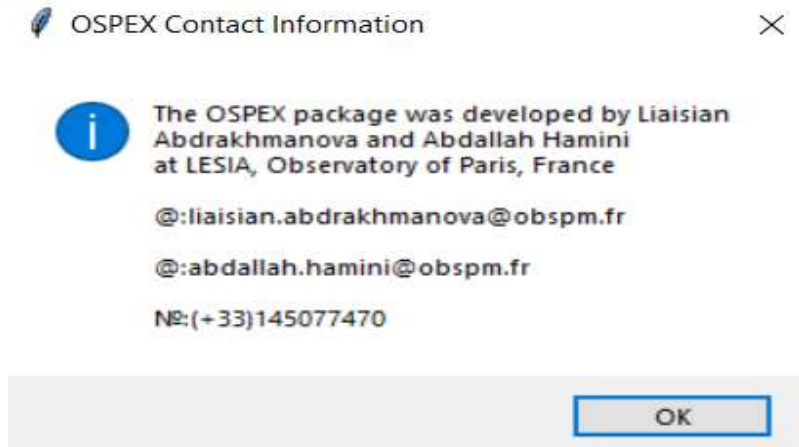


Figure 15. Contact information window

In addition to this, it is possible to go directly to the pages with official version of OSPEX software and new features added recently. The documentation of OSPEX is in HTML format.

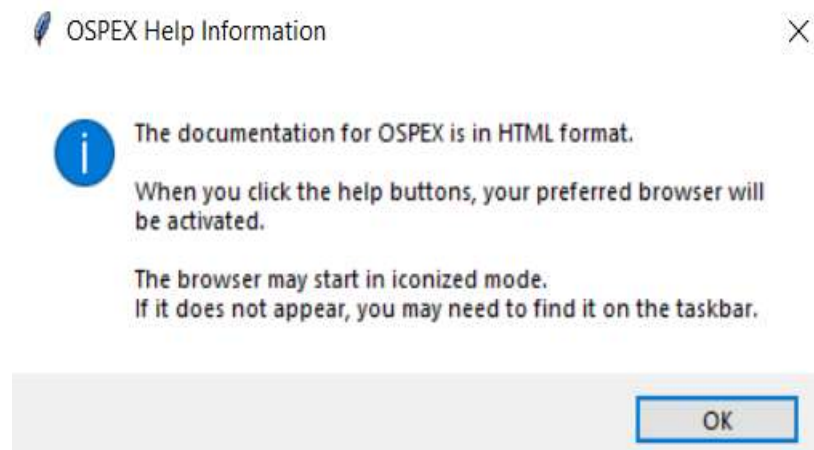


Figure 16. Help information window

Next, opening "Select Input" menu item we are in new window. Here we have the widgets to choose the sample file, read the information from header and data, visualize them, select analyzed intervals and do Spectrum, Time profile and Spectrogram plotting.



