The Astrophysical Journal Supplement Series, 267:6 (24pp), 2023 July https://doi.org/10.3847/1538-4365/acd343

Data-constrained Solar Modeling with GX Simulator

Gelu M. Nita[1], Gregory D. Fleishman[1,2], Alexey A. Kuznetsov[3], Sergey A. Anfinogentov[3], Alexey G. Stupishin[4], Eduard P. Kontar[5], Samuel J. Schonfeld[6], James A. Klimchuk[7], and Dale E. Gary[1]
[1] New Jersey Institute of Technology, Newark 07102-1982, NJ, USA; gelu.m.nita@njit.edu
[2] Leibniz-Institut für Sonnenphysik (KIS), Freiburg, D-79104, Germany
[3] Institute of Solar-Terrestrial Physics, Irkutsk 664033, Russia
[4] Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034 Russia
[5] University of Glasgow, Glasgow, G12 8QQ, UK
[6] Institute for Scientific Research, Boston College, Newton, MA 02459, USA
[7] NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA
Received 2023 January 2; revised 2023 March 24; accepted 2023 April 18; published 2023 June 29

Abstract

To facilitate the study of solar flares and active regions, we have created a modeling framework, the freely distributed GX Simulator IDL package, that combines 3D magnetic and plasma structures with thermal and nonthermal models of the chromosphere, transition region, and corona. Its object-based modular architecture, which runs on Windows, Mac, and Unix/Linux platforms, offers the ability to either import 3D density and temperature distribution models, or to assign numerically defined coronal or chromospheric temperatures and densities, or their distributions, to each individual voxel. GX Simulator can apply parametric heating models involving average properties of the magnetic field lines crossing a given voxel, as well as compute and investigate the spatial and spectral properties of radio, (sub)millimeter, EUV, and X-ray emissions calculated from the model, and quantitatively compare them with observations. The package includes a fully automatic model production pipeline that, based on minimal users input, downloads the required SDO/HMI vector magnetic field data, performs potential or nonlinear force-free field extrapolations, populates the magnetic field skeleton with parameterized heated plasma coronal models that assume either steady-state or impulsive plasma heating, and generates non-LTE density and temperature distribution models of the chromosphere that are constrained by photospheric measurements. The standardized models produced by this pipeline may be further customized through specialized IDL scripts, or a set of interactive tools provided by the graphical user interface. Here, we describe the GX Simulator framework and its applications.

Unified Astronomy Thesaurus concepts: Solar active regions (1974); Solar flares (1496); Microwave spectroscopy (2251); Solar electromagnetic emission (1490); Astronomy data modeling (1859); Nonthermal radiation sources (1119)

1. Introduction

The fundamental problems of modern solar physics require analysis of multiple vast data sets obtained with a multitude of ground- and space-based instruments. The sheer level of complexity in newly available data sets calls for adequate theoretical modeling in order to derive the target physical parameters of a given measurement. Examples include extrapolating the photospheric magnetic field data from optical observations, or deducing the distribution of thermal coronal plasma from extreme ultraviolet (EUV) observations. Larger-caliber theoretical and modeling efforts are needed to meaningfully combine and cross-validate multiple data sets. These data come from space missions, e.g., Helioseismic and Magnetic Imager (HMI; Scherrer et al. 2012) on board the Solar Dynamic Observatory (SDO; Pesnell et al. 2012), as well as ground-based high-resolution optical and infrared (IR) instruments such as Goode Solar Telescope (GST; Cao et al. 2010; Goode & Cao 2012) and Daniel K. Inouye Solar Telescope (DKIST; Rimmele et al. 2010; Tritschler et al. 2016). Fundamental enhancements of theory and modeling are demanded in order to fully exploit new microwave and millimeter-wave imaging spectropolarimetry data from the Expanded Owens Valley Solar Array (EOVSA; Nita et al. 2016; Gary et al. 2018), the Siberian Radio Heliograph (SRH; Lesovoi et al. 2017; Altyntsev et al. 2020), and the Atacama Large Millimeter/submillimeter Array (ALMA), in addition to more traditional radio, X-ray, and EUV data, e.g., from the Nobeyama Radioheliograph (Nakajima et al. 1994), Submillimeter Solar Telescope (Kaufmann et al. 2001), RHESSI (Lin et al. 2003), the Atmospheric Imaging Assembly (AIA; Lemen et al. 2012) on board SDO, or the Spectrometer/Telescope for Imaging X-rays (STIX; Krucker et al. 2020). Dynamic solar phenomena that constitute solar activity either occur in or are sensitive to the physical conditions in the solar corona. The dominant form of energy in the solar corona is magnetic energy; thus, knowledge of the coronal magnetic field is central for understanding coronal physics. However, there is no observational technique that provides the 3D magnetic vector field over a significant coronal volume. This is why data-constrained modeling of the coronal magnetic field is extremely important.

The GX Simulator modeling framework that we present here is based on a magnetic model (magnetic skeleton), which, once created, can be populated by thermal plasma and nonthermal particles, and then various emissions can be computed from the volume and compared with observations. When all synthesized observables match all available data, the model is proved to be valid.

Our modeling framework solves the following challenges: (i) automated creation of the magnetic model; (ii) addition of an objectively defined thermal structure of the corona and chromosphere; (iii) rigorous calculation of radio, EUV, and X-ray continuum emission from the model; and (iv) provision for model-to-data comparison. To facilitate the creation and manipulation of the models, the tool offers numerous options. Earlier versions of the tool were described by Nita et al. (2015) for flare science and Nita et al. (2018) for active region (AR) science. This paper summarizes the functionality of those initial versions and describes numerous updates and enhancements of the tool.

2. The GX Simulator Modeling Package

The current version of the GX Simulator modeling tool consists of a collection of interconnected programs developed using the Interactive Data Language (IDL) programming environment, and it is integrated as an optional package in the community contributed, open-source SolarSoftWare (SSW; Freeland & Handy 1998) public repository. Therefore, the GX Simulator package must be initially installed along with the SSWIDL environment following the instructions found at `SolarSoftWare`. However, given the reliance of its core functionality on external libraries developed on FORTRAN and C++, it is strongly recommended that users overwrite the initial SSW installation of GX Simulator by following the installation instructions provided in the `README.md` file included in its GitHub repository (`Gelu-Nita/GX_SIMULATOR`), which also provides platform specific steps that should be taken in order to ensure full functionality on Windows, Mac, and Unix/Linux systems.

As an open-source software package, GX Simulator integrates several GitHub submodules developed and maintained by collaborators, which may be updated independently, sometimes on a daily basis. GX Simulator does not currently rely on a strict, numerical versioning system, but rather the date of its most recent GitHub update can be used in lieu of a version number. Nevertheless, one may consider the publication dates of the previous papers describing the GX Simulator functionality as major revisions of the package, i.e., GX v1.0 (Nita et al. 2015), GX v2.0 (Nita et al. 2018), and GX v3.0 (this paper[8]).

To ensure that the most up-to-date version is installed on a local system, rather than relying on the possibly delayed SSW upgrading cycle, it is recommended that users follow the upgrading procedure described on its GitHub repository web page, which will ensure synchronization with the most up-to-date version of both the main code and all external submodule dependencies.

3. The GX Simulator Automatic Model Production Pipeline

3.1. *General Description of the Pipeline Functionality*

To facilitate the use of the GX Simulator modeling package (Nita et al. 2015, 2018), we have developed a fully automatic model production pipeline (AMPP) that, based on minimal input from the user, downloads the required vector magnetic field data produced by the Helioseismic and Magnetic Imager (HMI) on board the Solar Dynamics Observatory (SDO; Scherrer et al. 2012) and (optionally) the contextual Atmospheric Imaging Assembly (AIA; Lemen et al. 2012) maps, performs potential and/or nonlinear force-free field (NLFFF) extrapolations, populates the magnetic field skeleton with parameterized heated plasma coronal models that assume either steady-state or impulsive plasma heating, and generates non-LTE density and temperature distribution models of the chromosphere that are constrained by photospheric measurements. The standardized models produced by this pipeline may be further customized through a set of interactive tools provided by the GX Simulator graphical user interface (GUI).

The AMPP submodule is exposed to the users through a single top-level IDL routine, namely `gx_fov2box.pro`, which provides a series of options that may be used to customize its functionality, as detailed in Appendix A, where we provide an AMPP script to generate a magneto-thermal model for an instance of AR11520 observed on 2012 July 12 04:58:26 UT, which we use as an illustrative example in the subsequent sections.

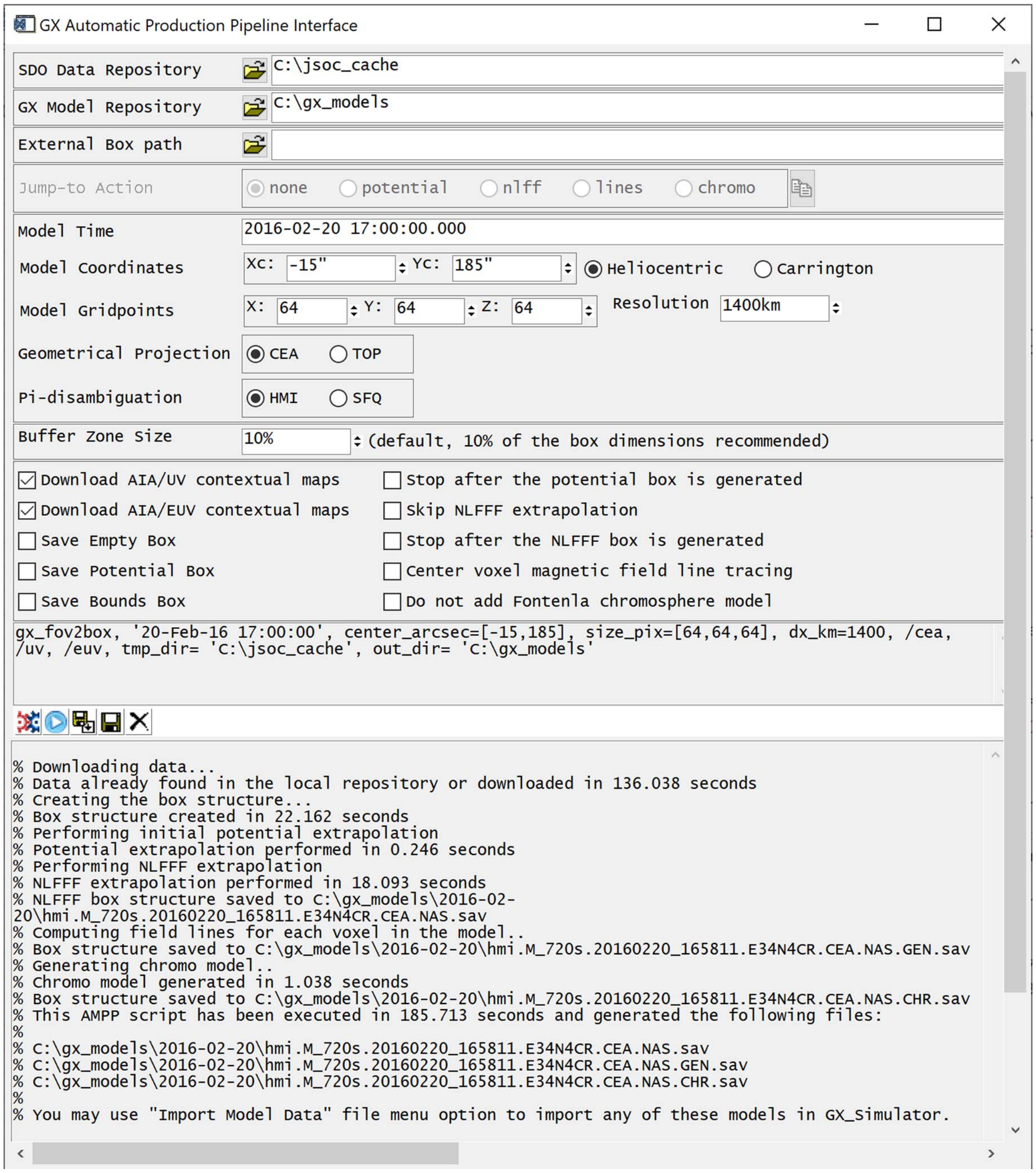
The GX Simulator package also provides a standalone GUI application, `gx_ampp.pro`, which may be used to conveniently generate and run AMPP scripts, as illustrated in Figure 1, which displays a set of default settings and corresponding runtime messages that match the demo script `create_box_20160220.pro` included in the `demo` subfolder of the GX Simulator distribution.

Any interactive change of the input fields of the `gx_ampp.pro` GUI updates the functional `gx_fov2box.pro` script, which may be launched from the interface, or copied and run directly from the IDL command line. The `gx_ampp.pro` GUI also provides the option of uploading an already existing GX Simulator–compatible box structure (such as a potential or NLFFF extrapolation box), which may be used as a starting point for adding properties to the model, such as optional magnetic field tracing parameters and/or chromosphere models, as described in the subsequent sections.

Figure 2 illustrates the main building blocks and the workflow of the GX Simulator AMPP module, and Figure 3 displays a series of snapshots of the 3D magnetic model produced by the AMPP script provided in Appendix A. The initialization of an AMPP run, illustrated by the first two blocks of the workflow diagram shown in Figure 2, requires only the time, field of view (FOV), height, and desired spatial resolution of the model. The time and location input parameters are used by the AMPP to identify and download the available SDO HMI/AIA maps closest to the requested time, after checking the specified local repository in case they were already downloaded during a previous AMPP run.

These input SDO HMI/AIA data products are used to prepare a data structure and the boundary conditions needed to perform the subsequent AMPP tasks (blocks 3 and 4 of Figure 2). To do so, the AMPP creates an initial empty volume structure on top of the photospheric vector magnetogram boundary conditions that are prepared by performing Carrington-heliographic or helioprojective-Cartesian projection (Thompson 2006), as illustrated in panels (a) and (b) of Figure 3. Unlike the standard, general-purpose Spaceweather HMI Active Region Patch (SHARP; Bobra et al. 2014) data products routinely used as boundary conditions by other

[8] In addition, copies of GX Version 3.0+ releases are being deposited to `Gelu-Nita/GX_SIMULATOR: gx_simulator` on Zenodo (Nita & Kontar 2023).



GX Automatic Production Pipeline Interface

SDO Data Repository	C:\jsoc_cache
GX Model Repository	C:\gx_models
External Box path	
Jump-to Action	[✓ none] potential nlff lines chromo
Model Time	2016-02-20 17:00:00.000
Model Coordinates	Xc: -15" Yc: 185" [✓ Heliocentric] Carrington
Model Gridpoints	X: 64 Y: 64 Z: 64 Resolution 1400km
Geometrical Projection	[✓ CEA] TOP
Pi-disambiguation	[✓ HMI] SFQ
Buffer Zone Size	10% (default, 10% of the box dimensions recommended)

✓ Download AIA/UV contextual maps	☐ Stop after the potential box is generated
✓ Download AIA/EUV contextual maps	☐ Skip NLFFF extrapolation
☐ Save Empty Box	☐ Stop after the NLFFF box is generated
☐ Save Potential Box	☐ Center voxel magnetic field line tracing
☐ Save Bounds Box	☐ Do not add Fontenla chromosphere model

```
gx_fov2box, '20-Feb-16 17:00:00', center_arcsec=[-15,185], size_pix=[64,64,64], dx_km=1400, /cea,
/uv, /euv, tmp_dir= 'C:\jsoc_cache', out_dir= 'C:\gx_models'
```

```
% Downloading data...
% Data already found in the local repository or downloaded in 136.038 seconds
% Creating the box structure...
% Box structure created in 22.162 seconds
% Performing initial potential extrapolation
% Potential extrapolation performed in 0.246 seconds
% Performing NLFFF extrapolation
% NLFFF extrapolation performed in 18.093 seconds
% NLFFF box structure saved to C:\gx_models\2016-02-
20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.sav
% Computing field lines for each voxel in the model..
% Box structure saved to C:\gx_models\2016-02-20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.GEN.sav
% Generating chromo model..
% Chromo model generated in 1.038 seconds
% Box structure saved to C:\gx_models\2016-02-20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.CHR.sav
% This AMPP script has been executed in 185.713 seconds and generated the following files:
%
% C:\gx_models\2016-02-20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.sav
% C:\gx_models\2016-02-20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.GEN.sav
% C:\gx_models\2016-02-20\hmi.M_720s.20160220_165811.E34N4CR.CEA.NAS.CHR.sav
%
% You may use "Import Model Data" file menu option to import any of these models in GX_Simulator.
```

Figure 1. Snapshot of the gx_ampp GUI application displaying a set of default settings and corresponding runtime execution messages that match the demo script `create_box_20160220.pro` included in the `demo` subfolder of the GX Simulator distribution.

magnetic field reconstruction packages, the AMPP boundary condition maps are exactly centered on the user-requested FOV, which minimizes, to the greatest extent possible, the unavoidable projection effects.

In the next stage (blocks 5 and 6 of Figure 2), the AMPP applies the method described in Section 3.2.2 to produce an initial potential field extrapolation 3D structure, which is used in the next stage as an initial condition for generating an NLFFF model using the optimization code described in Section 3.2.3. If not explicitly disabled by the user, the next AMPP block computes the averaged magnetic field $\langle B \rangle$ and length L of the potential or NLFFF magnetic field lines crossing each volume voxel. This enables GX Simulator to interactively dress the magnetic skeleton with a parameterized thermal structure, as detailed in Section 3.5. Panels (c) and (d) in Figure 3 illustrate a series of magnetic field lines and their associated magneto-thermal structure, corresponding to the NLFFF magneto-thermal model generated by the AMPP script presented in Appendix A. Finally, if not explicitly disabled by the user, the last block of the AMPP workflow diagram replaces the bottom layers of the potential or NLFFF model with a non-LTE chromosphere model, as described in Section 3.4.

