Modeling Quasi-Static Solar Coronal Loops with Nonuniform Energy Input

Donald G. Luttermoser
Department of Physics and Astronomy
East Tennessee State University

1 Introduction

The NASA SkyLab mission had a solar telescope on board that took many high-resolution images of the Sun at X-ray wavelengths. One of the most important findings of SkyLab was that the solar corona is composed of numerous magnetic loop-like structures that can last from days to weeks in a relatively static state. As such, these structures have been come to known as **quasi-static solar coronal loops** (SCL). Rosner, Tucker, & Vaiana (1978) was one of the first groups to try and model these SCLs. Solar astrophysicists have continued to improve the modeling these features through the 21st century (*e.g.*, Aschwanden & Schrijver 2002). The work reported here was carried out in the fall of 1980 for the author's undergraduate thesis under the direction of Dr. Richard Teske of the Department of Astronomy at the University of Michigan in Ann Arbor. This manuscript has been updated in 2005 to be used as a possible project for students enrolled in the author's course *Computational Physics* at East Tennessee State University.

Here I report on a code, written in FORTRAN 77, that generates quasi-static solar coronal loop (SCL) models for various types of energy input functions. The differential equations solved in this work are from Vesecky, Antiochos, and Underwood's (1979) paper on SCL modeling. The code reported here includes nonuniform energy inputs into the loop, whereas Vesecky *et al.* paper only describe a uniform energy input.

The basis of this code is that a magnetic loop is generated by a line dipole buried beneath the chromosphere or photosphere (depending on the loop geometry). The loops that are observed on the Sun, appear to have lifetimes on the order of days. Hence, a model can be constructed that describes the *quasi-static* portion of the loop's life, that is, when the mass flow into or out of the loop is negligible. Also, because the observed lifetime of these loops is frequently greater than typical timescales for energy losses by radiation or conduction, a continuous energy input must be assumed. Thus, the energy input must be balanced by the radiative loss flux and the conductive energy flux into (or out of) the loop.

2 Solar Coronal Loop Physics

We will idealize a solar coronal loop as shown in Figure 1. The loop will remain stable as long as the energy into the loop is balanced by the energy loss from the loop, and internal pressure is balanced by the weight of the gas in the loop. This energy balance equation can be expressed as

$$\vec{\nabla} \cdot \vec{F_c} = \epsilon - E_r, \tag{1}$$

where F_c is the conductive flux [erg/s/cm^2] along the loop, ϵ is the energy input per unit volume [erg/s/cm^3], and E_r is the radiative loss per unit volume [erg/s/cm^3]. Because the conductive flux is along magnetic field lines, Eq. (1) simplifies to the one dimensional case described by

$$\frac{1}{A(s)} \frac{d}{ds} \left[A(s)\, \kappa \, \frac{dT}{ds} \right] = n_e^2 \, \Lambda(T) - \epsilon. \tag{2}$$

Here, $A(s)$ is the cross-sectional area of the loop at the axis position s, $\kappa = \kappa_\circ T^{5/2}$ is the classical thermal conductivity ($\kappa_\circ = 10^6$), n_e is the electron density, and $\Lambda(T)$ is the radiative loss function from Raymond, Cox, & Smith (1976). The plasma is also assumed to be completely ionized and in hydrostatic equilibrium (HSE),

$$\frac{dP}{ds} = \rho \, g_s, \tag{3}$$

where $g_s = g_\odot \, \sin\phi$ is the component of gravity parallel to the magnetic field, hence parallel to s, $g_\odot$ is the surface gravity of the Sun, and the angle ϕ is related to the path length s and the radius R of the circular loop structure via

$$s = R\,\phi \tag{4}$$

with $\phi = 0$ representing the direction to the line dipole buried beneath the solar surface.

The variation of the cross-sectional area along s is described by

$$A(s) = A_t \sin^2(\phi/2), \tag{5}$$

where A_t is the cross-sectional area at the loop's apex and is related to the cross-sectional area at the loop's base via

$$\Gamma = A_t/A_b = D/d. \tag{6}$$

Here, Γ essentially describes the geometry of the loop and is considered to be one of the main input parameters to the code. D corresponds to the full diameter of the loop and d is the depth of the line dipole beneath the chromosphere (note that the top of the chromosphere is equivalent to the base of the loop). These two lengths are related to the height h that the loop rises above the chromosphere through $h = D - d$. With these definitions, the position of the base of the loop along s is described by