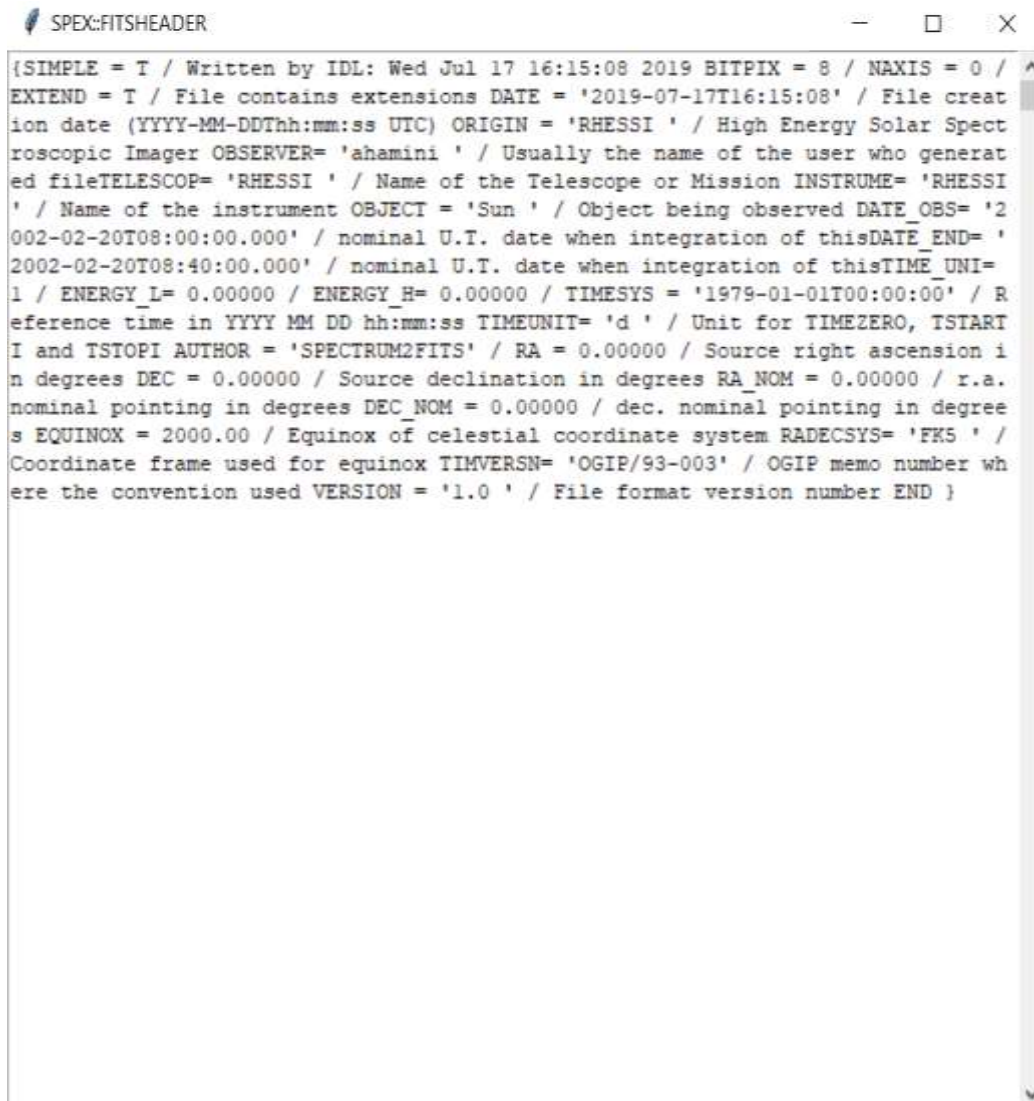
Figure 17. OSPEX Input Options window

Turning to the “Summarize” button, we have a generalized information about observed event.



Figure 18. “Header” window

Next widget “Header” load the text from the header and display it in new window. It is very convenient to work with content of the data in this form.



```
{SIMPLE = T / Written by IDL: Wed Jul 17 16:15:08 2019 BITPIX = 8 / NAXIS = 0 /
EXTEND = T / File contains extensions DATE = '2019-07-17T16:15:08' / File creat
ion date (YYYY-MM-DDThh:mm:ss UTC) ORIGIN = 'RHESSI' / High Energy Solar Spect
roscopic Imager OBSERVER= 'ahamini' / Usually the name of the user who generat
ed file TELESCOP= 'RHESSI' / Name of the Telescope or Mission INSTRUME= 'RHESSI
' / Name of the instrument OBJECT = 'Sun' / Object being observed DATE_OBS= '2
002-02-20T08:00:00.000' / nominal U.T. date when integration of this DATE_END= '
2002-02-20T08:40:00.000' / nominal U.T. date when integration of this TIME_UNI=
1 / ENERGY_L= 0.00000 / ENERGY_H= 0.00000 / TIMESYS = '1979-01-01T00:00:00' / R
eference time in YYYY MM DD hh:mm:ss TIMEUNIT= 'd' / Unit for TIMEZERO, TSTART
I and TSTOP I AUTHOR = 'SPECTRUM2FITS' / RA = 0.00000 / Source right ascension i
n degrees DEC = 0.00000 / Source declination in degrees RA_NOM = 0.00000 / r.a.
nominal pointing in degrees DEC_NOM = 0.00000 / dec. nominal pointing in degree
s EQUINOX = 2000.00 / Equinox of celestial coordinate system RADECSYS= 'FK5' /
Coordinate frame used for equinox TIMVERSN= 'OGIP/93-003' / OGIP memo number wh
ere the convention used VERSION = '1.0' / File format version number END }
```

Figure 19. “Summarize” window

Menu option “Select intervals and do fit” allows user to work in details with spectrum and apply physical models to forward fitting. This part will be discussed deeply in the next section.

4.3 Spectroscopic analysis

The question is how do we figure out what the true spectrum is? We have to guess. The method that we will use is called forward fitting. Basically the process is:

- assume a photon model spectrum with a set a variable parameters;
- convolve this spectrum with the instrument response to produce counts in the instruments energy space;
- compare these trial counts with the observed counts statistically;
- iterate until a convergence criterion is met.