As illustrated in Figure 1, there is a set of optional keyword switches to skip some optional execution blocks and/or to save, in addition to the final model, any intermediary models generated by the workflow. Thus, to help distinguish these AMPP products without the need to inspect the output files, we

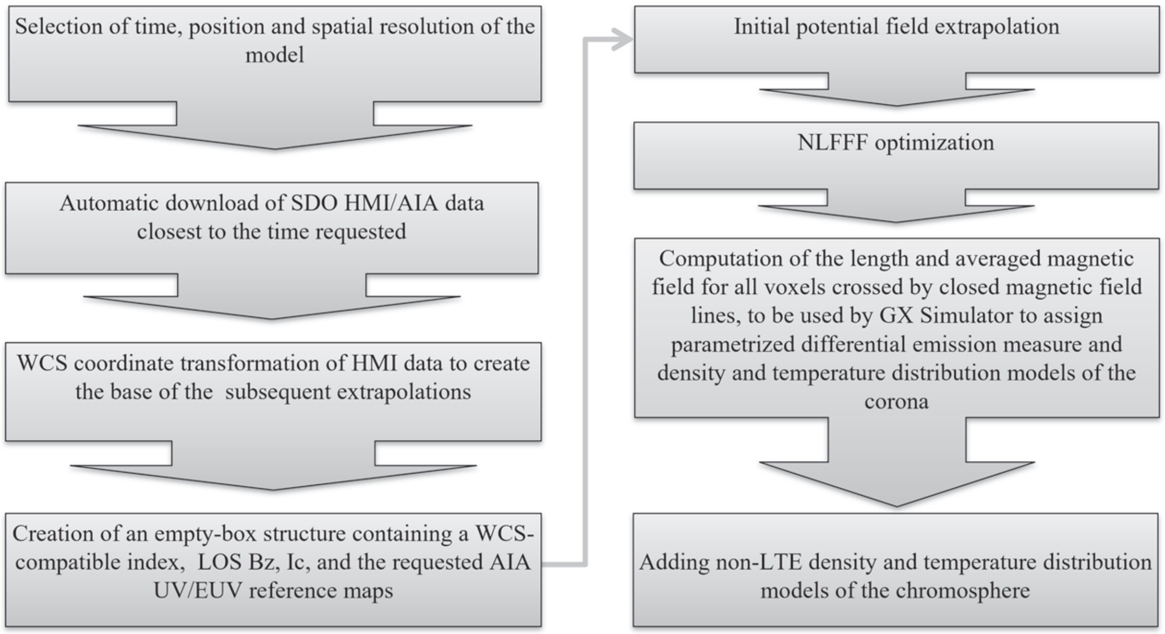

Selection of time, position and spatial resolution of the model
Automatic download of SDO HMI/AIA data closest to the time requested
WCS coordinate transformation of HMI data to create the base of the subsequent extrapolations
Creation of an empty-box structure containing a WCS-compatible index, LOS Bz, Ic, and the requested AIA UV/EUV reference maps
Initial potential field extrapolation
NLFFF optimization
Computation of the length and averaged magnetic field for all voxels crossed by closed magnetic field lines, to be used by GX Simulator to assign parametrized differential emission measure and density and temperature distribution models of the corona
Adding non-LTE density and temperature distribution models of the chromosphere

Figure 2. GX Simulator Automatic Model Production Pipeline workflow.

have adopted a file-naming convention that combines a series of distinctive tags that uniquely identify each type of model, as listed in Table 1. For example, an AMPP output file tagged as "`.NAS.GEN.CHR.`" would indicate a NLFFF model augmented by adding $\langle B \rangle$–L properties and a non-LTE chromosphere, while "`.POT.GEN.CHR.`" would denote that the same additional properties were added to a potential magnetic field model, if the user chooses to skip the NLFFF optimization block. Any IDL structure produced by the AMPP contains a string tag named "EXECUTE," which provides an exact copy of the execution script used to generate it.

3.2. AMPP NLFFF Magnetic Field Models

3.2.1. Preparation of the Boundary and Initial Conditions for the NLFFF Extrapolation

The first stage of the AMPP is the production of initial and boundary conditions for the subsequent NLFFF extrapolation. The user provides AR coordinates, observation time, and the size and spatial resolution for the resulting 3D data cube. Then the pipeline will download the required data and produce a data cube with the photospheric measurements of the magnetic field vector in its bottom layer. The rest of the volume will be filled with the extrapolated potential field. These operations are fully automated and do not require any additional actions from the user. The flowchart of production of the initial and boundary condition is shown in Figure 4. The individual steps of this AMPP stage are described below.

First, the pipeline script automatically downloads, from the JSOC data-processing center, SDO/HMI vector magnetograms (data series:hmi.B_720s) taken at the time closest to the time requested by a user. For further processing, the limited field-of-view (FOV) maps are cut out from the full-Sun magnetograms. The precomputed π disambiguation provided by the JSOC data-processing center is applied using the `HMI_DISAMBIG` routine from the Solar Soft library. In the case of disambiguation artifacts, there is an option to perform π disambiguation with the Super Fast and Quality azimuth disambiguation library (SFQ; Rudenko & Anfinogentov 2014), also known as the new disambiguation method, which is supplied as a part of the AMPP. Although both methods, which may be interchanged by using the HMI/SFQ switch, work comparably well (Fleishman et al. 2017), in some cases, especially for near-limb observations, the SFQ library may provide better results than the standard HMI disambiguation (Rudenko & Anfinogentov 2014).

After the disambiguation, the vector magnetic field map is deprojected from the LOS coordinate system to the spherical components B_ϕ, B_θ, and B_r using the `HMI_B2PTR` procedure from the SDO/HMI package in Solar Soft. These components will become B_x, $-B_y$, and B_z components in the Cartesian coordinate system of the computational box.

At the next step, the deprojected magnetic field components are remapped to the local coordinate system of a computational box. Because the current version of AMPP uses Cartesian coordinates, the magnetic field maps are projected from the spherical surface of the Sun to the flat bottom of the box. The AMPP supports two projections: top view, which is a simple parallel projection; and cylindrical equal area (CEA) projection. The latter is the default option, and it is preferable for extrapolation purposes because it preserves the area of magnetic elements—and hence the magnetic flux is not changed by the projection effects. While performing coordinate transformation, AMPP relies on a WCS general-purpose library that is supplied by the Solar SoftWare (SSW) repository. To improve the quality of remapping, we use cubic interpolation when the requested resolution of the computational box is higher than or comparable to the pixel size of the available magnetograms. In the opposite case, when the spatial resolution of the computational box is lower than the resolution of a magnetogram, we use an oversampling antialiasing technique by dividing every computational pixel into eight subpixels. The resulting magnetic field components are then converted to the computational resolution by direct summation of the values interpolated to subpixels.

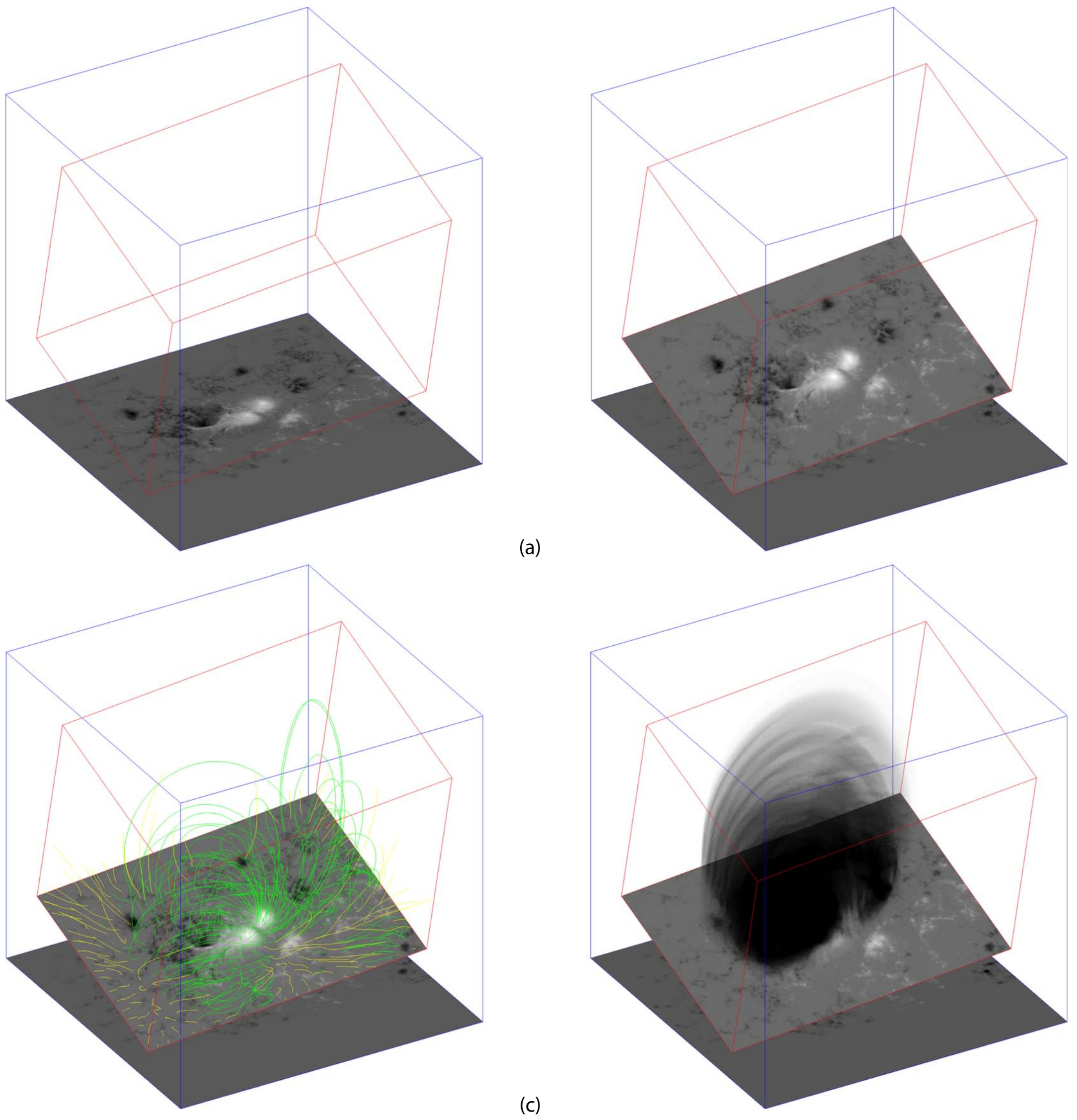


Figure 3. Snapshots of the AR11520 model generated by the script presented in Appendix A, illustrating different stages of the AMPP process. (a) The photospheric LOS magnetic field map, located at the bottom of a rectangular box coaligned with the observer's LOS (blue lines). The inscribed rectangular box (red lines), which is aligned with the direction normal to the solar surface, defines the 3D volume in which the magnetic field extrapolation is performed. (b) The elements in (a) plus B_z, obtained by projecting the photospheric vector magnetic field HMI map onto the bottom boundary. Two projection options are provided by AMPP: the default cylindrical area projection (CEA), or a simple parallel projection (TOP), selected by using the "Geometrical Projection" radio button shown in Figure 1. (c) A subset of model field lines that do, or do not, close within the 3D model boundaries (green and yellow lines, respectively), illustrating the magnetic connectivity in the model. (d) The coronal temperature distribution along the closed field lines, which corresponds to the parameterized magneto-thermal model defined by the AMPP script (refer to Section 3.5 for a detailed description). A user-specified hydrostatic model is used for the volume outside these closed field lines.

After remapping, the map of a deprojected magnetic field is placed in the computational box as a bottom layer, with the rest of the computational box consisting of zeroed arrays, ready to store the Cartesian components of the magnetic field model not yet generated. This geometrical structure, which may be optionally saved to disk as a file with "`.NONE.`" tag, is forwarded to the next stage of the AMPP.

3.2.2. Potential Field Initialization of the AMPP Model

During this stage of the AMPP process, the empty-box volume is filled with a potential field solution obtained from the normal component of the magnetic field at the lower boundary using the fast Fourier transform (FFT) method described in Alissandrakis (1981). The FFT solution for the potential field problem implies periodic, flux-balanced boundary conditions at lateral boundaries, which is not realistic. To simulate more appropriate "open" boundaries, we expand the computational domain ($L_{x,y}$) by $L_{x,y}/2$ in each direction. Then, the normal component of the field at the lower boundary is padded with a constant, generally nonzero value. This value is computed such that the total signed magnetic flux from the added areas perfectly compensates the unbalanced flux at the original lower boundary. The final potential field solution is then obtained by cutting out from the expanded domain, and it is then used as the initial condition for the NLFFF extrapolation.

3.2.3. NLFFF Reconstruction and Magnetic Field Line Tracing Dynamic Link Library

The NLFFF reconstruction code employed by AMPP was developed in C++ using multithreaded functionality. Its

Table 1
GX Simulator Filename Extension Naming Convention

Tag	Model Type
.NONE.	An empty-box IDL structure that contains all geometrical information and context SDO/AIA maps requested, as well as properly sized zeroed arrays ready to store the Cartesian components of the magnetic field model not yet generated, as described in Section 3.2.1.
.POT.	An IDL structure containing a true potential solution based on only the B_z base map component, as described in Section 3.2.2.
.BND.	An IDL structure containing potential solution except for the bottom layer, which is replaced by the observed B_x, B_y, and B_z components—to be used as initial conditions for the following NLFFF optimization step.
.NAS.	An IDL box structure filled with a nonlinear force-free magnetic field model (Stupishin 2020), as described in Section 3.2.3
.GEN.	An IDL box structure containing the length L and averaged magnetic field $\langle B\rangle$ along the field lines crossing a given volume voxel. These additional parameters are ready to be used for the purpose of adding a parameterized heated plasma coronal model that assumes either steady-state or impulsive plasma heating, as described in Section 3.5
.CHR.	An IDL box structure containing a set of additional tags used to define a nonuniform height, non-LTE density, and temperature distribution model of the chromosphere that is constrained by photospheric measurements (Fontenla et al. 2009), as described in Section 3.4.

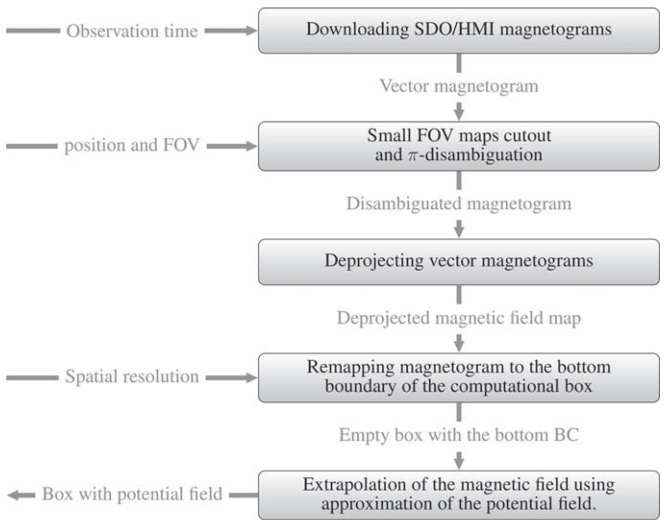

Observation time
Downloading SDO/HMI magnetograms
Vector magnetogram
position and FOV
Small FOV maps cutout and π-disambiguation
Disambiguated magnetogram
Deprojecting vector magnetograms
Deprojected magnetic field map
Spatial resolution
Remapping magnetogram to the bottom boundary of the computational box
Empty box with the bottom BC
Box with potential field
Extrapolation of the magnetic field using approximation of the potential field.

Figure 4. Flowchart of the production of initial and boundary conditions. Gray boxes represent individual steps of the pipeline, while intermediate data products are shown as arrows with labels.

source, the compilation scripts designed for Windows and Linux platforms, a set of compiled libraries for both platforms, and their calling IDL wrappers, are included in the GX Simulator SSW distribution package, and they are automatically updated from their independently maintained GitHub development repository.[9] In addition, the package may be directly downloaded from a Zenodo© digital repository (Stupishin 2020). Because the general platform compatibility of the precompiled Linux library is not guaranteed, the AMPP automatically invokes the distributed source code to compile and save a local copy[10] of the shared library on its first call on a Linux platform, which is used in all subsequent calls.

This NLFFF reconstruction code follows the development proposed by Wheatland et al. (2000) and Wiegelmann (2004). The basic idea is to reduce the Lorentz force in the coronal volume (i.e., to eliminate transverse electric currents) and reduce the field divergence as much as possible by minimization of the functional

$$L = \int_V [B^{-2}[[\nabla \times \boldsymbol{B}] \times \boldsymbol{B}]^2 + |\nabla \cdot \boldsymbol{B}|^2]w(x, y, z)dV, \quad (1)$$

where the first term, $(B^{-2}[[\nabla \times \boldsymbol{B}] \times \boldsymbol{B}]^2)$, represents the Lorentz force, and the second one, $(|\nabla \cdot \boldsymbol{B}|^2)$, evaluates the field divergence, while $w(x, y, z)$ is a "weight" function. The weight function is intended to diminish the influence of uncertainties of the field at the side and top boundaries, and it can be adjusted by the user. By default, the weights are constant (=1) in the entire volume except for 10 % of the length of each dimension on each side, where the weights decrease to zero on the boundaries following a cosine function (the bottom boundary is not weighted, because it is set to the observed photospheric field).

The initial state of the magnetic field may be inferred from several reasonable approaches. By default, the algorithm uses the preliminary potential field reconstruction described in Section 3.2.2. Alternatively, the NLFFF reconstruction may be started from an AMPP-compatible geometrical box prefilled with an initial magnetic field configuration obtained by any means, from which the boundary conditions are also inferred with or without buffer zones, as indicated by the user.

If one considers the magnetic field as a function of the conditional evolution parameter t, $\boldsymbol{B}(x, y, z, t)$, the evolution of the functional may be estimated by computing $\partial\boldsymbol{B}/\partial t$ at the ith step (L_i) and modifying $\boldsymbol{B}$ for the next $(i+1)$th step as $\boldsymbol{B}_{i+1} = \boldsymbol{B}_i + (\partial\boldsymbol{B}/\partial t)\Delta t$ (where Δt is a small evolution step), to get the next functional value L_{i+1}. The step size is varied so as to increase the iteration speed, being chosen automatically depending on the speed of convergence: it is increased by 10 % at a successful step and decreased by 10 % at an unsuccessful one.