$$\phi_b = \frac{s_b}{R} = \frac{2}{\sin(1/\sqrt{\Gamma})}. \tag{7}$$

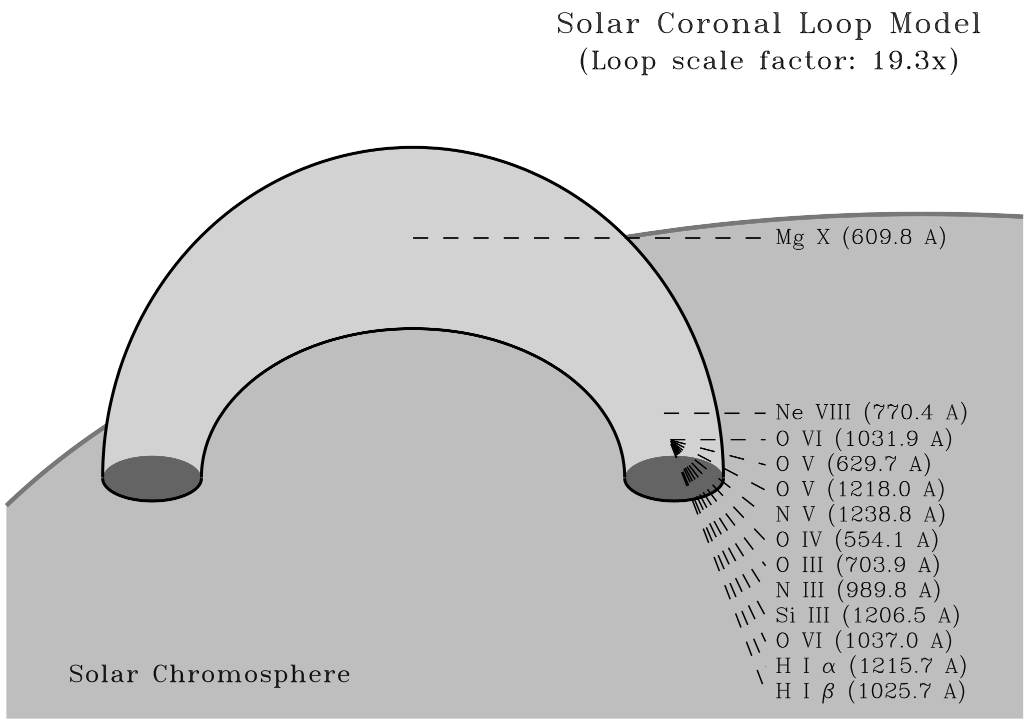

Figure 1: An idealized solar coronal loop. The labels indicate where the spectral lines of various ions form.

The energy input can either be assumed uniform (*i.e.*, independent of temperature) or nonuniform. In any case, it is described by the equation:

$$\epsilon = \gamma\, T^{-\alpha}, \tag{8}$$

where γ is a constant coefficient and α is the heating mechanism exponent. In this work, only 3 different values for α, hence heating mechanism candidates, will be assumed possible:

- Alfv́en mode/anomalous conduction damping ($\alpha = 1$).
- Acoustic mode/shock wave damping ($\alpha = 0.5$).
- Mode/mode conversion ($\alpha = 0$, uniform energy input).

The coefficient of the heating mechanism, γ, is dependent upon the dominance of the thermal conductive heating in the energy balance at the base of the loop. For stable loops to exist, the energy input at the base of the loop must be small with respect to the radiative losses there. One can define the ratio

$$\beta = \frac{\gamma\, T_b^{-\alpha}}{n_b^2\, \Lambda_b}, \tag{9}$$

or solving for γ,

$$\gamma = \beta\, n_b^2\, \Lambda_b\, T^{\alpha}. \tag{10}$$

From the modeling carried out in this work, loop stability was only achieved for $0 < \beta \leq 0.2$. The code written for this work allows the user to input either a value for β or to manually input γ. Note that the code warns the user if the input value of γ violates the stability criterion above.

Reformatting the momentum (*i.e.*, HSE) and energy equations, Eq. (3) and (2) respectively, one can write 3, first-order, ordinary differential equations with temperature T, electron density n_e, and conductive flux F_c as dependent variables and the distance s along the loop from the dipole origin up through the apex of the loop as the independent variable. The resulting equations are

$$\frac{dT}{ds} = \frac{-F_c}{A(s)\,\kappa_\circ\,T^{5/2}}, \tag{11}$$

$$\frac{dn_e}{ds} = \frac{n_e}{T}\left[\frac{m\,g_s}{2\,k} + \frac{F_c}{A(s)\,\kappa_\circ\,T^{5/2}}\right], \tag{12}$$

$$\frac{dF_c}{ds} = A(s)\,\left[\epsilon(T) - n_e^2\,\Lambda(T)\right], \tag{13}$$

where m is the proton mass, k is the Boltzmann constant, and the rest of the variables are as described above.

The boundary conditions at set such that $F_c = 0$ at both the base and the apex of the loop; zero at the base since the temperatures are so low and zero at the apex since we are assuming that $T_{\max}$ occurs at the apex of the loop. The temperature and electron density at the base must also be input to the code.