There are 47 fitting functions in the original OSPEX version(Information for 2015). In order to evaluate the spectrum with function we should create an algorithm that would seek out the best values for variable functions and identify minimal data state of converging. Thus, users specify a model form for the flare spectrum, which can be combination of simple functions.

As we know from the previous chapters, two types of emission are analyzed: thermal and non-thermal. We started spectroscopic analysis with the first type.

4.3.1 Spectrum evaluation with ChiantiPy and SunXpex packages

Simulating spectral line and continuum intensities can be done using CHIANTI atomic database. The goal of the CHIANTI atomic database is to provide a set of atomic data for the interpretation of astrophysical spectra emitted by collisionally dominated, high temperature, optically thin sources. A complete set of ground level ionization and recombination rate coefficients has been assembled for all atoms and ions of the elements of H through Zn.

To study the behavior of thermal emission, we delved into the study of the variable thermal (vth) function, applied in OSPEX version written in IDL. This is a optically thin thermal bremsstrahlung radiation function as a differential spectrum seen at Earth in units of photon/(cm² s keV). Vth is relative to coronal abundance for Chianti.

OSPEX software written in IDL has an integrated software libraries, data bases, and system utilities which provide a common programming and data analysis environment to interact with CHIANTI database.

Since Python language is just beginning to be introduced into the field of solar flare research, the question arises of studying new tools to accomplish the task. Such a tool in our work is the ChiantiPy library. ChiantiPy was not developed to be a clone of the CHIANTI IDL code. Therefore, we could not accurately predict whether this package would be effective in our work. Despite this, we could draw lines and continuum for elements from the database and managed to build the lines for Iron 26. This element is an indicator of X-ray emission and helps to visually examine the specifics of the manifestation of lines in the wavelength range from 1 to 3 Angstroms. There are some examples of the plots from the ChiantiPy library are illustrated below.

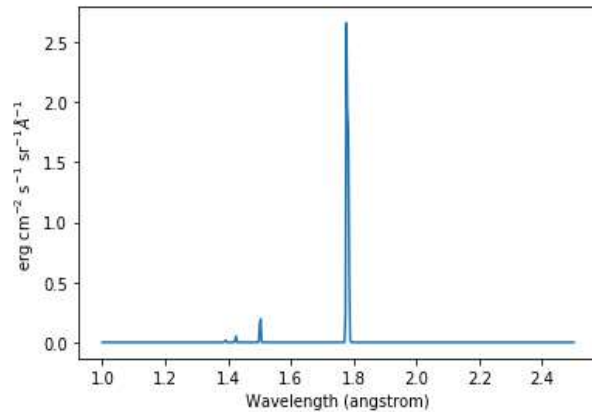


Figure 20. Spectra of one ion(Fe26)

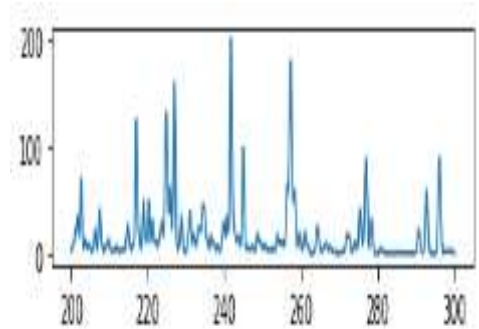


Figure 21. Spectra of multiple ions

The spectrum for a selection of all of the ions of the elements of H through Zn in the CHIANTI database can also be calculated.

This is a function of wavelengths for the intensities, loaded directly from Chianti database. Calculation is performed at emission measure = $1.e+27 \text{ cm}^{-3}$ and temperature = $2 \times 10^7 \text{ K}$.

The module continuum provides the ability to calculate the free-free and free-bound spectrum for a large number of individual ions. The value is set include all elements of Chianti database. The computation is performed for the same values as for lines. Next plots illustrate the results for free-free and free-bound continuum.