Due to the numerical errors affecting the computation of the functional, the value of the functional may slightly increase at the next iteration even for a small step Δt, which is allowed by the algorithm up to a 10^{-4} relative factor. If the functional increases above this limit, the step is reduced by 10 %. The iterations stop when the step becomes too small, i.e., less than 1 % of the initial value. In addition, the iterations are terminated if (i) the relative variation of the function for the last 10 iterations is small (less than $5 \cdot 10^{-4}$), or (ii) if the maximum value of $|L_i/L_{i+1} - 1|$ does not exceed 10^{-4} during the previous 100 iterations. In such cases, no further significant decrease of the functional is expected, and it is assumed that the current solution is reasonably close to the optimal one.

Another approach used to decrease the computation time is the technique of "multigrids" (Metcalf et al. 2008). Instead of

[9] Magnetic-Field_Library on GitHub (https://github.com/Alexey-Stupishin/Magnetic-Field_Library).
[10] The user may invoke the IDL command line "print, gx_libpath("nlfff")" to retrieve the location of the shared library, or "print, gx_libpath("nlffff,"/update)" to also request a new compilation of the library, provided that a g++ compiler is installed on the system.

performing the computations directly using the desired volumetric grid resolution (e.g., 257 × 257 × 129), the initial potential field is first computed over a smaller resolution grid, let us say, 65 × 65 × 33, a first-stage NLFFF functional minimization is performed, and then the procedure is repeated twice, increasing the grid resolution at each step (to 129 × 129 × 65, and then finally to 257 × 257 × 129), while interpolating the solution obtained at each step to the next grid resolution and using it as the initial condition for the next step.

The same code may also be used to compute the magnetic field lines passing through each voxel of the volume, or through a set of predefined "seed voxels" (box coordinates). The lines are computed using the Runge–Kutta–Feldberg algorithm of fourth (to fifth) orders (the code was ported from original FORTRAN implementation (see Forsythe et al. 1977) and adapted to C++ using multithread functionality). When computation of a set of seeded lines is requested by the user, the code returns each line as a collection of fractional box indices indicating at least one intersection point for each volume element that is intersected. Such lines may be used for the purpose of visualizing the magnetic field connectivity, as well as for constructing flux tubes that may be used in flare studies, as described in Section 5.

3.3. AMPP Default Coronal Models

The computation of the magnetic field lines passing through each voxel of the volume is performed for dressing the magnetic field structure with a thermal plasma model (Nita et al. 2018; see Section 3.5 for more details). In this case, the code returns the following parameters associated with each voxel of the volume:

1. length of the field line intersecting the voxel,
2. average magnetic field along the line,
3. connectivity with the two boundary voxels associated with the line,
4. a flag indicating whether the voxel is intersected by a closed field line (both footpoints at the chromospheric layer are located inside the box) or by an open line (only one footpoint is located inside the box).

For a quantitative assessment of the computational speed, the reader may refer to the console messages generated when running the AMPP script presented in Appendix A, which was used to generate a 240 × 168 × 200 magnetic field cube for an instance of AR11520 on a Windows 10 system equipped with an eight-core 2.4GHz Intel Xeon E-2286M CPU and 64 GB RAM. In this particular case, the NLFFF reconstruction was performed in ∼250 s, and the computation of the lines intersecting all volume voxels was performed in ∼105 s. For a given size of the computational box, the computational time of an NLFFF reconstruction may vary as much as one order of magnitude, depending on the complexity of the magnetic field configuration (e.g., isolated sunspot versus a complex AR), while the speed of the full-volume line computation scales roughly linearly with the number of volume elements. More detailed benchmark tests performed for the purpose of assessing the code accuracy when compared with a ground-truth magnetic field model may be found in Fleishman et al. (2017), where the code is referred to as the AS NLFFF reconstruction code.

3.4. AMPP Default Chromosphere Models

The general approach employed by the AMPP to populate the chromospheric volume, described in detail in Nita et al. (2018), uses observationally established thresholds to distinguish seven quiet-Sun (QS) and AR features based on a corresponding HMI limb-darkening-removed white-light map and LOS magnetogram, and selects one of a set of seven corresponding 1D solar atmospheric models proposed by Fontenla et al. (2009) to fill the chromospheric volume above a particular chromospheric pixel. The seven feature types comprise three QS components, namely internetwork (IN), network lane (NW), and enhanced network (ENW), and four AR features: sunspot umbra (UBR), penumbra (PEN), plage (PL), and faculae (FA).

The chromospheric volume thus generated is then used to replace the bottom layers of the uniformly spaced magnetic skeleton with a composite slab having the minimum thickness needed to contain the variable height chromosphere, and any height-dependent properties of the original volume are interpolated and transferred to the nonuniform chromospheric voxels. The GX Simulator model structures that include such chromosphere models are by default stored on the disk with a file name that includes the ".CHR." tag (although GX Simulator does not rely on this naming convention to recognize the type of models produced by the pipeline).

However, one may choose to skip this step of the model production pipeline, and assign instead a chromosphere represented by a uniform slab of adjustable height, constant temperature T_{chr}, and constant density n_{chr}, interactively chosen through the GX Simulator GUI, this option being available for any of the POT, NAS, NAS.GEN, or POT.GEN models produced by the pipeline. This simpler option is often appropriate for modeling of flaring loops (Nita et al. 2015; Kuroda et al. 2018; Fleishman et al. 2018, 2021b).

Nevertheless, as indicated in Section 4.5 and detailed in Appendix C, the structured data architecture of a standardized GX Simulator model allows the experienced user to replace the default chromosphere model generated by AMPP with alternative models that may be used for, e.g., AR modeling (Selhorst et al. 2005, 2008) or flaring loop modeling (Machado et al. 1980; Trottet et al. 2015).

3.5. AMPP User-adjustable Coronal Models

The default background corona is populated with a simplistic, analytically defined, horizontally uniform hydrostatic equilibrium model (Nita et al. 2018). For practical data-constrained modeling, a more realistic field-aligned hydrodynamic model may be used to replace the background thermal plasma in voxels on closed loops. This model assumes a heating along the individual magnetic flux tubes defined by the extrapolated field line structure. The default hydrodynamic simulation code employed by GX Simulator is the Enthalpy-Based Thermal Evolution of Loops (EBTEL; Klimchuk et al. 2008; Cargill et al. 2012a, 2012b; Barnes et al. 2016; Bradshaw & Viall 2016; Ugarte-Urra et al. 2017), which assumes an impulsive heating and includes a link between the corona and lower atmosphere.

As illustrated in Figure 2 (see Section 3.3), the AMPP automatically generates models ready to be populated with EBTEL solutions by computing the average magnetic field $\langle B\rangle$ and length L of the magnetic field lines crossing each volume voxel. These parameters are used to compute the time-averaged volumetric heating rate, $\langle Q\rangle$, obtained from

$$Q(t) = Q_0\left(\frac{\langle B\rangle}{B_0}\right)^a\left(\frac{L_0}{L}\right)^b f(t), \tag{2}$$

and assign it to each voxel crossed by a closed field line. Here, the heating profile $f(t)$ incorporates the duration Δt of the nanoflares, as well as the time interval between successive events τ, and it may also include a dependence on mass density ρ. We have adopted the normalization convention $\langle f(t)\rangle \equiv 1$, which, for any heating model, ensures that the averaged heating rate, $\langle Q\rangle$, stays the same for a fixed choice of the Q_0, a, and b parameters.

The model includes five parameters that are independent of each other: Q_0, a, b, τ, and Δt. Q_0 represents the typical heating rate, which may vary depending on the driver velocity v and the electric current density along the flux tube, or the force-free parameter α. Thus, different flux tubes can have different values of Q_0. The actual value of Q_0 (expressed in units of erg cm^{-3} s^{-1}) depends on the two normalization constants that are chosen as $B_0 = 100$ G and $L_0 = 2\times10^9$ cm. The power-law indices a and b are fixed values that depend on the chosen heating model, such as the *critical shear angle model* (Mandrini et al. 2000), where $a = 2$ and $b = 1$.

The time constants, τ and Δt, are additional free parameters of the model, which are informed by analysis of the EUV AR lightcurves (Viall & Klimchuk 2012) and EBTEL modeling of these line-of-sight-integrated light curves (Viall & Klimchuk 2013). EBTEL can accurately simulate a wide range of τ values, ranging from "steady" to fully "impulsive" regimes (see Nita et al. (2018) for more details).

As detailed in Section 4.3, for a given set of input free parameters, the most recent version of the hydrodynamic simulation code, dubbed EBTEL++, outputs a pair of distributions over a relevant temperature range, which are the commonly used differential emission measure (DEM),

$$\xi(T) = \frac{n_e^2(T)dV}{VdT}, \quad [\mathrm{cm^{-6}\,K^{-1}}], \tag{3}$$

and the differential density metrics (DDM; Fleishman et al. 2021),

$$\nu(T) = \frac{n_e(T)dV}{VdT}, \quad [\mathrm{cm^{-3}\,K^{-1}}]. \tag{4}$$

When only the DEM distributions are available, as is the case of the output provided by the original EBTEL code, or if explicitly requested by the user, GX Simulator uses them to assign effective density and temperature pairs to any model voxel crossed by a closed magnetic field line characterized by a $\{\langle B\rangle, L\}$ pair:

$$n_\xi \equiv \sqrt{\langle n_e^2\rangle} = \left(\int \xi(T)dT\right)^{1/2},$$
$$T_\xi = \frac{\int T\cdot\xi(T)dT}{{n_\xi}^2}. \tag{5}$$

However, if the DDM distributions defined by Equation (4) are also available, as is the case for the output provided by the upgraded EBTEL++ code, GX Simulator computes, by default, the effective density–temperature pairs defined as

$$n_\nu = \int \nu(T)dT,$$
$$T_\nu = \frac{\int T\cdot\nu(T)dT}{n_\nu}. \tag{6}$$

Given the fact that a typical GX Simulator model contains a large number of coronal voxels that need to be populated at runtime with EBTEL solutions, the practical approach that has been adopted is to run the EBTEL code offline, to precompute several thousand combinations of the flux tube lengths, L, and nanoflare magnitudes (heating-model-specific time-averaged volumetric heating rates), $\langle Q\rangle$, to create lookup tables that contain the coronal and transition region DEM and DDM distributions for each pair of flux tube length and nanoflare magnitude. Thus, using the $\langle B\rangle$ and L properties computed by the pipeline for each coronal or transition region voxel, the adjustable Equation (2) is used at runtime to select the corresponding nanoflare magnitudes and assign the DEM and DDM distributions from precomputed lookup tables to a given voxel, an approach that has been tested and validated by Nita et al. (2018). By default, GX Simulator assigns the DEM-DDM pair corresponding to the closest $\{\langle Q\rangle, L\}$ neighbor grid node found in the lookup tables, but the GUI provides a series of alternative irregular grid interpolation methods from which to choose, including four-closest-neighbor weighted interpolation.

The DEM distributions are used by the GX Simulator EUV radiation transfer codes to compute synthetic emission maps corresponding to the SDO/AIA channels. The effective thermal plasma density and temperature pairs inferred from the DDM or DEM distributions are used by the GX Simulator GUI for volume visualization purposes, and by the legacy Fast Codes microwave rendering routines to compute synthetic multifrequency radio emission. However, following the recent upgrade of the microwave Fast Codes (Fleishman et al. 2021), GX Simulator offers the option to select a rendering routine that employs these codes to compute synthesized microwave emission maps directly from the DEM/DDM distributions, as detailed in Section 4.4.

4. Customization of Active Region Pipeline Models

4.1. Command Line Customization Scripts

The current release of the GX Simulator package includes a series of macro commands that allow batch mode customization of the pipeline models and generation of the multiwavelength synthetic maps, as well as a series of benchmark tools that may be used to perform quantitative model-to-data spectral and image comparison for the purpose of model validation, as described in this section. To illustrate how some of the macro commands included in the GX Simulator package may be used to customize an AR model generated by AMPP and synthesize microwave emission maps corresponding to a user's selected field of view (FOV), we list in Appendix B an IDL script that performs the following actions:

1. imports a magneto-thermal structure prepared by the AMPP script provided in Appendix A;
2. defines the desired FOV and spatial resolution for producing the synthesized maps;
3. defines a set of volumetric heating rate parameters;

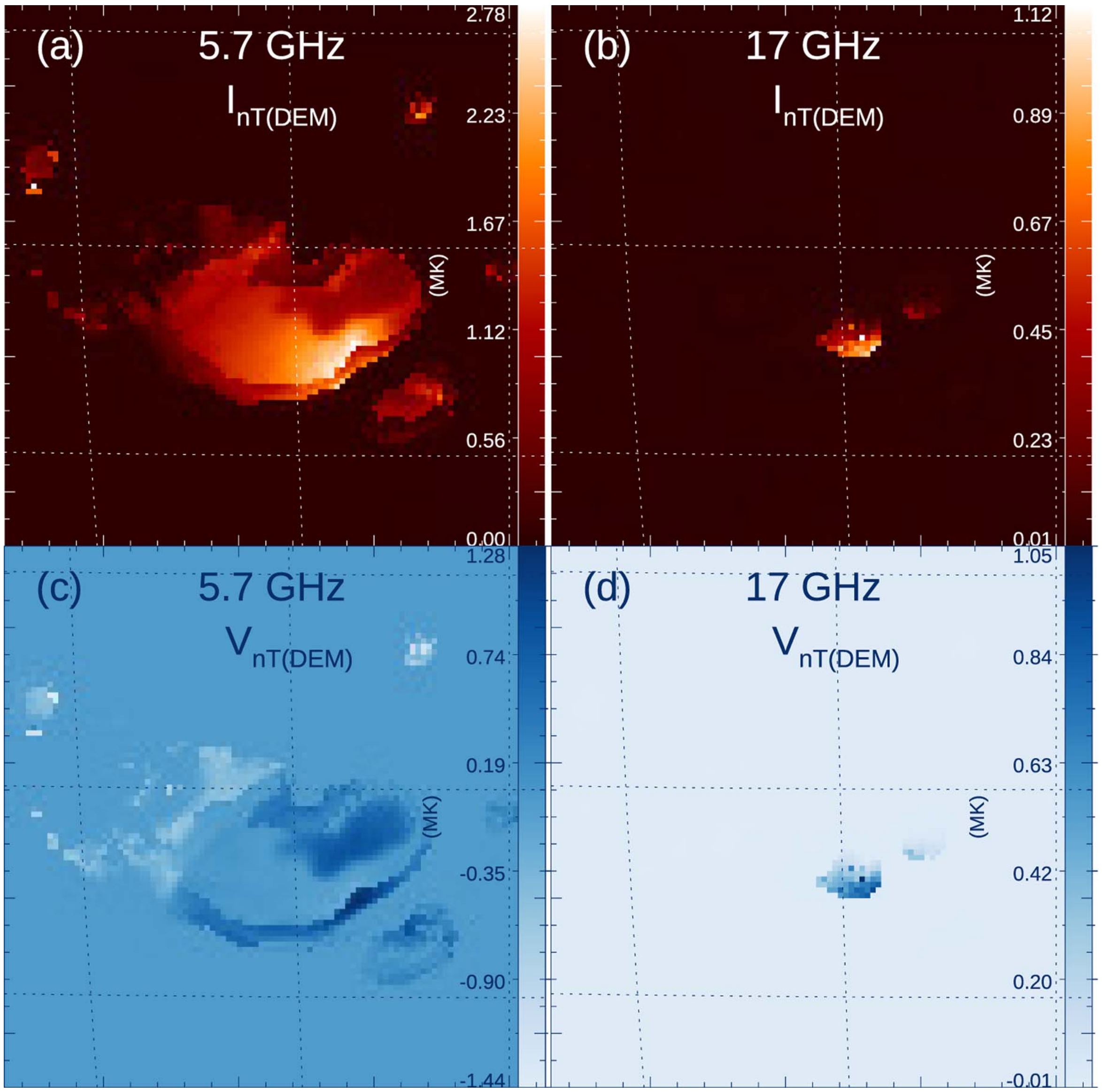


Figure 5. Brightness temperature Stokes *I* (top row) and *V* (bottom row) at 5.7 GHz (left column) and 17 GHz (right column) produced by the IDL script listed in Appendix B.

4. defines a heating rate formula following Equation (2), which takes into account the user-defined input parameters and the AMPP precomputed $\langle B\rangle$ and L voxel properties;
5. selects a specific EBTEL table;
6. defines a set of frequencies for which the synthetic microwave maps will be computed;
7. calls a microwave rendering module that performs the geometrical rendering of the 3D models and solves the radiation transfer equation along each image LOS to produce the set of requested microwave maps;
8. saves on disk the output data produced by this script in two alternative forms, namely an SSW-compatible IDL map object and a GX-specific IDL structure that contains both image data, as well as metadata documenting the entire process involved in generating the script output.

The Stokes *I* and *V* brightness temperature maps generated by this script are illustrated in Figure 5. When combined with the data-to-model comparison tools described in Section 4.2, the script presented in Appendix B may be employed to perform a fully automatic systematic search in a multidimensional parameter space for the combination of such magneto-thermal model parameters that produce synthesized maps that are simultaneously in best agreement with all multiwavelength observational data available for a given instance of an AR, a methodology successfully employed by Fleishman et al. (2021a) for finding the best scaling parameters within the low-frequency heating assumption for the same instance of AR 11520 illustrated here.

4.2. Model-to-data Comparison Tools

The current release of GX_Simulator provides a series of data-to-model comparison routines, located in the `metrics` submodule, that are integrated in the GUI interface but may be also called programmatically by customized analysis IDL scripts. The top-level routine included in this submodule, `gx_metrics_map.pro`, takes, as the input arguments, a model map structure and a reference map structure to be compared with, and returns a structure containing spatially resolved and FOV-averaged metrics such as absolute and normalized residuals, as well as χ^2 metrics, if a map of standard deviations representing the observational or model uncertainties is provided as optional input.

By default, before computing the data-to-model metrics, this top-level routine also performs a map alignment by computing a cross-correlation of the input map images, to which an optional mask may be applied. However, the user may choose not to align the input maps, or to apply a user-supplied spatial shift.

The FOV-averaged absolute and normalized residuals metrics returned by the top-level routine are defined as follows:

$$\begin{aligned}\langle\mathcal{R}\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}(m_{ij}\otimes d_{\mathrm{PSF}} - d_{ij}),\\ \langle\rho\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}\left(\frac{m_{ij}\otimes d_{\mathrm{PSF}}}{d_{ij}} - 1\right),\\ \langle\chi\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}\left(\frac{m_{ij}\otimes d_{\mathrm{PSF}} - d_{ij}}{\sigma_{ij}}\right),\end{aligned} \quad (7)$$

where d_{ij} is the observed brightness in the image pixel $_{ij}$ and $m_{ij}\otimes d_{PFS}$ is the corresponding model map brightness convolved with the instrumental point-spread function (PSF), the model or observational uncertainties are denoted by σ_{ij}, and N represents the total number of pixels in the region of interest (ROI) over which the comparison is made (which may be all or only a subset of the FOV pixels selected by applying an optional, user-defined byte mask).