3 Execution of the Code

The user is able to enter the input parameters from either a "front-end" GUI (written in either IDL or Java) or from a data file as shown in Table 1. Once the input data has been read, the code simultaneously solves the three differential equations given in Eqs. (11-13) using Eq. (8) for the energy input equation. Fifty-four (54) models are generated using the following combinations of input parameters: LENGTH (cm) = 8.00E+08, 2.00E+09, 5.00E+09; GAMMA = 2, 10, 50; BETA = 0.05, 0.15; ALPHA = 0, 0.5, 1; RAD-BASE (cm) = 4.00E+08; T-BASE = 3.00E+04; and NE-BASE (cm^{-3}) = 1.00E+10 (see the next section about this input parameter). The temperature gradients are set at 0 at both the top and base of the loop (however the user has the option to change this). See the discussion about dT/ds at the loop's apex in the next section. One then choose one of the Vesecky *et al.* models and calculates it with the code for comparison with their results (*i.e.*, in order to make sure this code is operating as it should — the ECOEFF input parameter is used for this operation). **A test model is first ran before running all 54 nonuniform**

Table 1: Input Parameters for the Standard SCL Model

Parameter	Value	Definition
NZONES	100	Number of zones in the model.
GAMMA	2.0	Cross-sectional area ratio between apex and base.
LENGTH	2.00E+09	Arc length (cm) of the loop from base to apex.
RAD-BASE	4.00E+08	Radius (cm) of the loop cross section at the base.
T-BASE	3.00E+04	Temperature (K) at the base of the loop.
NE-BASE	1.00E+10	Electron density (cm^{-3}) at the base of the loop.
DTDS-BASE	0.0	Temperature gradient (K/cm) at the base of the loop.
DTDS-TOP	0.0	Temperature gradient (K/cm) at the apex of the loop.
ALPHA	0.0	Exponent for the energy input scaling law (0, 0.5, 1).
BETA	0.1	Energy input coefficient parameter (0.02 to 0.2).
ECOEFF	5.00E-04	Energy input coefficient, BETA ignored if given.

energy input models. The results are then analyzed by making plots of T, n_e, and F_c as a function of s (*i.e.*, path length along the loop axis) for all models in order to see how the input parameters affect your models.

Finally, the differential emission measure, $\xi(T)$, is calculated, which is an indication of the amount of the emitting material near temperature T, is given by the following formula:

$$\xi(T) = \frac{A(s)\, n_e^2}{|dT/ds|}. \tag{14}$$

Since the boundary conditions require that $dT/ds \rightarrow 0$ at the base and the apex of the loop, $\xi(T) \rightarrow \infty$ at these points. These singularities, however, are integrable, as required by the observations.

4 The Details of the Code

This computer program numerically integrates Equations (11), (12), & (13) as it steps through the loop from its base to its apex using the differential equation solver described by Shampine & Gordon (1975), which is basically a variable order of interpolation, variable step size formulation of the classic Adams method. The stepping procedure is subject to a local error criterion containing both relative and absolute error, set to 10^{-4} and unity, respectively. The temperature, electron density, and conductive flux are evaluated at NZONES locations (default is 100) along this loop axis. Typically one chooses either 10,000 K or 30,000 K as the base temperature. The *geometry* parameter, Γ, should lie between 2 and 50 (corresponding to the types of loops seen on the Sun) and the length of the loop, from base to apex, must also be input. The temperature gradients at the loop's apex and base are necessary boundary conditions since they set the conductive flux at those 2 points. These should be set to 0 on input.

Note that an apex temperature gradient of zero will never be achieved in a numerical integration. As such, the code sets the convergence criterion such that dT/ds must be less than 10^{-8} (K/cm) at the loop's apex when the value 0 is input. In order for convergence to be achieved, $dT/ds \leq 10^{-8}$ (K/cm) must be valid at the loop's apex. Note that this convergence criterion can be set to a value other than 10^{-8} (which is the default when '0' is entered in the input data) by setting DTDS-APEX to whatever value is desired.

Finally, the most critical input parameter is the electron density at the base of the loop. From past modeling experiences, the accuracy of this value is paramount in converging a loop. Since one does not know *a priori* what this value should be for a given loop geometry, your code should continuously modify this input value until convergence is achieve, or until the maximum number of iterations is reached. Note that if the maximum number of iterations is reached before convergence is achieved, the loop model is still delivered to the user, but an error message is written to the output file.

Three output files are generated by the code. The *long* output file (*e.g.*, loop.out) contains every output message that the code generates. This file is used to investigate any troubles should convergence not be achieved. The *model* output file (*e.g.*, loop.mod) contains a listing of the model itself. Finally, the differential emission measure data is stored in a third output file (*e.g.*, loop.ems). Note that in order to prevent overwriting of files on a Unix or Windows machine, a version number is generated and included as part of the filename (*i.e.*, the "01" in loop01.out).

Since this code is now used as a possible project for *Computational Physics*, the Analysis and Conclusion sections have been removed from this manuscript.

5 References.

Aschwanden, M.J., & Schrijver, C.J. 2002, ApJS, 142, 269.

Raymond, J.C., Cox, D.P., & Smith, B.W. 1976, ApJ, 204, 290.

Rosner, R., Tucker, W.H., & Vaiana, G.S. 1978, ApJ, 220, 643.

Shampine & Gordon 1975, Computer Solution of Ordinary Differential Equations (San Francisco: Freeman).

Vesecky, J.F., Antiochos, S.K., & Underwood, J.H. 1979, ApJ, 233, 987.