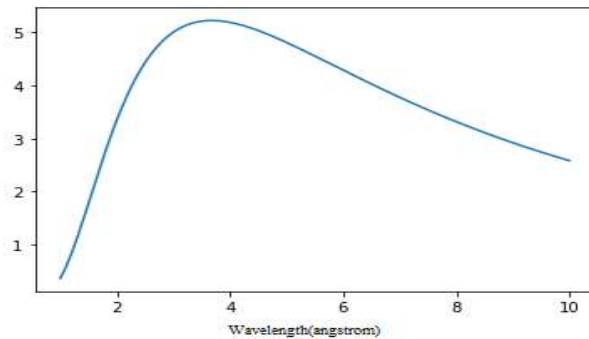


Figure 22. Free-free spectrum

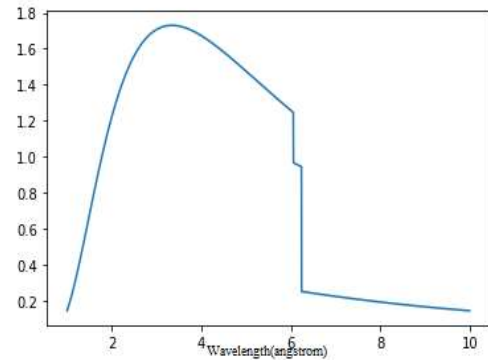


Figure 23 . Free-bound spectrum

Thus, we could provide the comparison of the spectra for free-free, free-bound transitions for coronal abundances of elements.

The main mechanisms and transitions responsible for plasma emission in the X-ray region are described in Chapter 2.

In next step, to understand how to build the variable thermal function from components we consider each method applied in it.

In fact, variable thermal function calls several methods. The main one, which allows to determine the ions and their abundances corresponding to X-ray emission, is called *Chianti_keV*. In order to restore this function in our program, we need to build chianti lines and continuum.

Since the creation of fitting models is an important task in the field of the study of solar flares, it was decided to contact a team based on working with RHESSI satellite data. After discussions, it turned out that NASA programmers began the development of the SunXpex package, which provides the modules responsible for the thermal radiation fitting part.

Our contribution was oriented towards determining the necessary input data and obtaining the output spectrum. The big advantage of this package is that it contains a module that allows to restore 50 abundances for X-ray emission. Using this method, we proceeded to the input data definition part:

- T = electron temperature in MK (may be a vector);
- Energy edges = two dimensional array of energies in keV, taken from the header;
- Emission measure = $1.e44 \text{ cm}^{-3}$, default value.

According to the fact that thermal emission since has a significant manifestation at the temperature range from 1 to 100 MK, we illustrate the results for this interval. Below are the results for temperatures of 10 MK and 30 MK. In addition to this fact, we are more interested in energy ranges up to 10 keV.

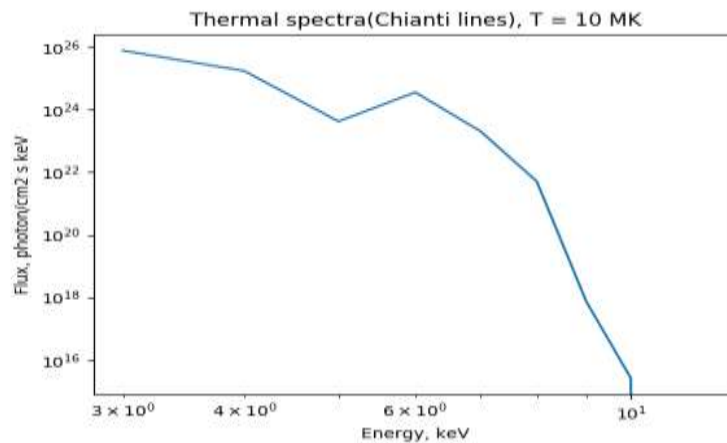


Figure 24. Thermal spectra at $T = 10 \text{ MK}$

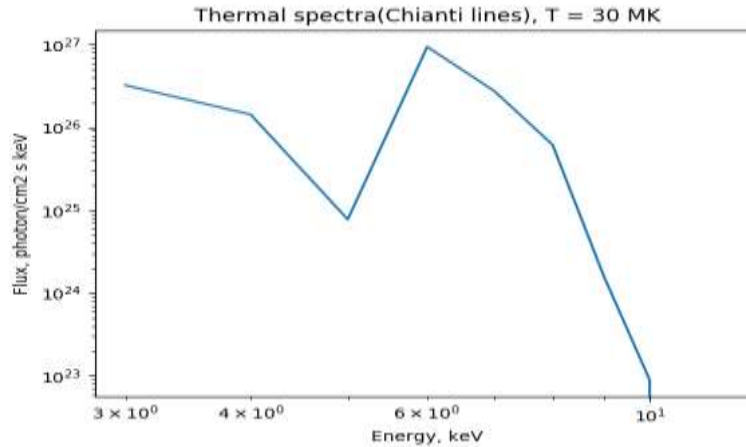


Figure 25. Thermal spectra at $T = 30$ MK

Thus, the figures show the chianti lines at given temperatures(10 MK, 30 MK) for energies from 3 keV to 10 keV. It should be noted that the temperature was converted to Astropy units in our computations.