The corresponding squared metrics are defined as follows:

$$\begin{aligned}\langle\mathcal{R}^2\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}(m_{ij}\otimes d_{\mathrm{PSF}} - d_{ij})^2.\\ \langle\rho^2\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}\left(\frac{m_{ij}\otimes d_{\mathrm{PSF}}}{d_{ij}} - 1\right)^2,\end{aligned} \quad (8)$$

and the metrics of success for the data-to-model comparison are defined as

$$\begin{aligned}\sigma_\rho^2 &= \frac{1}{N}\sum_{\mathrm{ROI}}\left(\frac{m_{ij}\otimes d_{\mathrm{PSF}}}{d_{ij}} - 1 - \langle\rho\rangle\right)^2 = \langle\rho^2\rangle - \langle\rho\rangle^2\\ \sigma_{\mathcal{R}}^2 &= \frac{1}{N}\sum_{\mathrm{ROI}}(m_{ij}\otimes d_{\mathrm{PSF}} - d_{ij} - \langle\mathcal{R}\rangle)^2 = \langle\mathcal{R}^2\rangle - \langle\mathcal{R}\rangle^2,\\ \langle\chi^2\rangle &= \frac{1}{N}\sum_{\mathrm{ROI}}\left(\frac{m_{ij}\otimes d_{\mathrm{PSF}} - d_{ij}}{\sigma_i}\right)^2 - \langle\chi\rangle^2,\end{aligned} \quad (9)$$

where, as motivated by Fleishman et al. (2021a), the last term of each of the metrics of success defined above is subtracted to account for any imperfections in fine-tuning the model, which may result in minimized but not exactly null averaged residuals that, as defined by Equation (7), are expected to vanish in the case of a perfect data-to-model match. As an example of data-to-model comparison performed using the GX Simulator metrics routine, we reproduce in Figure 6 the results obtained by Fleishman et al. (2021a) using observational maps obtained with data from the Siberian Solar Radio Telescope (SSRT; Grechnev et al. 2003) and the Nobeyama Radio Heliograph (NoRH; Nakajima et al. 1994) and the 5.7 and 17 GHz synthetic images shown in Figures 5(a) and (b), which were convolved with the corresponding instrumental beams before their coalignment with the observational maps.

4.2.1. Coronal Heating Modeling Pipeline (CHMP)

To identify the combination of $\{a, b\}$ and Q_0 parameters that provides the best possible model-to-data agreement, quantitatively measured by the metrics described in Section 4.2, we have included in the most recent release of the GX Simulator package a macro routine, namely `gx_search4bestq.pro`, which, starting from a pair of initial guess heating rates, $\{Q_{0_1}, Q_{0_2}\}$, performs a self-adaptive search for the optimal EBTEL heating rate Q_0 corresponding to a predefined $\{a, b\}$ pair chosen by the user.

The `gx_search4bestq.pro` macro routine provides the core functionality for a top-level command line application, namely `chmp.pro`, which allows the user to interactively set up and start a multithreaded search for the best possible EBTEL model over the $\{a, b\}$ parameter grid, which takes advantage of the fact that such a search is an embarrassingly parallel process.

The options provided by the CHMP command line application may be explored by calling the routine with no keyword arguments, which generates the following console messages:

```
IDL> chmp
% CHMP_SELF: CHMP has been initialized!
% IDL-> chmp, nthreads; to set,increase, or
decrease
            the number of ashyncronious
threads to be used
% IDL-> chmp, /alist; to print the current
a-parameter list
% IDL-> chmp, /blist; to print the current
b-parameter list
% IDL-> chmp, /levels; to print the current
ROI levels list
% IDL-> chmp, /fov; to print the current FOV
settings
% IDL-> chmp, /res; to print the current map
resolution settings
% IDL-> chmp, /refdatapath;
            to print the current reference
data path
% IDL-> chmp, /gxmpath; to print the current
GX model data path
% IDL-> chmp, /bridges; to print the current
status
            of the parallel execution
threads
% IDL-> chmp, /status; to report the status
            of the application, including
all of the above
% IDL-> chmp, /start; to start processing
the task queue
% IDL-> chmp, /flush; to flush the pending
task queue
% IDL-> chmp, /abort; to abort all active
tasks
            and flush the pending task queue
```

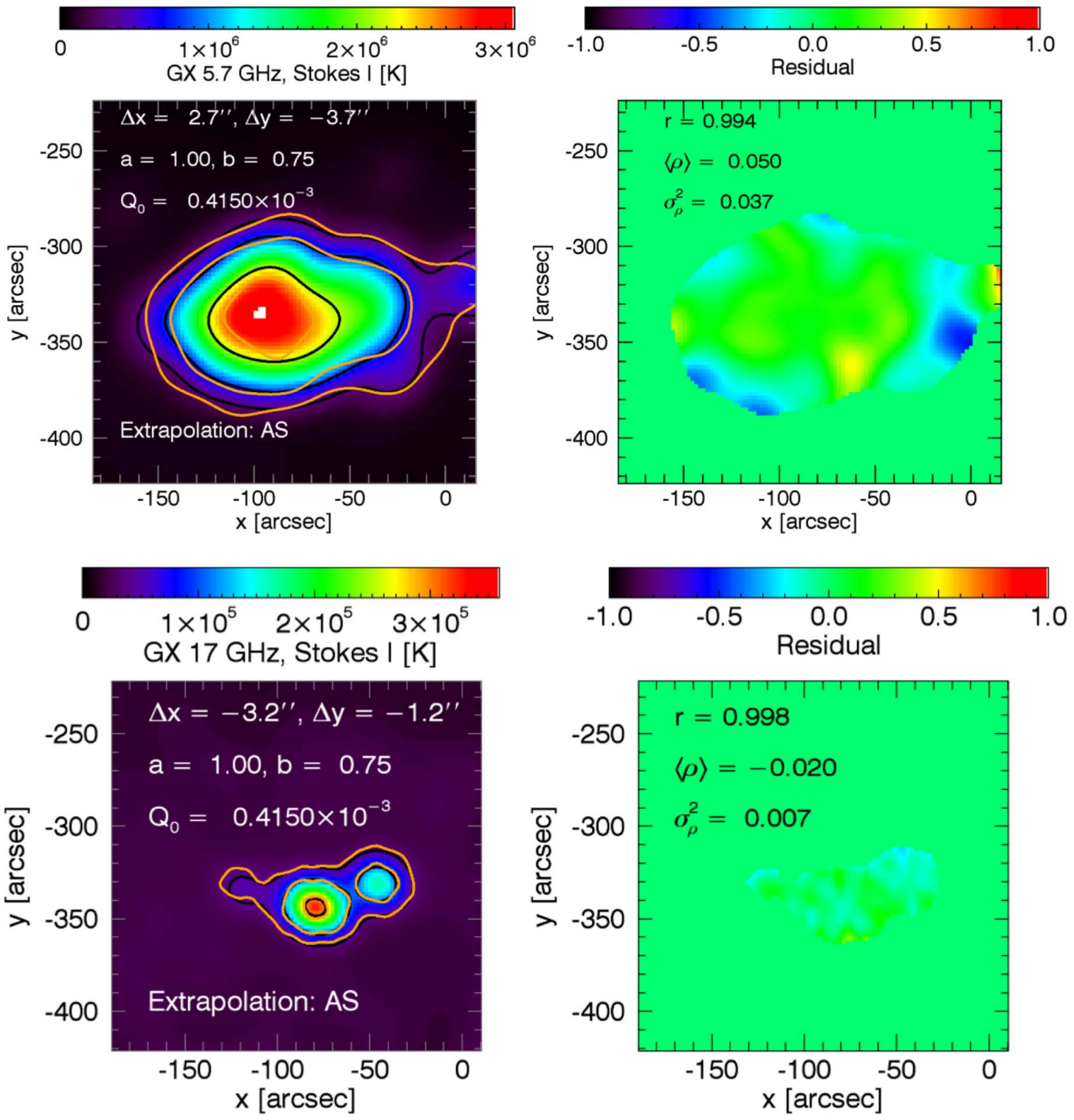


Figure 6. Model-to-data comparison for the AR11520 model (reproduced from Fleishman et al. 2021a). Top left panel: The [12, 30, 80]% contours of the 5.7 GHz SSRT observational map (yellow) and the synthetic map (black) are shown on top of synthetic 5.7 GHz map background image. The alignment shifts, Δx and Δy, and the heating parameters, Q_0, a, and b, are indicated in the figure inset. Top right panel: corresponding model-to-data residual map normalized by the observed SSRT brightness temperature. Pearson cross-correlation coefficient, r, the averaged normalized residual, ρ, and the squared residual metrics, ρ^2, are indicated in the figure inset. Bottom row: the same data-to-model comparison as in the top row, between the 17 GHz synthetic image and the NoRH 17 GHz image.

```
% IDL-> chmp, /quiet; to turn off
                 run-time execution progress
messages
% IDL-> chmp, /loud; to turn on run-time
                 execution progress messages
% IDL-> chmp, /exit; to abort all active
tasks,
                 flush the pending task queue,
                 and exit the application
% CHMP_HELP: % Any logical combination of
the arguments
                 and keywords listed above should
result
                 in a valid single-line calling
sequence
```

The left panel of Figure 7 displays a flowchart that illustrates the iterative search process of the CHMP application. As an illustrative example, the right panel of Figure 7 displays the distribution of the best $\langle\chi^2\rangle$ metrics over the searched parameter space that has been obtained for the same AR11520 GX Simulator model that was previously tuned by Fleishman et al. (2021a) using a direct trial-and-error approach, which sought to minimize the σ_ρ^2 metrics using reference observational microwave data at 17 GHz provided by NoRH.

The optimal parameter combination found by the automated CHMP application in this illustrative example, ($a = 0.4$, $b = 1.4$, $\langle\chi\rangle^2 = 6.70$), is marked as a red square on the metrics image shown in the right panel of Figure 7. It is different from the brute-force solution found by Fleishman et al. (2021a), ($a = 1.0$, $b = 0.75$, $\langle\chi\rangle^2 = 13.98$), marked as a cyan rectangle

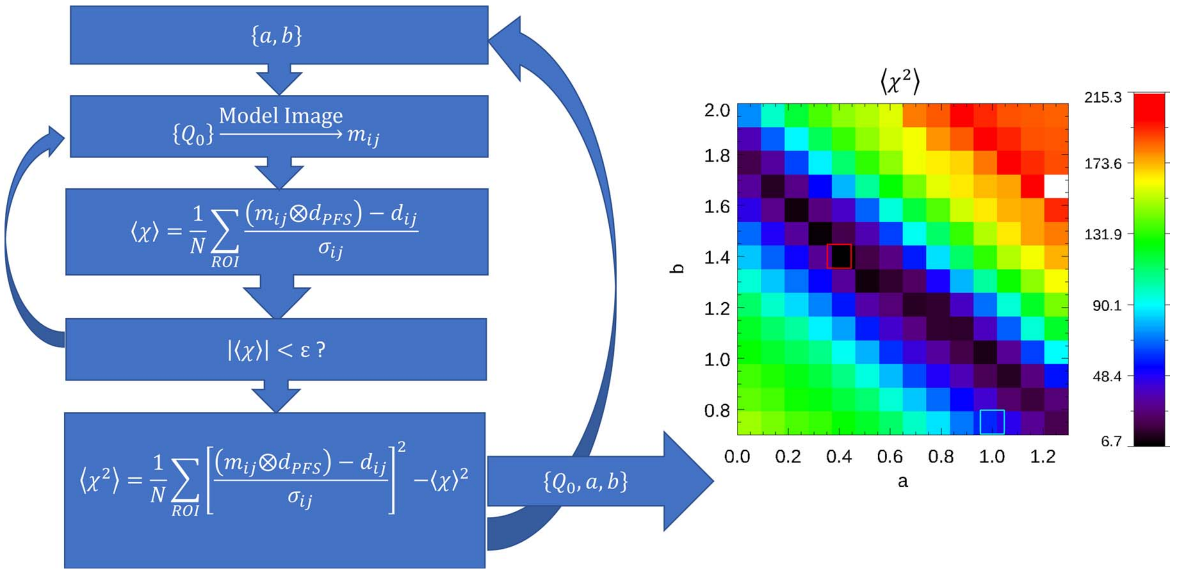

{a, b}
Model Image
$\{Q_0\} \longrightarrow m_{ij}$
$\langle\chi\rangle = \frac{1}{N}\sum_{ROI} \frac{(m_{ij} \otimes d_{PFS}) - d_{ij}}{\sigma_{ij}}$
$|\langle\chi\rangle| < \varepsilon$?
$\langle\chi^2\rangle = \frac{1}{N}\sum_{ROI}\left[\frac{(m_{ij} \otimes d_{PFS}) - d_{ij}}{\sigma_{ij}}\right]^2 - \langle\chi\rangle^2$
$\{Q_0, a, b\}$
$\langle\chi^2\rangle$
b
2.0
1.8
1.6
1.4
1.2
1.0
0.8
a
0.0
0.2
0.4
0.6
0.8
1.0
1.2
215.3
173.6
131.9
90.1
48.4
6.7

Figure 7. The automated CHMP architecture and illustrative example. Left panel: The CHMP iterative flowchart. The outer loop steps through a user-defined grid of $\{a, b\}$ parameter pairs for which the inner loop employs the `gx_search4bestq.pro` core macro to search for the optimal heating rate Q_0 by attempting to bring the absolute value of $\langle\chi^2\rangle$ metrics below a predefined maximum threshold ϵ. Right panel: The output of the automated CHMP search using NoRH 17 GHz observational data for the same AR11520 GX Simulator model as previously tuned by Fleishman et al. (2021a). The image displays, in logarithmic scale, the $\langle\chi^2\rangle$ corresponding to each investigated grid point. As indicated by the right-side color bar, the $\langle\chi^2\rangle$ metrics range in this case between 6.70 and 215.3. The red rectangle marks the grid point corresponding to the best found metrics ($a = 0.4$, $b = 1.4$; $\langle\chi\rangle^2 = 6.70$), which may be compared with the $\langle\chi^2\rangle = 13.98$ metrics corresponding to the best parameter combination, ($a = 1.0$, $b = 0.75$), found by Fleishman et al. (2021a) by minimizing the σ^2 metrics.

on the grid, although the data-to-model comparison metrics are of the same order of magnitude. As revealed by the $\langle\chi^2\rangle$ metrics distribution image, both solutions belong to the same region of comparatively good metrics, which stand out as a diagonal path against the surrounding parameter space. We interpret this preliminary result as an indication of a certain degree of degeneracy of the EBTEL solution, which we speculate might be possible to remove by combining the results of a CHMP search independently performed at more than one observational frequency, an avenue that we consider worth being pursued, but out of the scope of this paper.

4.2.2. CHMP Graphical User Interface (GX_CHMP)

In addition to the CHMP command line application, the core `gx_search4bestq.pro` routine includes many more built-in options that provide more user flexibility, including the option of performing a search over an arbitrarily shaped, or sparse, grid space. The full flexibility of the `gx_search4bestq.pro` routine is exposed to the user by the `gx_chmp.pro` application. Figure 8 shows this GUI application, which allows one to take advantage of all available options provided by the core macro routine. It also interactively visualizes, at runtime, the data-to-model comparison metrics maps corresponding to each grid point for which a solution has been computed.

4.3. DEM and DDM EBTEL/EBTEL++ Tables

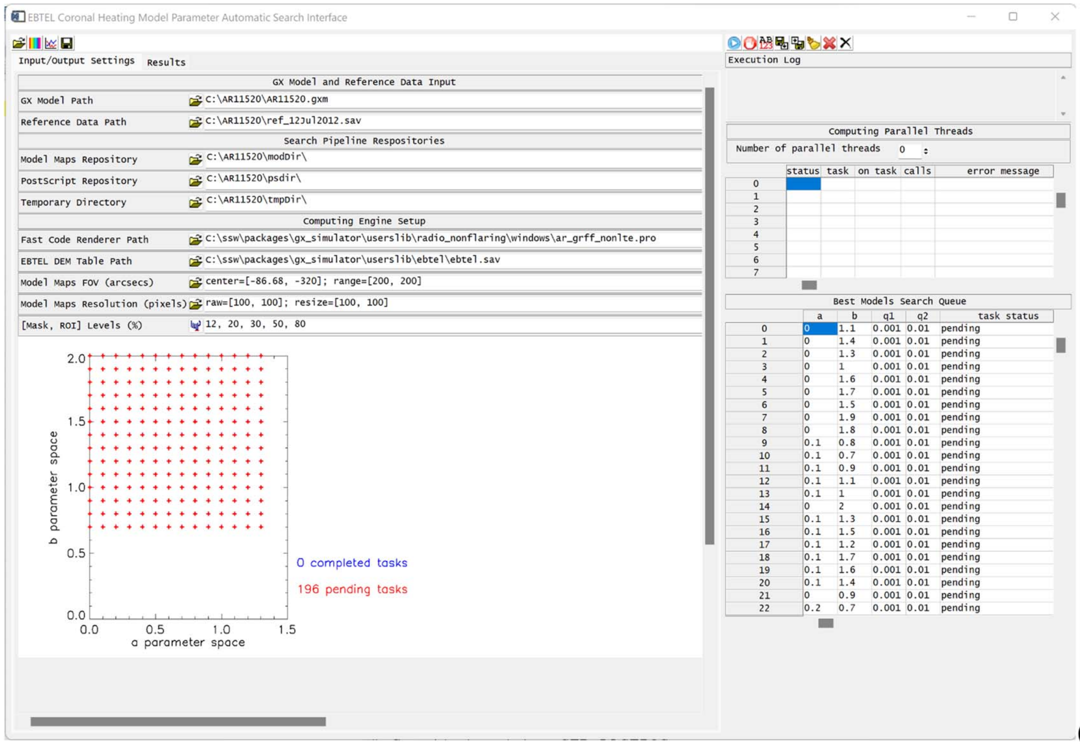
GX Simulator includes EBTEL++ results from a number of different coronal heating scenarios. Each is provided in the form of an IDL `sav` file containing the following floating point array variables:

1. LOGTDEM[N_T]: logarithm of temperature bins over which the DEM/DDM distributions are computed.
2. QRUN [$N_Q \times N_L$]: the average heating rates, $\langle Q\rangle$ corresponding to each grid point.
3. LRUN [$N_Q \times N_L$]: the loop half-lengths, $L_{1/2} \equiv L/2$, corresponding to each grid point.
4. DEM_COR_RUN[$N_T \times N_Q \times N_L$]: DEM distributions for coronal voxels.
5. DEM_TR_RUN[$N_T \times N_Q \times N_L$]: DEM distributions for transition region voxels.
6. DDM_COR_RUN[$N_T \times N_Q \times N_L$]: DDM distributions for coronal voxels.
7. DDM_TR_RUN[$N_T \times N_Q \times N_L$]: DDM distributions for transition region voxels.
8. TRUN [$N_Q \times N_L$]: maximum electron temperature achieved at any point during the associated EBTEL run (only included for informational purposes, and not used by GX Simulator).

where N_T denotes the number of temperature bins in the DEM/DDM distributions and $N_Q \times N_L$ represents the dimension of the parameter grid space over which the distributions have bin computed.