Since there are no official sources about the use of SunXpex, this work shows the first results of this package.

4.3.2 Fitting process

Non-thermal emission is represented in solar flares by a One dimensional Power Law and Broken Power Law functions. Let's look at the components of each of them.

1. One dimensional Power Law parameters.

- amplitude – model amplitude at the reference energy;
- x – reference energy;
- alpha – power law index.

2. Broken Power Law function - One dimensional power law model with a break. Parameters:

- amplitude - model amplitude at the break energy;
- x_break – break energy;
- alpha 1 – power law index for $x < x_{\text{break}}$;
- alpha 2 – power law index for $x > x_{\text{break}}$.

Thus, we determined the input parameters for the functions. The next step is to select an algorithm which allows to find best fit values for these components.

Astropy library provides three algorithms in the fitting section. After testing all of them, it was revealed that the Levenberg – Marquardt algorithm, which was described in more detail in Chapter 3, was the most optimal. The next 2 figures show a comparison of the results for the

Levenberg – Marquardt algorithm(LevMarLSQFitter) and Sequential Least Squares Programming optimization algorithm(SLSQPLSFitter) .

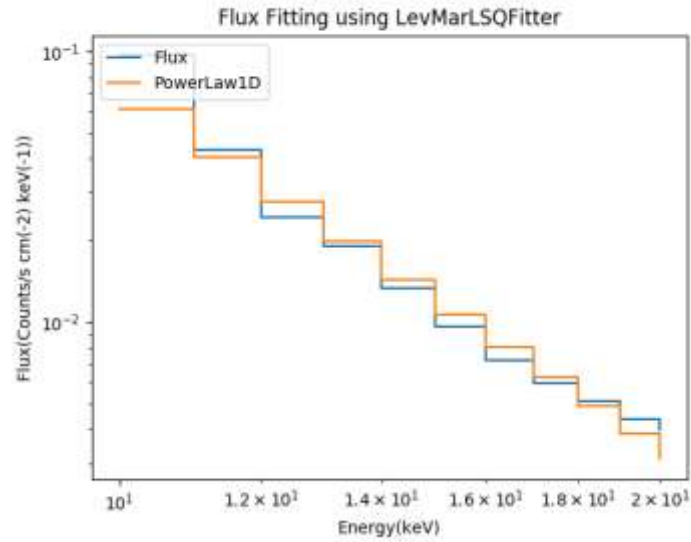


Figure 26. Levenberg – Marquardt algorithm for One Dimensional Power Law model

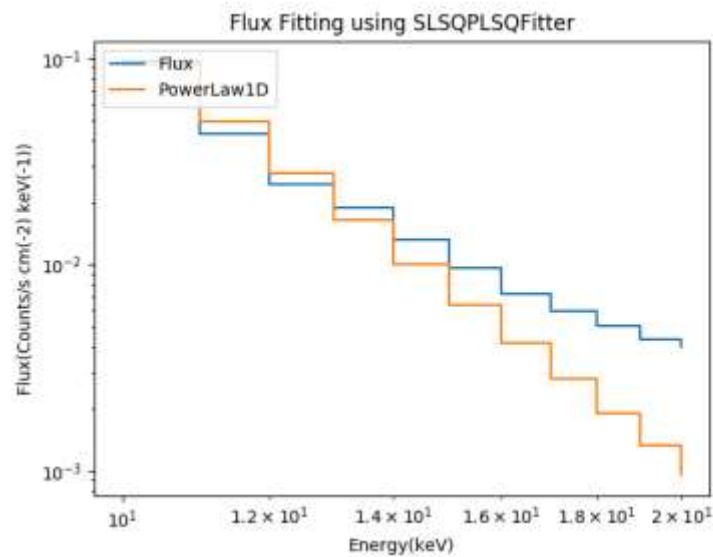


Figure 27. Sequential Least Square Programming optimization algorithm for One Dimensional Power Law model

As it can be seen from the figures this is a calculation of the photon fluxes of the photon model for each photon energy bound. Since the results for the Levenberg - Marquardt algorithm showed the best values for the fitting, we chose it as the main algorithm for our work.

In the analysis, we exclude the energy channels and time intervals are not viable for the processing. The figures are plotted for the energy range from 3 keV to 20 keV and time interval from 08:09 to 08:10.