The same as the previous versions of GX Simulator, the current version includes two EBTEL tables representing impulsive and steady heating. The impulsive heating table was generated using triangular nanoflare profiles with $\Delta t = 20$ s and $\tau = 10{,}000$ s, while the steady heating table was generated with constant heating. These two tables cover the same $\sim 10^6$ cm $\leqslant L_{1/2} \leqslant \sim 4 \times 10^{10}$ cm range, but have different time-averaged ranges of heating rate, $\langle Q\rangle$, with the steady heating encompassing $\sim 3 \times 10^{-9}$ erg cm^{-3} s$^{-1} \leqslant \langle Q\rangle \leqslant \sim 2 \times 10^6$ erg cm^{-3} s^{-1} and the impulsive heating covering $\sim 6 \times 10^{-10}$ erg cm^{-3} s$^{-1} \leqslant \langle Q\rangle \leqslant \sim 0.2$ erg cm^{-3} s^{-1} on different irregular grids.

In addition to these idealized heating scenarios, the current version of GX Simulator also includes six new tables designed to emulate more physically realistic coronal heating



GX Model and Reference Data Input

GX Model Path	C:\AR11520\AR11520.gxm
Reference Data Path	C:\AR11520\ref_12Jul2012.sav

Search Pipeline Respositories

Model Maps Repository	C:\AR11520\modDir\
PostScript Repository	C:\AR11520\psdir\
Temporary Directory	C:\AR11520\tmpDir\

Computing Engine Setup

Fast Code Renderer Path	C:\ssw\packages\gx_simulator\userslib\radio_nonflaring\windows\ar_grff_nonlte.pro
EBTEL DEM Table Path	C:\ssw\packages\gx_simulator\userslib\ebtel\ebtel.sav
Model Maps FOV (arcsecs)	center=[-86.68, -320]; range=[200, 200]
Model Maps Resolution (pixels)	raw=[100, 100]; resize=[100, 100]
[Mask, ROI] Levels (%)	12, 20, 30, 50, 80

0 completed tasks

196 pending tasks

Computing Parallel Threads

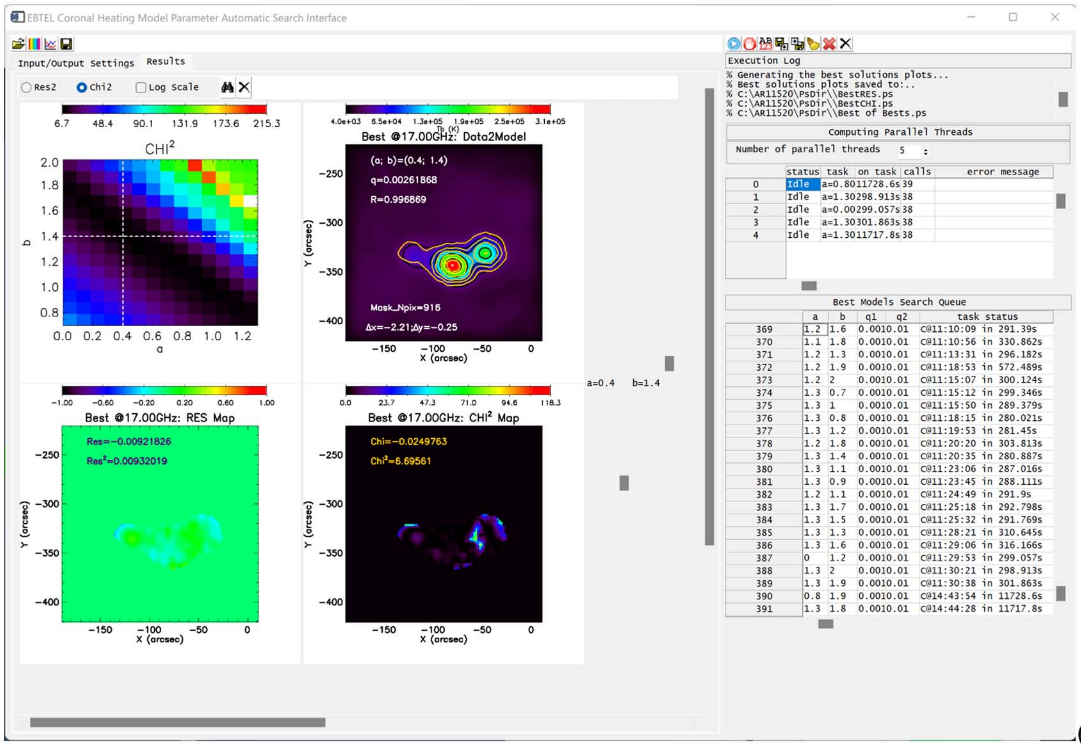
Number of parallel threads 0

	status	task	on task	calls	error message
0					
1					
2					
3					
4					
5					
6					
7					

Best Models Search Queue

	a	b	q1	q2	task status
0	0	1.1	0.001	0.01	pending
1	0	1.4	0.001	0.01	pending
2	0	1.3	0.001	0.01	pending
3	0	1	0.001	0.01	pending
4	0	1.6	0.001	0.01	pending
5	0	1.7	0.001	0.01	pending
6	0	1.5	0.001	0.01	pending
7	0	1.9	0.001	0.01	pending
8	0	1.8	0.001	0.01	pending
9	0.1	0.8	0.001	0.01	pending
10	0.1	0.7	0.001	0.01	pending
11	0.1	0.9	0.001	0.01	pending
12	0.1	1.1	0.001	0.01	pending
13	0.1	1	0.001	0.01	pending
14	0	2	0.001	0.01	pending
15	0.1	1.3	0.001	0.01	pending
16	0.1	1.5	0.001	0.01	pending
17	0.1	1.2	0.001	0.01	pending
18	0.1	1.7	0.001	0.01	pending
19	0.1	1.6	0.001	0.01	pending
20	0.1	1.4	0.001	0.01	pending
21	0	0.9	0.001	0.01	pending
22	0.2	0.7	0.001	0.01	pending

(a)



Res2 [✓ Chi2] Log Scale

```
% Generating the best solutions plots...
% Best solutions plots saved to:..
% C:\AR11520\PsDir\\BestRES.ps
% C:\AR11520\PsDir\\BestCHI.ps
% C:\AR11520\PsDir\\Best of Bests.ps
```

Computing Parallel Threads

Number of parallel threads 5

	status	task	on task	calls	error message
0	Idle	a=0.80	11728.6s	39	
1	Idle	a=1.30	298.913s	38	
2	Idle	a=0.00	299.057s	38	
3	Idle	a=1.30	301.863s	38	
4	Idle	a=1.30	11717.8s	38	

Best Models Search Queue

	a	b	q1	q2	task status
369	1.2	1.6	0.001	0.01	C@11:10:09 in 291.39s
370	1.1	1.8	0.001	0.01	C@11:10:56 in 330.862s
371	1.2	1.3	0.001	0.01	C@11:13:31 in 296.182s
372	1.2	1.9	0.001	0.01	C@11:18:53 in 572.489s
373	1.2	2	0.001	0.01	C@11:15:07 in 300.124s
374	1.3	0.7	0.001	0.01	C@11:15:12 in 299.346s
375	1.3	1	0.001	0.01	C@11:15:50 in 289.379s
376	1.3	0.8	0.001	0.01	C@11:18:15 in 280.021s
377	1.3	1.2	0.001	0.01	C@11:19:53 in 281.45s
378	1.2	1.8	0.001	0.01	C@11:20:20 in 303.813s
379	1.3	1.4	0.001	0.01	C@11:20:35 in 280.887s
380	1.3	1.1	0.001	0.01	C@11:23:06 in 287.016s
381	1.3	0.9	0.001	0.01	C@11:23:45 in 288.111s
382	1.2	1.1	0.001	0.01	C@11:24:49 in 291.9s
383	1.3	1.7	0.001	0.01	C@11:25:18 in 292.798s
384	1.3	1.5	0.001	0.01	C@11:25:32 in 291.769s
385	1.3	1.3	0.001	0.01	C@11:28:21 in 310.645s
386	1.3	1.6	0.001	0.01	C@11:29:06 in 316.166s
387	0	1.2	0.001	0.01	C@11:29:53 in 299.057s
388	1.3	2	0.001	0.01	C@11:30:21 in 298.913s
389	1.3	1.9	0.001	0.01	C@11:30:38 in 301.863s
390	0.8	1.9	0.001	0.01	C@14:43:54 in 11728.6s
391	1.3	1.8	0.001	0.01	C@14:44:28 in 11717.8s

(b)

Figure 8. GX Simulator CHMP Graphical User Interface (GX_CHMP). Panel (a): Search setup input fields tab. Panel (b): Output solution display area tab. Top left plot: Best $\langle\chi^2\rangle$ (or σ_ρ^2) metrics corresponding to each grid point. The intersection of dotted vertical and horizontal lines indicates a grid point selected by the user, which, in this case, corresponds to the best metrics found. Top right plot: model-convolved map (background) and model (black) and data (yellow) user-defined contours delimiting the ROI area. Bottom right plot: map of the σ_ρ^2 metrics corresponding to the selected grid point in the top left panel. Bottom left plot: map of the χ^2 metrics corresponding the selected grid point in the top left pane. The regions outside the ROI area are set to neutral uniform values in both metrics maps.

conditions. Each of the EBTEL++ models (Barnes et al. 2016; code available on GitHub[11]) used to generate these new tables includes a range of heating event sizes and time delays between successive events combined with steady background heating. This produces stochastic heating more representative of the true conditions on the Sun. These grids cover a standard parameter space of $10^6\,\mathrm{cm} \leqslant L_{1/2} \leqslant \sim 6\times10^{10}\,\mathrm{cm}$ and $10^3\,\mathrm{erg\,cm^{-2}\,s^{-1}} \leqslant \langle Q\rangle L_{1/2} \leqslant \sim 8\times 10^8\,\mathrm{erg\,cm^{-2}\,s^{-1}}$, sampled with a resolution of 0.1 dex for a

[11] https://github.com/rice-solar-physics/ebtelPlusPlus

Table 2
EBTEL Coronal Heating Tables Included in the GX Simulator Distribution Package

Filename	$\langle\tau\rangle$	α
ebtel.sav	$\tau = 10^4$s	N/A
ebtel_ss.sav	constant	N/A
ebtel_scale=0.2_alpha=-1.sav	$0.2\tau_0$	-1
ebtel_scale=0.2_alpha=-2.5.sav	$0.2\tau_0$	-2.5
ebtel_scale=1_alpha=-1.sav	τ_0	-1
ebtel_scale=1_alpha=-2.5.sav	τ_0	-2.5
ebtel_scale=5_alpha=-1.sav	$5\tau_0$	-1
ebtel_scale=5_alpha=-2.5.sav	$5\tau_0$	-2.5

total of 2940 models. This construction of the heating rate ensures that all models encompass observed heat flux ranges (Withbroe & Noyes 1977) and is largely consistent with the parameter spaces covered by the original two tables.

In these new EBTEL++ tables, the steady heating term is chosen to sustain loops with apex temperatures of ∼0.3 MK, well below typical coronal temperatures. This is achieved by applying a steady background heating of $6.3 \times 10^{12}/L_{1/2}^2 \geqslant 10^{-7}$ erg cm^{-3} s^{-1}, where the minimum threshold only impacts the longest loops, which can become unstable without it. In cases where this background heating term equals or exceeds $\langle Q\rangle$, it is capped at $\langle Q\rangle$, resulting in cooler loops with no impulsive heating. As a consequence of this steady heating term, a significant fraction of the models in the new EBTEL++ tables (those with lower heat flux, in particular for the shortest and longest loops) undergo identical steady heating in each table. However, because this occurs in the least physically realistic corners of the parameter space, it should have only a minor impact on the resulting GX Simulator models.

The impulsive heating in the new EBTEL++ tables is their only differentiator. Each heating event is a triangular heat pulse with a $\Delta t = 100$ s duration (for a discussion of why longer-duration nanoflares may produce more realistic results, see Barnes et al. 2016) with an amplitude drawn from a power-law distribution of event sizes. The amplitude of each heating event is correlated with the delay until the next heating event, with a proportionality (accounting for the steady background heating) that enforces $\langle Q\rangle$ from the start of the heating event until the start of the next event. In practice, these amplitudes are computed from the time delay that is drawn randomly from a power-law probability distribution defined by the slope α, the median time between heating events $\langle\tau\rangle$, a maximum time between events chosen as $3\langle\tau\rangle$, and the automatically determined minimum delay. The median time between heating events is defined as some scaling multiple of the cooling time τ_0, which depends on $\langle Q\rangle$ and $L_{1/2}$, (Cargill 2014; Barnes et al. 2019) and the ratio of the longest delay (corresponding to the strongest event) to $\langle\tau\rangle$, which in this case is 3. The six new tables included in GX Simulator are computed with each combination of $\alpha = -1$ (large-) or -2.5 (small-event-dominated) and $\langle\tau\rangle = 0.2\tau_0$ (high-), $1\tau_0$ (intermediate-), or $5\tau_0$ (low-frequency heating). All the EBTEL/EBTEL++ tables distributed with the current version of GX Simulator, along with their $\langle\tau\rangle$ and α parameters, are listed in Table 2.

For each $\{\langle Q\rangle, L_{1/2}\}$ pair in the new EBTEL++ tables, the model is run for $1000\langle\tau\rangle$ and the heating events are drawn using the same random seed, i.e., the heating events are identical except for changes in the $\langle\tau\rangle$ and $\langle Q\rangle$ scaling. The coronal and transition region DEMs and DDMs for the model are generated by averaging the time-evolving DEMs and DDMs over the entire model run, excluding the first $10\langle\tau\rangle$ to ensure that the model's initial conditions (static equilibrium at the background heating rate) do not influence the results. The same time-averaging technique is applied in the original EBTEL tables, except the DEMs and DDMs are averaged over the entire model run (the original EBTEL runs assumed either steady heating or homogeneous nanoflares with a fixed delay of 10,000 s). This time averaging simulates the behavior of many small magnetic strands experiencing heating with the same properties but out of phase. Because the corona typically is optically thin and individual magnetic strands have cross sections well below observable resolutions, this is a simple way to represent the observable plasma distributions. More details about similar EBTEL++ models can be found in Schonfeld & Klimchuk (2020), and the code used to generate these EBTEL++ tables is available from a Zenodo© digital library (Schonfeld 2022).

As described in Section 3.5, GX Simulator uses Equation (6), when DDM distributions are available, or Equation (5) if only DEM distributions are available, to assign effective density–temperature pairs to each voxel crossed by a magnetic field line characterized by the averaged magnetic field $\langle B\rangle$ and length L for the purposes of visualization, or for computing emission using one of the radiation transfer codes that require them as input parameters. To help assess the differences between the parameters computed using these two approaches, we present a direct comparison in Figure 9.

The top row panels in Figure 9 show the density (panel (a)) and temperature (panel (b)) correlation plots between the respective parameters computed as moments of the DEM distributions (Equation (5)) and the DDM (Equation (6)) distributions provided by one of the EBTEL++ tables distributed with the GX Simulator package, namely `ebtel_scale=5_alpha=-2.5.sav`, which were precomputed for a set of 2,940 pairs of averaged volumetric heating rates, $\langle Q\rangle$, and magnetic loop half-lengths, $L_{1/2}$, spanning a parameter space ranging between $(1.58 \times 10^{-8} - 794.32)$ erg cm^{-3} s^{-1} and $(10^6 - 6.31 \times 10^{10})$ cm, respectively. In addition, the bottom row of panels in Figure 9 show a comparison between the corresponding density (panel (c)) and temperature (panel (d)) distributions computed by these two alternative means. Figure 9 demonstrates the expected good correlations of the DEM/DDM density and temperatures and a systematic shift toward lower values of the DDM-inferred n and T distributions.

4.4. Synthesized Microwave Emission from Multithermal Plasma Models

To illustrate the effect of the multithermal plasma distributions on the radio emission, we computed radio maps for the abovementioned AR 11520 model, using the full DEM/DDM treatment (Fleishman et al. 2021; see also Section 6). The resulting 5.7 and 17 GHz brightness temperature maps are displayed in Figure 10, where they are compared with the corresponding maps displayed in Figure 5, which were obtained using the "classical" isothermal treatment (based on the DEM moments, according to Equations (5)). The bright emission at these frequencies is produced mainly due to the thermal gyroresonance mechanism, which is sensitive to the DDM distribution, while the contribution of the thermal free–free emission (which is sensitive to the DEM distribution) is

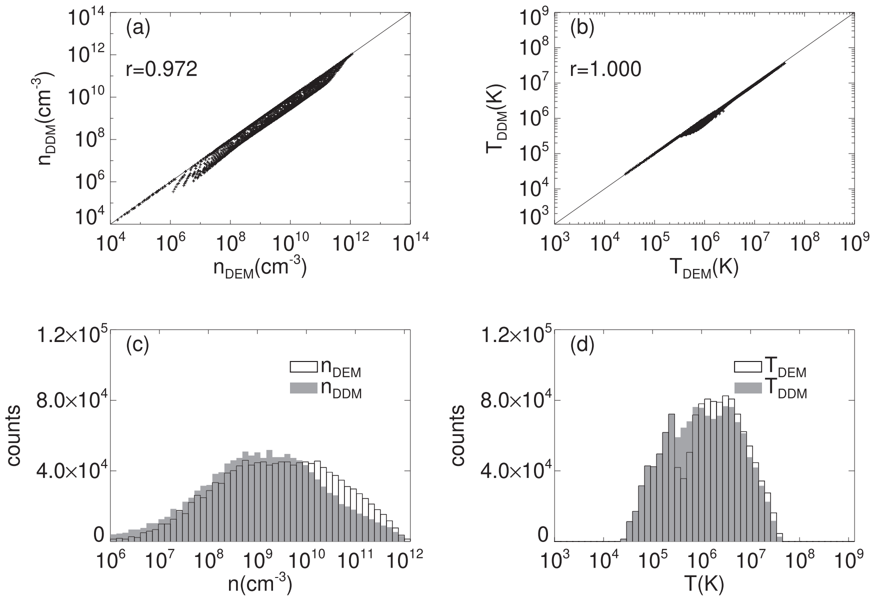


Figure 9. Comparison of thermal electron densities and temperature pairs computed from the moments of the DDM distributions (Equation (6)) vs. those computed from moments of the DEM distributions (Equation (5)) for the ebtel_scale = 5_alpha = −2.5.sav table. Top row: $n_{\rm DDM}$ vs. $n_{\rm DEM}$ (panel (a)) and $T_{\rm DDM}$ vs. $T_{\rm DEM}$ (panel (b)) correlation plots indicating a good linear correlation quantified by the correlation coefficients indicated in each panel. Bottom row: comparison between the DEM-derived distributions (black histograms) and the corresponding DDM-derived distribution (gray filled histograms) indicating a systematic shift toward lower values of the latter.

relatively low. One can see in Figure 10 that considering the multithermal plasma composition significantly affects both the image morphology (especially at lower frequencies) and the peak or average brightness temperatures; for the multithermal model, the brightness temperatures are higher due to contribution of electrons with energies above the average energies.

4.5. Integration of User-supplied Models in Pipeline-produced Model Skeletons

Although the top-level AMPP IDL procedure, i.e., `gx_fov2box.pro`, provides the user with a series of keyword switches that may be used to produce different types of standard GX Simulator models, for maximum flexibility, the experienced user may choose to design a custom AMPP IDL script by combining the low-level AMPP routines provided by the `gxbox` submodule. Moreover, the modular architecture of AMPP also allows for custom-designed computation blocks to be inserted to replace a standard block, at any point following the empty-box creation step, provided that the IDL box structures produced by such customized blocks remain compatible with the GX Simulator architecture. Such customized box structures may contain any number of additional, nonconflicting tags, which would be quietly ignored by GX Simulator.

For example, one may fill the empty-box `.NONE.` IDL structure produced by the AMPP initialization block with a magnetic field model obtained by alternate means. Similarly, one may replace an AMPP-generated chromo model with a nonstandard one, provided that all chromosphere-related tags of a standard "`.CHR.`" box structure are either replaced with equivalent data or kept unaltered.

Although the current GX Simulator release has no explicit provisions for allowing an alternative approach to our EBTEL-based coronal model, the user has the ability to import a DEM/DDM table produced by an alternative heating model, provided that the file structure of the standard EBTEL tables is preserved. Alternatively, one may generate a GX Simulator–compatible model populated with ready-to-use numerical thermal density and temperature pairs produced by any means (e.g., MHD simulations).

A detailed description of the internal data structure of a standard GX Simulator model produced by AMPP while allowing for its customization is provided in Appendix C.

5. Creation and Customization of Flaring Loop Models

The core of the flaring loop modeling capabilities included in the previous versions of the GX Simulator package have been described in Nita et al. (2015). In addition, the current version of GX Simulator includes the ability to use array-defined electron distributions for calculating the nonthermal radio emission, as well as a scripting feature for automation that allows users to step through many simulations to follow the temporal evolution of a flare.

The workflow of creating a flare model is to begin with a magnetic field model, usually generated by AMPP as described in Section 2, identify a likely field line in the magnetic field model that corresponds to the core (or axis) of a flaring loop, populate that loop with nonthermal particles (embedded in a nonflaring background solar atmosphere), and then calculate the emission of choice (among multiple EUV passbands, X-ray energies, or microwave frequencies). If one is interested in the initiation of a flare, a vector magnetogram close in time but prior to the flare may be the best choice, especially when a newly formed flux rope is involved. To model emission from a loop that has formed as a result of a flare, a vector magnetogram taken sometime after the flare may be a better choice. Once the flaring loop has been selected and populated, the properties of the model may be adjusted, and loops added, consistent with the spectroscopic imaging observations (i.e., parallel to EUV loops or joining HXR footpoints), until the emission from the model (when convolved with the instrument's PSF) adequately fits all observational constraints.

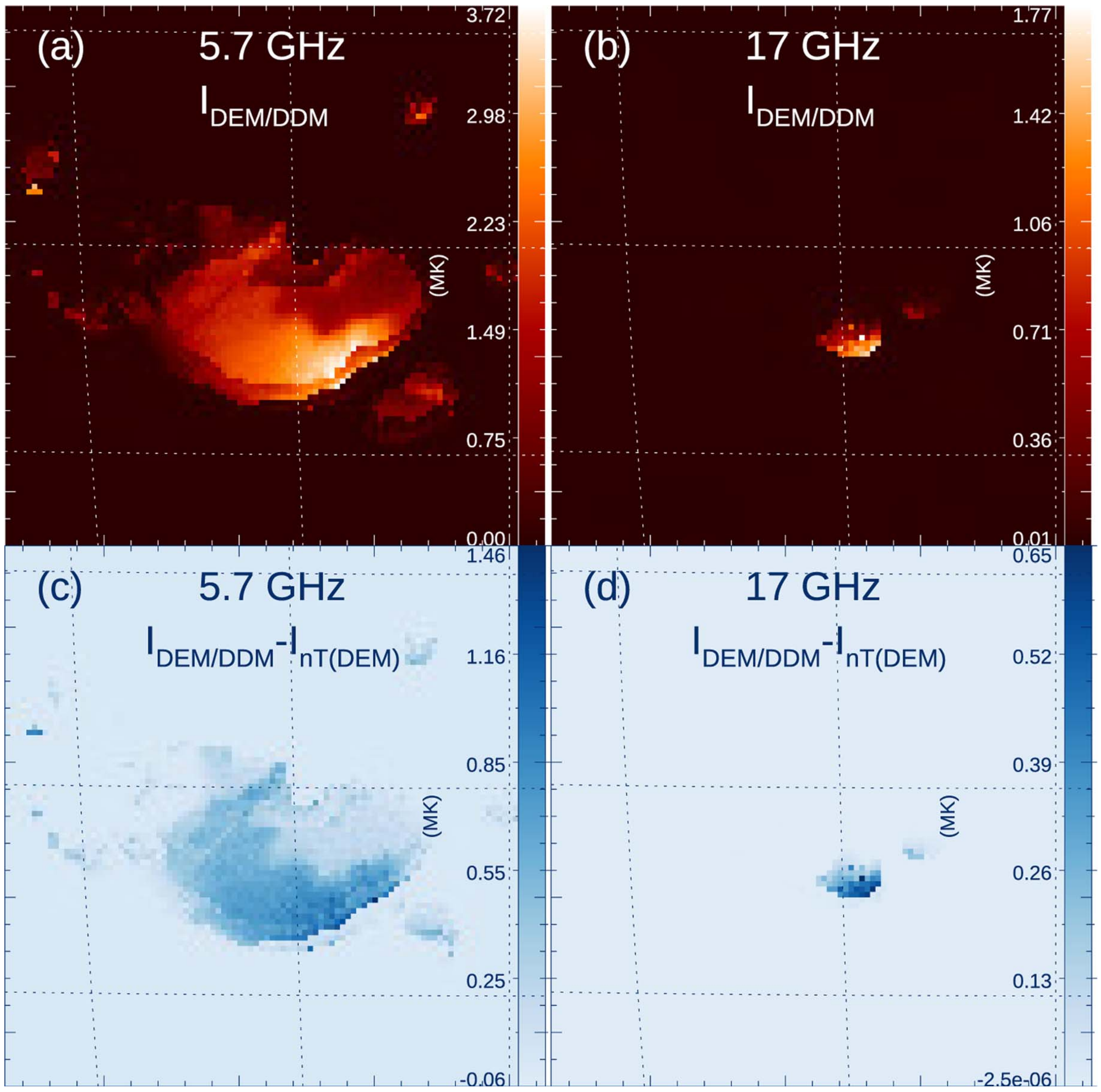


Figure 10. Comparison between the brightness temperature maps for AR 11520 computed using the full DEM/DDM treatment and the isothermal approximation, (shown on the top row in Figure 5), at frequencies 5.7 GHz (left column) and 17 GHz (right column). Top row: maps based on the multithermal DEM/DDM treatment. Bottom row: the respective difference maps. The color bars are in units of megakelvin.

Matching not only the morphology but also the frequency/wavelength dependence (shape and brightness of the microwave spectra, for example) is highly constraining, and when successful, the resulting forward fit model accounts naturally for the source inhomogeneity and finite instrument resolution limitations that adversely affect spectral inversions.

Once a successful model is obtained for one time in the flare, the new GX Simulator's scripting capability allows similar models to be quickly generated with evolving parameters to follow the temporal evolution of the flare. Fleishman et al. (2018) employed this sequential approach to model the evolution of the radio emission observed by EOVSA during the peak phase of the `SOL2015-06-22T17:50` `M6.5` solar flare. Fleishman et al. (2018) generated an evolving sequence of 30 models obtained by fine-tuning the analytically defined nonthermal electron distributions populating two of the four flux tubes that were previously identified and used by Kuroda et al. (2018) to model a single time frame corresponding to a local peak at 18:05:32 UT. Figure 11 displays three selected snapshots of this evolving model evolution (top row) and the corresponding match between the FOV-integrated synthetic microwave spectra computed by GX Simulator and the corresponding observational spectra produced by EOVSA.

Until recently (including the studies by Fleishman et al. (2018) and Kuroda et al. (2018) mentioned above), the nonthermal particle distribution that could be assigned in GX Simulator to a flaring loop model was limited to a single energy distribution. Thus, when a value for a power-law index and pitch angle were chosen for a flux tube, those same values were applied everywhere in the loop. The fast codes that calculate microwave emission have now been upgraded to permit array-defined distributions in energy and pitch angle that can vary in an arbitrary manner (Kuznetsov & Fleishman 2021; see Section 6.1 for more details). This upgrade opens the possibility to create a wide range of particle distributions as a function of time and position along a loop, based for example on 1D Fokker–Planck simulations, and from them calculate the emission maps as before. The importance of this approach to

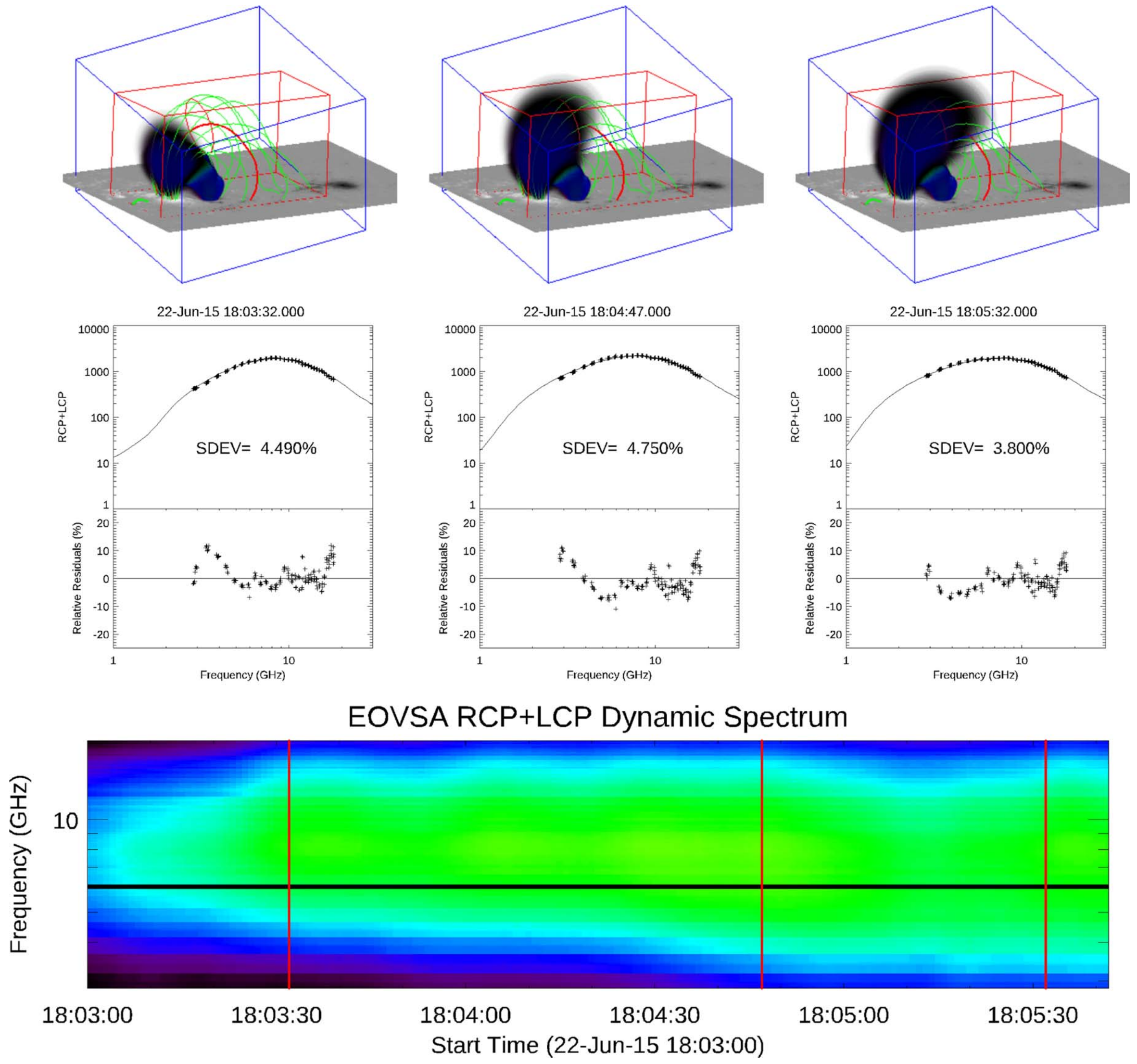


Figure 11. Dynamic 3D modeling with GX Simulator for the peak of the `SOL2015-06-22T17:50` flare observed with EOVSA. Top row: Analytically defined nonthermal electrons distributions in a system formed by four loops. Middle row: Model (lines) to EOVSA data (symbols) spectral comparisons and relative residuals, corresponding to the selected time frames illustrated in the top row. Bottom row: EOVSA Dynamic spectrum during the fitting interval (2.4–18 GHz, 4 s time resolution), with vertical red lines showing the times illustrated in the upper rows. (Adapted from Fleishman et al. 2018.)

the particle transport problem is that a time-dependent, physics-based simulation with varying levels of enhanced energy and pitch-angle diffusion can now be explored in a realistic loop geometry and compared quantitatively with the actual multi-wavelength observational data provided by instruments such as EOVSA, SRH, RHESSI, or STIX.

To make use of this added functionality, the current version of GX Simulator allows the user to export the physical and geometrical properties (B, n_e, T, α) along the axis of a selected flux tube. These properties may be then used in a time-dependent 1D particle transport code to calculate externally such numerical particle distributions. Then they can be imported back into the GX model and assigned to each volume element, from which various emissions can be calculated. This workflow is illustrated in Figure 12 for the event of 2017 September 4, an M5.4 event.

Following the steps in the order indicated by the gray circular arrow, one starts with (1) the multifrequency observations (EOVSA in this example, multicolored filled contours) and uses them to identify (2) the magnetic field line that fits the morphology. Then, (3) one may further compare with other data (a RHESSI image, in this example) to fine-tune the selection. Once a suitable axis of a flaring flux tube is identified matching the morphology of the flare, one has to quantitatively adjust plasma and particle parameters in the loop based on fitting all parameters inferred from observations (e.g., EOVSA, AIA, and other diagnostics). Then, (4) one has to extract from the model the magnetic field, density, temperature, and other atmospheric parameters as a function of distance along the axis. Panel 4 of Figure 12 illustrates the value of B/B_0 for the particular loop chosen for illustration. From here, any model that can calculate a 1D time-dependent electron distribution $f(E, \mu, s, t)$ can be used, where E is the electron energy, μ is the cosine of pitch angle, s is the distance in the 1D model, and t is the time.