Integrated residual is computed. Reduced chi squared value with LinearLSQFitter = 0.0016. Viewing the results, we can summarize that the algorithm satisfies our requirements and shows good fitting values for input data and minimal data state of converging.

This algorithm has also been tested with the One Dimensional Broken Power Law model for given intervals.

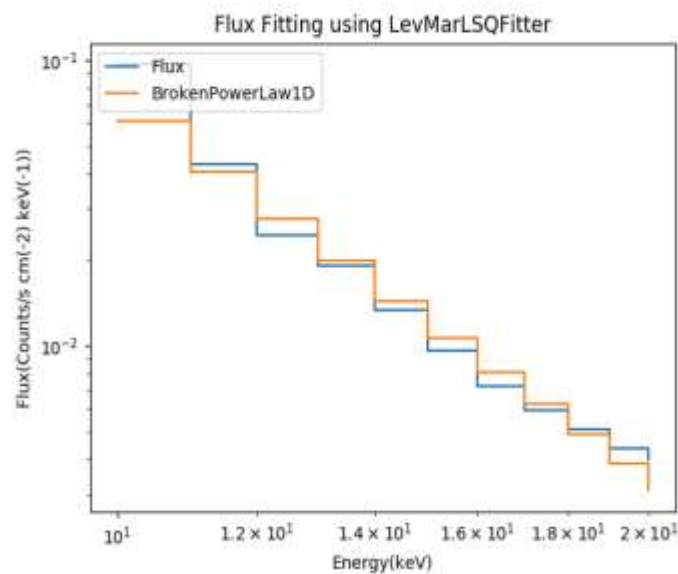


Figure 28. Levenberg – Marquardt algorithm for One Dimensional Broken Power Law model

The terminal displays the process of compiling the algorithm, parameters, proposed by the model, and the reduced chi squared value:

```
Model: PowerLaw1D
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Parameters:
    amplitude      x_0      alpha
-----
```

0.027 12.083 4.296

Optimization terminated successfully.

Current function value: 0.00014

Iterations: 29

Function evaluations: 149

Gradient evaluations: 29

Model: PowerLaw1D

Inputs: ('x',)

Outputs: ('y',)

Model set size: 1

Parameters:

amplitude	x_0	alpha
-----------	-----	-------

7.913	5.109	6.614
--------------	--------------	--------------

Model: BrokenPowerLaw1D

Inputs: ('x',)

Outputs: ('y',)

Model set size: 1

Parameters:

amplitude	x_break	alpha_1	alpha_2
-----------	---------	---------	---------

0.094	9.029	0.0	4.296
--------------	--------------	------------	--------------

Reduced Chi Squared with LinearLSQFitter: **0.0016**

Reduced Chi Squared with SLSQPLSQFitter: **0.0014**

This procedure has been integrated into the GUI. Fitting parameters can be selected by referring to the “Select Fit Options and Do Fit” menu option.

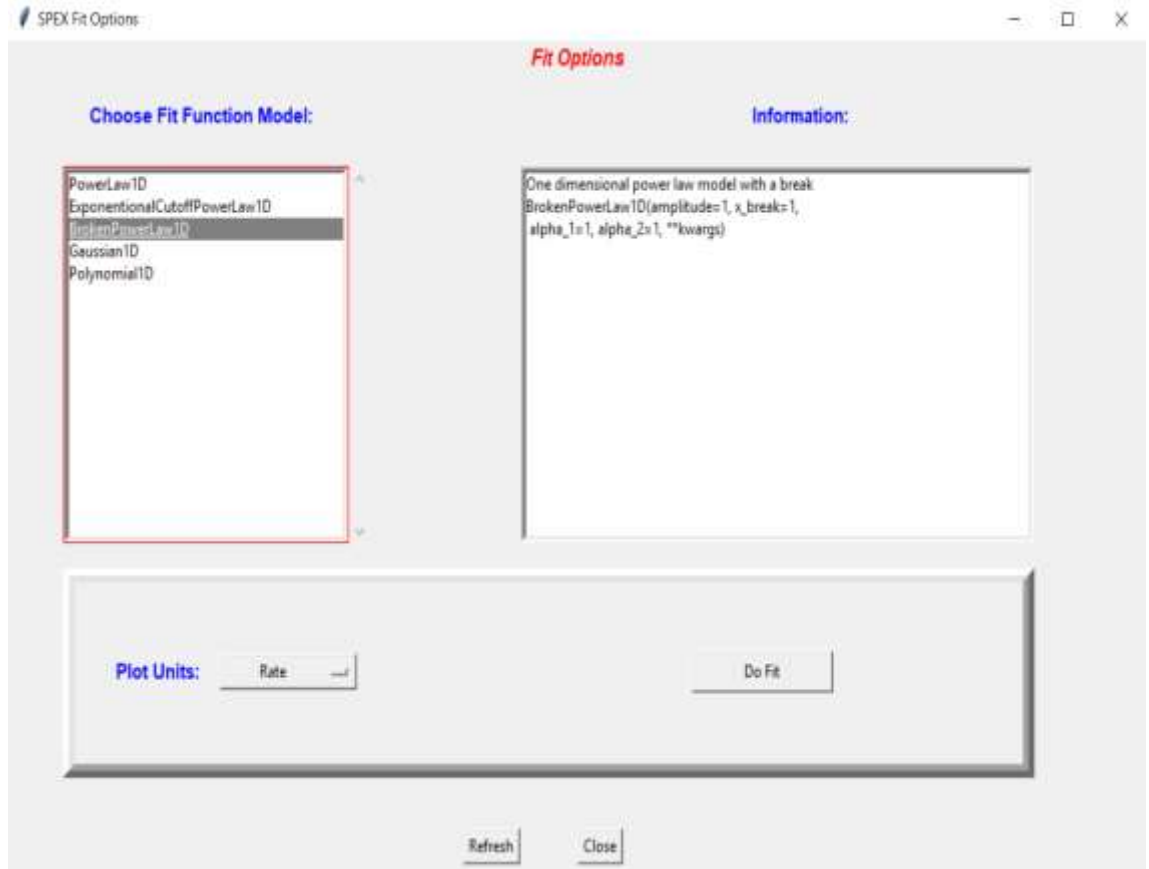


Figure 29. Fitting menu window

This is the fitting menu window. As we see, the user can select the desired model. After selection, information about this function is displayed in the right frame. After pressing the “Do fit” button, iteration takes place to find the best parameters for the observed function and the final plot is displayed in a new window.

Conclusion

This work is based on the development of the application in the Python programming language which implements part of the OSPEX software. This application allows to perform spectral analysis of RHESSI data.

During the internship, the tasks set were realized and the following results were obtained:

1. The analysis of FITs files has been performed, which are the input data of the application. A detailed review of the file parameters made it possible to restore the energy distributions of electrons in solar flares for analysis and their spectra. Based on them we could visualize and plot the data.

The big advantage of using the Python language is the user can easily take the features of the package to manage plots, select desired intervals, axes, visualize data on multiple windows simultaneously, save the analysis results in text format or as an image. It is worth noting that the OSPEX application written in IDL does not allow working with plots directly. To do this, it has a separate menu option. We excluded this item in our work, thereby improving the functionality of the application.

2. A new interface has been developed for an application in the Python programming language. Since this language is more modern than IDL, it allowed to build a more compact GUI. It contains basic widgets for working with data. As mentioned above, graphic management is more versatile. Data visualization windows are flexible, which allows the user to analyse information in different formats.

IDL does not have such properties. The user has to close the current window in order to perform a new action in the application. To save results and manage data, the software requires to turn to additional menu options, which are contained in a separate block. This creates a complex task and actions that can affect the speed and quality of the work.

The new interface is able to simplify the task by implementing the benefits and tools of the Python language.

3. Plots were obtained for the thermal spectrum (line and continuum) using the new ChiantiPy and SunXpex libraries.

It is worth noting that the SunXpex package became available for public use only in March 2019. This package is under development. Our contribution allowed us to evaluate the first results. Contacts and discussions with SunXpex developers have provided new ideas for further OSPEX software upgrades.

The ChiantiPy library displays the results of using the CHIANTI database in Python. This database makes a great contribution to understanding the structure and behavior of processes in solar flares.

4. A technique for fitting performance based on the Levenberg – Marquardt algorithm has been developed. Determining the input parameters for the One Dimensional Power Law and One Dimensional Broken Power Law functions allowed us to estimate the convergence of the real

spectrum with the observed models. The results showed that the values of the parameters proposed by the algorithm display a good data approximation. We calculated the chi-squared value, which is an indicator of data convergence. The value for the Levenberg – Marquardt algorithm showed greater convergence of the spectra than the result in the original version of the OSPEX application.