One such model framework is the 1D electron transport simulations along the loop, wherein a source term injection of particles is introduced and the evolution of that distribution as it diffuses in both pitch angle and energy provides the required distribution along the loop. Such approach could include

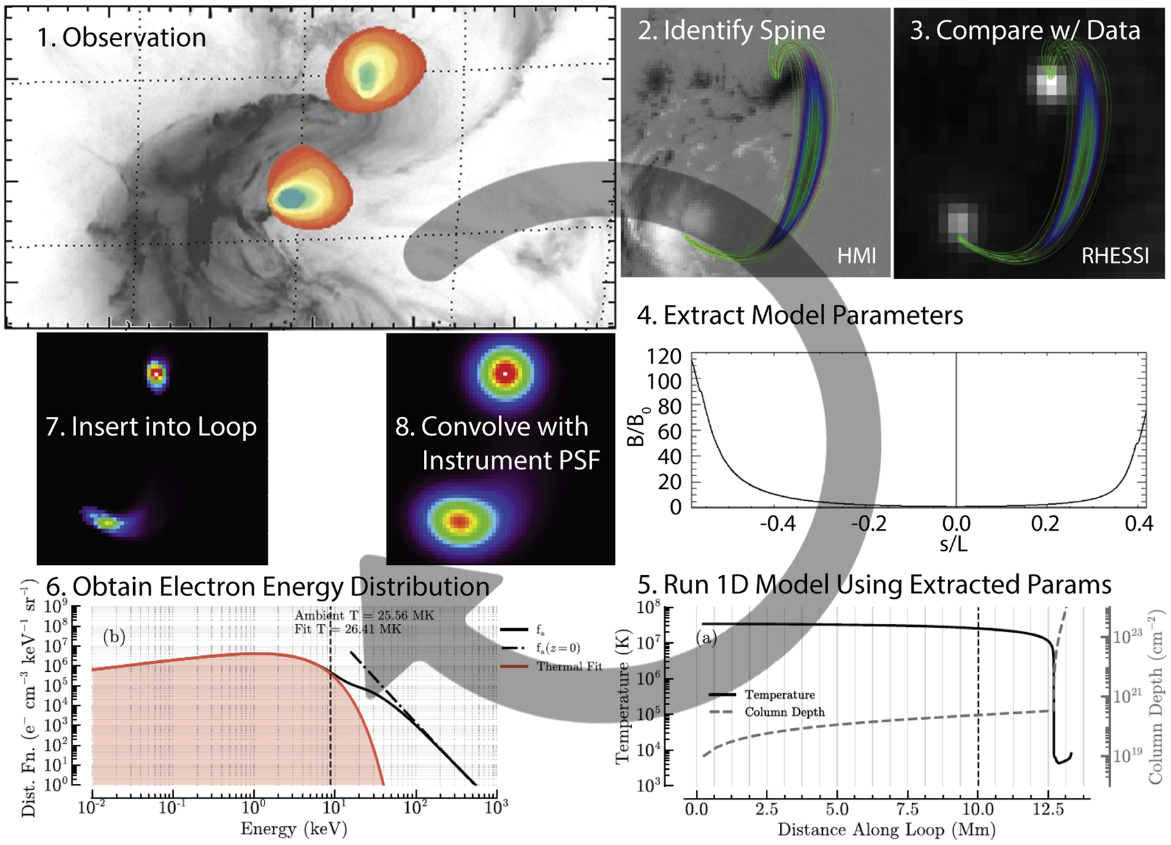


Figure 12. Illustration of the workflow from observation to simulated images. (1) The multifrequency EOVSA images for one time frame in the 2017 September 4 flare overlaid on an AIA 131 A image, showing two widely separated footpoints. (2) The HMI vector magnetogram (background gray scale is B_z) is used to calculate an NLFFF magnetic model from which a flux tube central field line is identified. (3) Other imaging data such as this comparison with RHESSI HXRs are used to refine the choice of central field line. (4) The model atmosphere parameters are extracted along the chosen field line. (5) The model parameters are used with a 1D Fokker–Planck code, starting from some assumed injected distribution and adjustable diffusion coefficients, to calculate (6) the energy distribution as a function of position and time. The energy distributions along the loop are then used by GX Simulator to calculate (7) the microwave emission maps, which are then (8) convolved with the EOVSA's PSF for quantitative comparison with EOVSA data.

stochastic acceleration (e.g., Park & Petrosian 1995), beam-plasma instability and Langmuir wave generation (e.g., Hannah et al. 2013), electric current associated with the magnetic field twist or return current (e.g., Gordovskyy et al. 2014), warm-target effects (Kontar et al. 2015), or electron anisotropy (Jeffrey et al. 2020). In Figure 12, step (5) shows the loop model temperature versus distance along the loop, but at each location (6) the Fokker–Planck code calculates an electron energy distribution; the one shown in this case, $f(E)$, is a thermal plus power-law distribution calculated at the 10 Mm distance. At a given time t, one may place the 1D model as a function of distance s along the loop back into GX Simulator, filling the 3D flux tube by a suitable lateral spread function (Gaussian or other) perpendicular to the central field line. This populates each of the relevant voxels in the 3D volume with $f(E, \mu)$, the array-defined quantity that can now be used to calculate the microwave emission and absorption along the LOS, and from that perform the radiative transfer (Section 6) to obtain the emergent brightness temperature at each frequency (step 7). Finally, the simulated maps at the resolution of the model must be convolved with the instrumental frequency-dependent point-spread function (step 8). By calculating the convolved emission for many time steps, a multifrequency simulation movie will result that can be compared directly with observations. Currently, we are implementing a transport code module that includes most of the relevant physical processes, which will compute and return numerical solutions for the flux tubes exported from GX Simulator.

6. Radiation Transfer Codes

The GX Simulator package provides a series of radiation transfer codes to compute microwave, X-ray, and EUV emission from the same 3D model, which either are written directly in IDL or use an IDL wrapper that calls platform specific external dynamic link libraries (DLL), which are precompiled for WIN32 or WIN64 OS systems, or shared library codes, which are automatically compiled when first called on Unix, Linux, or MAC platforms.

6.1. Fast GS and GR Codes

Initial versions of GX Simulator (Nita et al. 2015, 2018) included several radio radiation transfer codes obtained from either FORTRAN or C++ source codes developed by Fleishman & Kuznetsov (2010, 2014). For the flare case, the corresponding codes included gyrosynchrotron (GS) and free–free emission mechanisms, while for a nonflaring case, they included gyroresonance (GR) and free–free mechanisms. All codes took into account the mode coupling in the quasitransverse layers and the effect of limiting polarization (only circular, not linear, polarization survives while propagating through the coronal plasma).

In the most recent release, these radio radiation codes have been much improved and generalized. For the nonflaring case, the code performs radiation transfer in a multicomponent multithermal coronal plasma (Fleishman et al. 2021). The phenomena and physical processes included in the code are: (i) plasma chemical composition, (ii) ionization states of various elements (He–Zn) defined by the plasma temperature, (iii) tabulated exact Gaunt factors, (iv) contributions of the free–free emission due to collisions of thermal electrons with neutral atoms of hydrogen and helium, and (v) the ambient magnetic field. In addition, the code permits the plasma to be described by DEM/DDM, rather than a single pair of the temperature and density values; this is applied to both the free–free and GR emission calculation. The functionality available in older versions, such as mode coupling and limiting polarization, is preserved in this new release. Currently, this is the most complete and accurate version of the radio radiation transfer code that includes both GR and free–free emission mechanisms.

For the flaring case, Kuznetsov & Fleishman (2021) generalized the fast GS codes by Fleishman & Kuznetsov (2010) such that the distribution function of the nonthermal electrons can now be defined not only in a simplified analytical form but also in the form of numerically defined arrays. Thus, the output of numerical solutions of acceleration/transport models can now be employed directly by the tool. As in the previous versions, the continuous GS approximation provides very high computation speed, while, if necessary, the harmonic structure at low frequencies can be reproduced at some cost of execution time, using the exact GS codes. In addition, the free–free component is now treated based on the new theory developed by Fleishman et al. (2021), similarly to what has been implemented in the nonflaring codes. The calling sequences and interfaces were updated to ease the use of the codes—both within GX Simulator and in standalone applications.

6.2. X-Ray Codes

In the current GX Simulator release, the X-ray calculating block (routines `xray_tt.pro` and `xray_tt_albedo.pro`) has been substantially enhanced to include: (i) Hard X-ray calculations using a thick-target model (Brown 1971); and (ii) X-ray albedo correction (photosphere Compton back-scattered X-rays; see Kontar et al. (2006) for details). In addition, the thermal emission is now calculated using the CHIANTI database[12] for plasma temperatures above 0.09 keV. The default values use the OSPEX software (Schwartz et al. 2002) default abundances,[13] so that comparisons between OSPEX fits from RHESSI data and GX Simulator are readily available. The threshold temperature for CHIANTI calculations is controlled by the parameter `Te_thr = 0.09`. The X-ray flux for temperatures lower than `Te_thr` is calculated using a simplified, computationally faster, free–free bremsstrahlung expression. Although the CHIANTI-based method is slower, this approach provides more precise calculation of soft X-ray emission accounting for free–free, free–bound, and line emissions (see example spectrum in Figure 4.2 from Kontar et al. 2011). Similarly, the soft X-ray calculations are performed using the same routines as in OSPEX, allowing for direct comparisons with RHESSI observations and to change the default element abundances.

The hard X-ray flux from an emitting volume at a distance R ($\approx$1 au in the case of the Sun) is calculated using the mean electron flux (Brown et al. 2003):

$$I(\epsilon) = \frac{1}{4\pi R^2}\int_\epsilon^\infty n\bar{V}F(E)Q(\epsilon, E)dE, \qquad (10)$$

where $n\bar{V}F(E)$ is a mean electron flux integrated along line-of-light voxels. The cross section $Q(\epsilon, E)$ is approximated by the cross section used in OSPEX and tabulated as a lookup table for speed.

To account for both coronal and chromospheric hard X-ray emission, the hard X-ray flux is calculated differently in the chromosphere and the solar corona. The coronal X-ray emission is calculated directly using thin-target emission with $n\bar{V}F(E)$ defined by the modeler, while the chromospheric X-ray emission adopts the thick-target approach. The electrons penetrating into the chromosphere are used to calculate thick-target emission, so the mean electron flux in Equation (10) is calculated as

$$n\bar{V}F(E) = \frac{A}{K}E\int_E^\infty F_0(E_0)dE_0, \qquad (11)$$

where $K = 2\pi e^4\Lambda = 2.6\times10^{-18}\,\mathrm{cm^2\,keV^2}$ and A is the cross-sectional area of the electron beam.

The hard X-ray albedo contribution (angle-dependent Green's function correction for photospheric albedo) is computed based on the approach by Kontar et al. (2006) and uses pre-calculated Green's matrices used in OSPEX.[14] Using an anisotropic X-ray electron distribution, hard X-ray fluxes in upward and downward directions are calculated. These fluxes are used to calculate the anisotropic X-ray flux to the observer (Kontar & Brown 2006; Jeffrey & Kontar 2011; Dickson & Kontar 2013).

For illustration purposes, we show in the left panel of Figure 13 the result of the OSPEX fit model spectrum for the same `SOL2017-09-04T20:28:00 M5.5` event shown in Figure 12. The parameters estimated from this fit, i.e., thermal electron emission measure EM $= 3\times10^{48}\,\mathrm{cm^{-3}}$, plasma temperature $T = 1.51$ keV $= 1.75\times10^7\,K$, nonthermal electron flux $n_{\mathrm{th}} = 2.3\times10^{34}$ electrons s^{-1}, nonthermal electron energy index $\delta = 3.64$, and nonthermal electron minimum cutoff energy $E_{\min} = 19.0$ keV, have been used as input parameters for a GX Simulator flaring loop model, resulting in the model spectrum shown in the right panel of Figure 13.

To validate a model over the entire electromagnetic spectrum covered by observations, users may need to calculate both multienergy X-ray and multifrequency microwave emissions from the same flaring loop model. However, the radiation transfer codes `xray.pro`, `xray_tt.pro`, and `xray_tt_albedo.pro` only utilize a set of analytically defined nonthermal electron distributions, which are described in Nita et al. (2015). Thus, if a numerically defined nonthermal electron distribution is assigned to a loop model, as described in section Section 6.1, these X-ray computation routines will silently ignore it. Nonetheless, the recently released `xray_tt_albedo_arr.pro` may handle both

[12] https://www.chiantidatabase.org

[13] https://hesperia.gsfc.nasa.gov/ssw/packages/spex/doc/OSPEX_explanation.htm

[14] https://hesperia.gsfc.nasa.gov/ssw/packages/spex/doc/ospex_explanation.htm##Albedo%20Correction

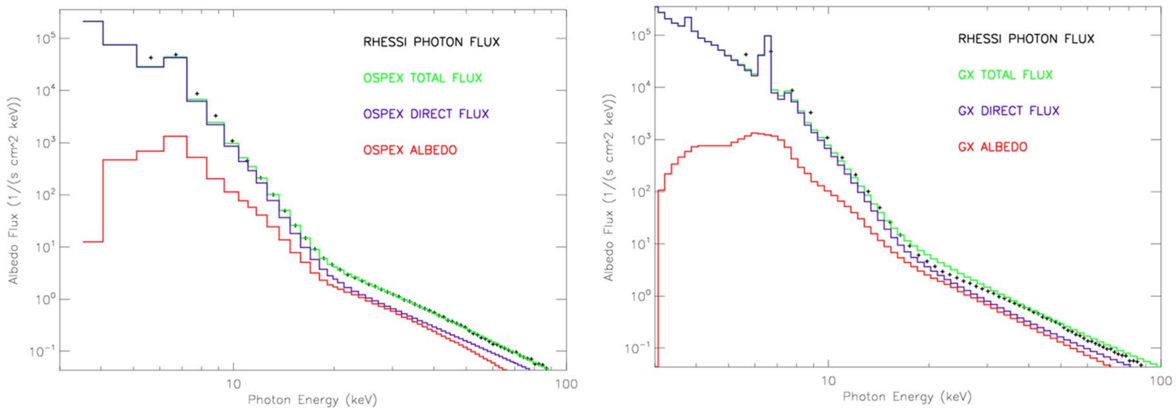


Figure 13. Left panel: OSPEX fit X-ray model spectrum for the same instance of the 2017 September 4 flare used for illustration purposes in Figure 12, showing observed photon spectrum (symbols), total OSPEX fit spectrum (green), direct OSPEX fit spectrum (blue), and OSPEX Albedo contribution (red). The parameters estimated from this fit are thermal electrons emission measure EM $= 3 \times 10^{48}$ cm^{-3}, plasma temperature $T = 1.51$ keV $= 1.75 \times 10^{7}$ K, nonthermal electrons density $n_{\rm th} = 2.3 \times 10^{34}$ electrons s^{-1}, nonthermal electron energy index $\delta = 3.64$, and nonthermal electron minimum cutoff energy $E_{\rm min} = 19.0$ keV. Right panel: The model X-ray spectrum generated by GX Simulator from a flaring loop model corresponding to the OSPEX fit parameters.

analytically and numerically defined nonthermal electron distributions.

6.3. EUV Codes

The main routine used by GX Simulator to compute EUV emission from a generic GX model produced by AMPP (Section 3.5), is `AIA.pro`, which computes the emission by convolving the DEM distribution from the model with the appropriate temperature response function, $G(T)$, of the SDO/AIA instrument:

$$I = \int \xi(T) G(T) dT. \tag{12}$$

The functionality of this routine was described by Nita et al. (2018), who discussed its performance and limitations based on a model-to-data comparison performed using a model of the AR 11072 on 2010 May 23 and observational SDO/AIA data. A recent upgrade of this rendering routine implemented the use of a time-dependent AIA response function that automatically adapts to the time of the model for which the synthetic EUV maps are computed. Currently, GX Simulator does not compute EUV spectra for comparison with EIS or IRIS data; adding this functionality will require an update of the tool.

6.4. Integration of User-supplied Radiation Transfer Codes

The GX Simulator plugin architecture allows straightforward integration of any user-supplied radiation transfer codes, provided that they are interfaced by IDL wrappers that abide by the GX Simulator calling convention given by the following prototype:

```
pro generic_wrapper, parms, rowdata,
info=info [, $
;optional input/output variables
nparms, rparms, user_arg1,user_arg1,..$
;optional input/output keyword variables
user_keyword1 = user_keyword1,..]
```

As indicated above, the user-defined IDL wrapper must accept three mandatory arguments: the strictly ordered and strictly named **`parms`** and **`rowdata`** variables, and the **`info`** keyword. In addition, the wrapper may accept two optional, strictly named input variables, **`nparms`** and **`rparms`**, as well as an arbitrary number of optional user-defined variables and/or keywords that may be used to pass internal user-defined data between subsequent calls of the wrapper that happen during the same geometrical scan of the model.

The use of the mandatory **`parms`** input variable is reserved for passing to the wrapper the model parameters corresponding to a two-dimensional geometrical slice of the 3D volume along the LOS direction, in the form of a $N_x \times N_z \times N_{\rm parms}$ double-precision numerical array, where N_x is the number of image pixels along one row of the synthetic map image, N_z is the number of nodes along any given line of sight, and N_{parms} the number of model properties needed to solve the radiation transfer equation.

Starting with the current release of the GX Simulator package, the optional, strictly named **`nparms`** and **`rparms`** have been introduced to allow more memory efficient passing of integer, or respectively, floating point model or setup parameters that are LOS-independent.

The use of the mandatory **`rowdata`** output variable is reserved for retrieving from the wrapper the multidimensional output data corresponding to one row of synthetic image, in the form of multidimensional floating point array that is expected to have at least two reserved dimensions, $N_x \times N_{chan}$, where N_{chan} represents the number of data channels to be computed, e.g., the number of microwave frequencies, the number of X-ray energy channels, or the number of EUV/UV passbands. However, the **`rowdata`** output variable may have additional user-defined dimensions, which may be used to accommodate, for example, different Stokes parameters and/or other radiation-mechanism-specific data channels.

The **`info`** mandatory keyword must be used by the wrapper to return, on request, metadata information describing the expected model parameters and the structure of the radiation transfer data output, in the form of an IDL structure. This structure contains a set of mandatory tags that are needed by GX Simulator to

1. retrieve the information needed to initialize the input parameter arrays described above and dynamically link them to the expected model data,

2. dynamically generate the wrapper specific user interface input fields needed to customize the options of the selected wrapper, and
3. customize the memory space and data visualization displays needed to hold and inspect the image cubes generated by the selected radiation transfer code.

Because a full description of the underlying GX Simulator architecture related to the radiation code calling conventions is beyond the scope of this paper, we refer the interested reader to inspect the IDL wrappers already included in the GX Simulator package, which may be used as templates for designing customized wrappers.

7. Conclusions

The GX Simulator tool is now mature enough to address various modeling needs in both flare and AR science. It offers convenient automatic ways to build 3D models, fine-tune them, generate synthetic observables, perform data-to-model comparison, and eventually produce 3D models compatible with all available observational constraints. Nevertheless, to expand its functionality as needed to address various science aspects increasingly demanding more sophisticated modeling approaches, incremental upgrades are released as they become available. A series of major upgrades are planned for the near future, which will be appropriately publicized as they become available. In addition, to address the need for increased computational efficiency, to improve its OS platform independence, and to potentially increase its user base by becoming less dependent on the proprietary nature of the IDL programming language, a gradual transition of the GX Simulator software infrastructure to modern, more widely used, open-source languages, such as Python or Julia, as well as Jupiter notebook integration, are being considered over the long term by the GX Simulator development team.