In addition to this fact, a new interface has also been proposed for fitting part. It allows the user to select the functions for the fitting, get information about the parameters and plot the fitted graphs in a new window.

5. Our project was successfully presented in the GitHub repository. It provides information about the software realization, installation and testing. Interested users can contribute to the project.

The cross-platform is a great achievement of the OSPEX application created in Python. It was tested on all operating systems and showed performance without technical failures.

As a future work, it is planned to implement the background extraction part and realize additional functional models for fitting. In addition, the goal is to include the data for processing from the STIX instrument. This will expand the versatility, capabilities of the program and make a great contribution to the study of solar flares.

References

- [1] Aschwanden M.J. Particle acceleration and kinematics in solar flares – a synthesis of recent observations and theoretical concepts (Invited Review) // Space Science Reviews. – 2002. – V.101. – №1. – P.1–227.
- [2] Kontar E.P., Brown J.C., Emslie A.G. et al. Deducing electron properties from hard X-ray observations // Space Sci. Rev. – 2011. – V.159. – №1-4. – P.301-355.
- [3] Aschwanden M.J. Physics of the solar corona. An introduction with problems and solutions (2nd edition) // Springer. – P.2005. – 892.
- [4] Kuznetsov V.D. Solar-Terrestrial Physics: Results of experiments on the CORONAS-F satellite // Fizmatlit - P.2009. – 488.
- [5] Lin R.P., Dennis B.R., Hurford G.J. et al. The Reuven Ramaty High-Energy Solar Spectroscopic Imager (RHESSI) // Solar Phys. – 2002. – V.210. – № 1. – P.3-32.
- [6] <http://lesia.obspm.fr/STIX-description-de-l-instrument.html>
- [7] Carmichael H. A process for flares // NASA Special Publication. – 1964. – V.50. – P.451-456.
- [8] Sturrock P.A. Model of the high-energy phase of solar flares // Nature. – 1966. – V.211. – №5050. – P.695–697.
- [9] Kopp R.A., Pneuman G.W. Magnetic reconnection in the corona and the loop prominence phenomenon // Solar Phys. – 1976. – V.50. – P.85-98.
- [10] Shibata K., Masuda S., Shimojo M. et al. Hot-plasma ejections associated with compact-loop solar flares // Astrophys. J. Lett. – 1995. – V.451. – P.L83-L85.
- [11] Shibata K. Evidence of magnetic reconnection in solar flares and a unified model of flares // Astrophys. and Space Sci. – 1999. – V.264. – №1/4. – P.129-144.
- [12] Priest E.R., Forbes T. Magnetic reconnection: MHD theory and applications // Cambridge University Press. – 2000. – 616 p.
- [13] Caspi A. Distinguishing Between Thermal and Non-Thermal Electrons in Solar Flares Using X-Ray Observations – 2005 – P. 1.

[14] Krucker S., Battaglia M., Cargill P.J. et al. Hard X-ray emission from the solar corona // Astron. & Astrophys. Rev. – 2008. – V.16. – P.155-208.

[15] Holman G.D., Aschwanden M.J., Auras H. et al. Implications of X-ray observations for electron acceleration and propagation in solar flares // Space Sci. Rev. – 2011. – V.159. – №1-4. – P.107-166.

[16] <https://hesperia.gsfc.nasa.gov/rhessi3/home/mission-concept/mission-concept/index.html>

[17] <https://hesperia.gsfc.nasa.gov/~schmahl/>

[18] https://hesperia.gsfc.nasa.gov/ssw/packages/spex/doc/ospex_explanation.htm

[19] Dere K.P., Landi E., Mason H.E. et al. CHIANTI - an atomic database for emission lines // A & A Supplement series. – 1997. – V.125. – P.149-173.

[20] Landi E., Feldman U., Dere K.P. CHIANTI - an atomic database for emission lines. V. Comparison with an isothermal spectrum observed with SUMER // Astrophys. J. Suppl. Ser. – 2002. – V.139. – №1. – P.281-296.

[21] Grim G. Plasma spectroscopy // Atomizdat . – 1969. P.452.

[22] <https://heasarc.nasa.gov/xanadu/xspec/manual/node10.html>

[23] Samuel W. Hasinoff, Google Inc. Photon, Poisson noise – 2012 – P. 1-4.

[24] https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm

[25] <https://docs.astropy.org/en/stable/>

[26] Dere K.P, ChiantiPy Documentation – 2018 – P. 8-29.

[27] <https://github.com/DanRyanIrish/sunxspex>

[28] <https://tkdocs.com/tutorial/widgets.html>