The study is supported by NSF grants AGS-2121632, AST-2206424, AGS-1743321, AGS-2130832, and NASA grants 80NSSC20K0718, 80NSSC20K0627, 80NSSC19K0068, and 80NSSC23K0090, to New Jersey Institute of Technology. S.J. S.'s contribution was supported by an appointment to the NASA Postdoctoral Program at the Goddard Space Flight Center, administered by Universities Space Research Association under contract with NASA. J.A.K. was supported by the Internal Scientist Funding Model competitive work package program at GSFC. A.A.K. and S.A.A. acknowledge financial support by the Ministry of Science and Higher Education of the Russian Federation.

Appendix A
GX SIMULATOR Automatic Model Production Pipeline IDL Script Example

Here, we list an AMPP script that may be used to generate a series of POT, BND, NAS, GEN, and CHR models (as described in Table 1) for AR11520.

```
;definition of input parameters
;time to search for SDO data:
time='12-Jul-2012 04:58:26'
;Carrington coordinates of the extrapola-
tion box base center (deg):
center=[82.45,-15.67]
;extrapolation box size:
size_pix=[240,168,200]
;extrapolation   box   resolution   in
kilometers:
dx_km=1000
;local model output repository:
out_dir='C:\gx_models'
;local data input repository:
tmp_dir='C:\jsoc_cache'
;AMPP calling sequence
gx_fov2box, time, center=center,$
size_pix=size_pix, dx_km=dx_km,$
out_dir=out_dir, tmp_dir=tmp_dir,$
/cea,/carrington, /euv, /uv,$
/save_bounds,out_files=out_files
```

At runtime, the script listed above prints out a set of console messages that may be used to to assess the system performance. Here, we list the console messages generated by this AMPP script when run on a Windows 10 system equipped with an Intel Xeon E-2286M CPU 2.4 GHz, 8 cores, 64 GB RAM:

```
;IDL console messages
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
BND.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
NAS.sav
Computing field lines for each voxel in the
model.
implementation in 104.818 seconds
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.NAS.
GEN.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.NAS.
CHR.CHR.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
BND.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
NAS.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.NAS.
GEN.sav
C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.NAS.
CHR.sav
```

Appendix B
IDL Script to Generate a Customized EBTEL Coronal Model of an Active Region and Associated Synthetic Microwave Maps

Here, we present a script that may be used to customize the EBTEL coronal model of the `hmi.M_720s.20120712_044626.W82S16CR.CEA.NAS.CHR.sav` AR11520 model produced by the AMPP script presented in the previous section and generate, for a selected field of view, synthetic microwave maps at two frequencies, 5.7 GHz and 17 GHz, matching the observing frequencies of the Siberian Solar Radio Telescope (SSRT) and Nobeyama Radio Heliograph (NoRH) radio interferometry arrays:

```
setenv, 'WCS_RSUN=6.96d8'; solar radius
assumed by GX AMPP
model=gx_importmodel('C:\gx_models\2012-
07-12\hmi.M_720s.20120712_044626.
W82S16CR.CEA.NAS.CHR.sav')
newgrid=model->SetFOV(xc=-86.68,yc=-320,
xfov=200,yfov=200,nx=100,ny=100); field of
view and map resolution setup
modDir= 'c:\moddir'
mapsDir=modDir+'\maps'
gxcDir=modDir+'\gxc'
if not file_test(modDir) then file_mkdir,
modDir
if not file_test(mapsDir) then file_mkdir,
mapsDir
if not file_test(gxcDir) then file_mkdir,
gxcDir
q=[0.000415000d, 100d, 1d9,1d, 0.75d];
volumetric heating formula parameters
q0_formula='q[0]';volumetric heating rate
formula
q_formula='q0*(B/q[1])^q[3]*(L/q[2])^q
[4]';volumetric heating formula
path=gx_ebtel_path('ebtel.sav'); select a
specific EBTEL table
f1=5.7d9; observing frequency of the Siber-
ian Solar Radio Telescope (SSRT)
f2=17d9 ; observing frequency of the
Nobeyama Radio Heliograph (NoRH)
df=alog10(f2/f1); logaritmic frequency
step between the two frequencies
t0=systime(/s)
map=gx_mwrender_ebtel(model,'AR_GRFF_-
nonLTE_64',q_parms=q,
q0_formula=q0_formula,
q_formula=q_formula,f_min=f1,df=df,
n_freq=2,gxcube=gxcube)
if obj_valid(map) then begin
    file=mapsDir+'\test.map'
    save,map,file=file
    message,'Map object saved to '+file,/cont
endif
if isa(gxcube) then begin
    file=gxcDir+'\test.gxc'
    save,gxcube,file=file
    message,'GXCUBE data structure saved to
    '+file,/cont
endif
message,string((systime(/s)-t0),for-
mat="('Synthetic maps computed in ',g0,'
seconds')"),/cont
```

The runtime console messages generated by this AMPP script when run on a Windows 10 system equipped with a 2.4GHz Intel Xeon E-2286M CPU and 64 GB RAM are listed below:

```
;IDL console messages
GX_FOV2BOX: Potential extrapolation performed
in 4.499 seconds
GX_FOV2BOX: Bound Box structure saved to
C:\gx_models\2012-07-12\hmi.M_720s.20120712_
044626.W82S16CR.CEA.BND.sav
GX_FOV2BOX: Performing NLFFF extrapolation
GX_FOV2BOX: NLFFF extrapolation performed
in 249.658 seconds
GX_FOV2BOX: NLFFF box structure saved to
C:\gx_models\2012-07-12\hmi.M_720s.20120712_
044626.W82S16CR.CEA.NAS.sav
GX_FOV2BOX: Computing field lines for each
voxel in the model.
GX_ADDLINES2BOX: Field line computation
performed using DLL implementation in
104.818 seconds
GX_FOV2BOX: Box structure saved to C:
\gx_models\2012-07-12\hmi.M_720s.20120712_
044626.W82S16CR.CEA.NAS.GEN.sav
GX_FOV2BOX: Generating chromo model.
GX_FOV2BOX: Chromo model generated in 9.248
seconds
GX_FOV2BOX: Box structure saved to C:
\gx_models\2012-07-12\hmi.M_720s.20120712_
044626.W82S16CR.CEA.NAS.CHR.CHR.sav
GX_FOV2BOX: This script generated the
following files:
GX_FOV2BOX: C:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
BND.savC:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
NAS.savC:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
NAS.GEN.savC:\gx_models\2012-07-12\hmi.
M_720s.20120712_044626.W82S16CR.CEA.
NAS.CHR.sav
```

Appendix C
Simple Steps to Generate GX SIMULATOR–Compatible Density–Temperature Models

C.1. Customized Coronal Models

To populate the coronal volume of the GXSimulator–compatible magnetic field model with numerical thermal plasma density and temperature pairs produced by alternative means, one may start from a standard `.GEN.` IDL structure, which contains three specific one-dimensional, same-size array tags, namely `IDX`, `LENGTH`, and `BMED`, where `IDX` is used to store the one-dimensional indices of the coronal voxels crossed by closed magnetic field line having the lengths and averaged magnetic fields stored in the other two arrays. One possible, straightforward approach would be to add to such a standard box structure two additional, same-size array tags, namely `N` and `T`, that would store the n, T pairs defined in exactly the same volume voxels referenced by the `IDX` tag. If this approach is chosen, the GX Simulator user would be offered the choice to synthesize multiwavelength thermal coronal emission directly from such n, T distributions, while still preserving the choice to use, as an alternative, the EBTEL approach described in Section 3.5 to compute and use the DEM-derived n, T pairs provided by Equation (5).

However, the `.GEN.` structure lists only voxels that are crossed by closed magnetic field lines within the model box. To assign n, T pairs to an unlimited set of voxels in the box, one may instead start from a standard `.POT.` or `.NAS.`, or compatible, structure to which the same-size array tags `IDX` and `N` and `T` may be added, in which case `IDX` may reference the entire volume or any portion of it.

C.2. Customized Chromosphere Models

As described in Section 3.4, a standard "`.CHR.`" IDL structure generated by AMPP contains a series of tags that define the properties of a nonuniform height chromosphere model that GX Simulator uses to replace the corresponding bottom layers of the uniform height coronal volume. These strictly named tags, relevant for such a combined chromospheric–coronal model, include:

1. BCUBE[$N_x \times N_y \times N_z$, 3]: the uniform height magnetic field model above photosphere produced by AMPP.
2. CORONA_BASE: the index of the lowest layer of the uniform height model *not to be replaced* by the chromosphere model.
3. CHROMO_LAYERS: number of nonuniform height vertical layers of the chromosphere model (default 90) to replace the first CORONA_BASE layers of the uniform model.
4. DZ [$N_x \times N_y \times (N_z -$ CORONA_BASE $+$ CHROMO_LAYERS)]: array defining the nonuniform heights (expressed in solar radius relative units) of each voxel of the combined model (all DZ heights are identical, starting with the CORONA_BASE: vertical layer).
5. CHROMO_BCUBE[$N_x \times N_y \times$ CHROMO_LAYERS, 3]: array containing the magnetic field components, B_x, B_y, and B_z, interpolated over the nonuniform height vertical layers of the chromosphere model.
6. TR [$N_x \times N_y$]: two-dimensional array containing the the vertical indices of the chromosphere model voxels corresponding to the transition region surface
7. TR_H [$N_x \times N_y$]: two-dimensional array containing the physical heights (expressed in solar radius relative units) of the TR voxels.
8. CHROMO_IDX: one-dimensional array containing the one-dimensional indices of the chromosphere volume for which the following physical plasma parameters may be provided as same-sized one-dimensional arrays:
   (a) CHROMO_N: electron densities.
   (b) CHROMO_T: plasma temperatures.
   (c) N_HTOT: hydrogen total densities (ionized + neutral).
   (d) N_HI: ionized hydrogen densities.
   (e) N_P: proton densities.

If the naming and array-sizing conventions defined above are strictly followed, the user may replace the default chromosphere model generated by the AMPP with a compatible model generated by alternative means.

ORCID iDs

Gelu M. Nita https://orcid.org/0000-0003-2846-2453
Gregory D. Fleishman https://orcid.org/0000-0001-5557-2100
Alexey A. Kuznetsov https://orcid.org/0000-0001-8644-8372
Sergey A. Anfinogentov https://orcid.org/0000-0002-1107-7420
Alexey G. Stupishin https://orcid.org/0000-0002-5453-2307
Eduard P. Kontar https://orcid.org/0000-0002-8078-0902
Samuel J. Schonfeld https://orcid.org/0000-0002-5476-2794
James A. Klimchuk https://orcid.org/0000-0003-2255-0305
Dale E. Gary https://orcid.org/0000-0003-2520-8396

References

Alissandrakis, C. E. 1981, A&A, 100, 197
Altyntsev, A., Lesovoi, S., Globa, M., et al. 2020, STP, 6, 30
Barnes, W. T., Bradshaw, S. J., & Viall, N. M. 2019, ApJ, 880, 56
Barnes, W. T., Cargill, P. J., & Bradshaw, S. J. 2016, ApJ, 829, 31
Bobra, M. G., Sun, X., Hoeksema, J. T., et al. 2014, SoPh, 289, 3549
Bradshaw, S. J., & Viall, N. M. 2016, ApJ, 821, 63
Brown, J. C. 1971, SoPh, 18, 489
Brown, J. C., Emslie, A. G., & Kontar, E. P. 2003, ApJL, 595, L115
Cao, W., Gorceix, N., Coulter, R., et al. 2010, AN, 331, 636
Cargill, P. J. 2014, ApJ, 784, 49
Cargill, P. J., Bradshaw, S. J., & Klimchuk, J. A. 2012a, ApJ, 752, 161
Cargill, P. J., Bradshaw, S. J., & Klimchuk, J. A. 2012b, ApJ, 758, 5
Dickson, E. C. M., & Kontar, E. P. 2013, SoPh, 284, 405
Fleishman, G., Nita, G., Kuroda, N., et al. 2018, ApJ, 859, 17
Fleishman, G. D., Anfinogentov, S., Loukitcheva, M., Mysh'yakov, I., & Stupishin, A. 2017, ApJ, 839, 30
Fleishman, G. D., Anfinogentov, S. A., Stupishin, A. G., Kuznetsov, A. A., & Nita, G. M. 2021a, ApJ, 909, 89
Fleishman, G. D., Kleint, L., Motorina, G. G., Nita, G. M., & Kontar, E. P. 2021b, ApJ, 913, 97
Fleishman, G. D., & Kuznetsov, A. A. 2010, ApJ, 721, 1127
Fleishman, G. D., & Kuznetsov, A. A. 2014, ApJ, 781, 77
Fleishman, G. D., Kuznetsov, A. A., & Landi, E. 2021, ApJ, 914, 52
Fontenla, J. M., Curdt, W., Haberreiter, M., Harder, J., & Tian, H. 2009, ApJ, 707, 482
Forsythe, G. E., Malcolm, M. A., & Moler, C. B. 1977, Computer Methods for Mathematical Computations (Englewood Cliffs, NJ: Prentice-Hall)
Freeland, S., & Handy, B. 1998, SoPh, 182, 497
Gary, D., Chen, B., Dennis, B., et al. 2018, ApJ, 863, 83
Goode, P. R., & Cao, W. 2012, Proc. SPIE, 8444, 844403
Gordovskyy, M., Browning, P. K., Kontar, E. P., & Bian, N. H. 2014, A&A, 561, A72
Grechnev, V. V., Lesovoi, S. V., Smolkov, G. Y., et al. 2003, SoPh, 216, 239
Hannah, I. G., Kontar, E. P., & Reid, H. A. S. 2013, A&A, 550, A51
Jeffrey, N. L. S., & Kontar, E. P. 2011, A&A, 536, A93
Jeffrey, N. L. S., Saint-Hilaire, P., & Kontar, E. P. 2020, A&A, 642, A79
Kaufmann, P., Raulin, J. P., Correia, E., et al. 2001, ApJL, 548, L95
Klimchuk, J. A., Patsourakos, S., & Cargill, P. J. 2008, ApJ, 682, 1351
Kontar, E. P., & Brown, J. C. 2006, ApJL, 653, L149
Kontar, E. P., Brown, J. C., Emslie, A. G., et al. 2011, SSRv, 159, 301
Kontar, E. P., Jeffrey, N. L. S., Emslie, A. G., & Bian, N. H. 2015, ApJ, 809, 35
Kontar, E. P., MacKinnon, A. L., Schwartz, R. A., & Brown, J. C. 2006, A&A, 446, 1157
Krucker, S., Hurford, G. J., Grimm, O., et al. 2020, A&A, 642, A15
Kuroda, N., Gary, D., Wang, H., et al. 2018, ApJ, 852, 32
Kuznetsov, A. A., & Fleishman, G. D. 2021, ApJ, 922, 103
Lemen, J. R., Title, A. M., Akin, D. J., et al. 2012, SoPh, 275, 17
Lesovoi, S., Altyntsev, A., Kochanov, A., et al. 2017, STP, 3, 3
Lin, R. P., Dennis, B. R., Hurford, G., et al. 2003, SoPh, 210, 3
Machado, M. E., Avrett, E. H., Vernazza, J. E., & Noyes, R. W. 1980, ApJ, 242, 336
Mandrini, C. H., Démoulin, P., & Klimchuk, J. A. 2000, ApJ, 530, 999
Metcalf, T. R., De Rosa, M. L., Schrijver, C. J., et al. 2008, SoPh, 247, 269
Nakajima, H., Nishio, M., Enome, S., et al. 1994, IEEEP, 82, 705
Nita, G., Fleishman, G., Kuznetsov, A., Kontar, E., & Gary, D. 2015, ApJ, 799, 236
Nita, G., Hickish, J., Macmahon, D., & Gary, D. 2016, JAI, 5, 1641009
Nita, G., Viall, N., Klimchuk, J., et al. 2018, ApJ, 853, 66
Nita, G. M., Fleishman, G. D., Kuznetsov, A. A., Kontar, E. P., & Gary, D. E. 2015, ApJ, 799, 236
Nita, G. M., Kontar, E., & Viktor, V. 2023, Gelu-Nita/GX_SIMULATOR: gx_simulator, v3.0.0, Zenodo, doi:10.5281/zenodo.7882023
Nita, G. M., Viall, N. M., Klimchuk, J. A., et al. 2018, ApJ, 853, 66
Park, B. T., & Petrosian, V. 1995, ApJ, 446, 699
Pesnell, W. D., Thompson, B. J., & Chamberlin, P. 2012, SoPh, 275, 3
Rimmele, T., Wagner, J., Keil, S., et al. 2010, Proc. SPIE, 7733, 77330G
Rudenko, G., & Anfinogentov, S. 2014, SoPh, 289, 1499
Scherrer, P. H., Schou, J., Bush, R. I., et al. 2012, SoPh, 275, 207

Schonfeld, S. J. 2022, schonfsj/ebtel_parallel: ebtel_parallel v1.0.0, v1.0.0, Zenodo, doi:10.5281/zenodo.7154827

Schonfeld, S. J., & Klimchuk, J. A. 2020, ApJ, 905, 115

Schwartz, R. A., Csillaghy, A., Tolbert, A. K., et al. 2002, SoPh, 210, 165

Selhorst, C., Silva, A., & Costa, J. 2005, A&A, 433, 365

Selhorst, C. L., Silva-Válio, A., & Costa, J. E. R. 2008, A&A, 488, 1079

Stupishin, A. 2020, Magnetic Field Library: NLFFF and magnetic lines, v3.4.22.1025-beta, Zenodo, doi:10.5281/zenodo.3896222

Thompson, W. T. 2006, A&A, 449, 791

Tritschler, A., Rimmele, T., Berukoff, S., et al. 2016, AN, 337, 1064

Trottet, G., Raulin, J.-P., Mackinnon, A., et al. 2015, SoPh, 290, 2809

Ugarte-Urra, I., Warren, H. P., Upton, L. A., & Young, P. R. 2017, ApJ, 846, 165

Viall, N. M., & Klimchuk, J. A. 2012, ApJ, 753, 35

Viall, N. M., & Klimchuk, J. A. 2013, ApJ, 771, 115

Wheatland, M. S., Sturrock, P. A., & Roumeliotis, G. 2000, ApJ, 540, 1150

Wiegelmann, T. 2004, SoPh, 219, 87

Withbroe, G. L., & Noyes, R. W. 1977, ARA&A, 15, 363