FastQSL: A Fast Computation Method for Quasi-separatrix Layers

PeiJin Zhang1,2,3,4, Jun Chen5,6, Rui Liu4,7, and ChuanBing Wang4,7

1 Institute of Astronomy and Astronomical Observatory, Bulgarian Academy of Sciences, Sofia 1784, Bulgaria
2 ASTRON, The Netherlands Institute for Radio Astronomy, Oude Hoogeveensedijk 4, 7991 PD Dwingeloo, The Netherlands
3 Astronomy & Astrophysics Section, Dublin Institute for Advanced Studies, Dublin 2, Ireland
4 CAS Key Laboratory of Geospace Environment, School of Earth and Space Sciences, University of Science and Technology of China, Hefei, Anhui 230026, People’s Republic of China
5 School of Astronomy and Space Science, Nanjing University, Nanjing 210023, People’s Republic of China; e12718chenjun@nju.edu.cn
6 Key Laboratory of Modern Astronomy and Astrophysics (Nanjing University), Ministry of Education, Nanjing 210093, People’s Republic of China
7 CAS Center for the Excellence in Comparative Planetology, University of Science and Technology of China, Hefei, Anhui 230026, People’s Republic of China

Received 2021 June 6; revised 2022 August 13; accepted 2022 August 26; published 2022 September 21

Abstract

Magnetic reconnection preferentially takes place at the intersection of two separatrices or two quasi-separatrix layers, which can be quantified by the squashing factor $Q$, whose calculation is computationally expensive due to the need to trace as many field lines as possible. We developed a method (FastQSL) optimized for obtaining $Q$ and the twist number in a 3D data cube. FastQSL utilizes the hardware acceleration of the graphics processing unit and adopts a step-size adaptive scheme for the most computationally intensive part: tracing magnetic field lines. As a result, it achieves a computational efficiency of 4.53 million $Q$ values per second. FastQSL is open source, and user-friendly for data import, export, and visualization.

Unified Astronomy Thesaurus concepts: Solar magnetic fields (1503); GPU computing (1969)

1. Introduction

Chromosphere flare ribbons often coincide with the footprints of separatrices or quasi-separatrix layers (QSLs) (Priest & Démoulin 1995; Démoulin et al. 1996, 1997), which embed the favorable sites for 3D magnetic reconnection, such as null points, separatrices (intersection of two separatrices), and quasi-separators (intersection of two QSLs) of the magnetic field (Priest & Forbes 2000; Pontin 2011), where a large gradient of magnetic connectivity is present.

The squashing factor $Q$ quantifies the connectivity change of magnetic field lines (Titov et al. 2002; Titov 2007); $Q$ is defined through a mapping from one surface $S_1$, which a magnetic field line threads at $(x_1, y_1)$, to another surface $S_2$, which the field line threads at $(x_2, y_2)$:

$$\Pi_{12}: (x_1, y_1) \rightarrow (x_2, y_2).$$

(1)

The Jacobian matrix of differential mapping is expressed as

$$D = \begin{pmatrix} \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial y_1} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial y_1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

(2)

A full description of $Q$ that considers the variance of the covariant metric tensor on $S_1$ and $S_2$ is defined by Equations (11), (12), and (14) in Titov (2007). If $S_1$ and $S_2$ are the two boundaries of a cuboid coordinated with the same Cartesian coordinate system, the value of $Q$ at $(x_1, y_1)$ is

$$Q(x_1, y_1) = \frac{a^2 + b^2 + c^2 + d^2}{|\det D|^{1/2}}.$$  

(3)

$|\det D|$ is often replaced by its equivalence $|B_{n,1}/B_{n,2}|$ for mitigating numerical error, where

$$B_{n,1} = (B \cdot n)|_{S_1},$$

(4)

is the component normal to $S_1$ of $(x_1, y_1)$ and $n$ is the normal unit vector of $S_1$; the form is similar for $B_{n,2}$. Since Titov et al. (2002) proved $Q(x_1, y_1) = Q(x_2, y_2)$, Aulanier et al. (2005) expanded the definition of $Q$ into 3D space by

$$B \cdot \nabla Q = 0,$$

(5)

i.e., values of $Q$ are invariant along a field line. Separatrices are located where $Q = \infty$, and QSLs are located where $Q \gg 2$, the theoretical minimum of $Q$. Along with tracing field lines, the twist number, a measure of how many turns two infinitesimally close field lines wind about each other, can be calculated without much additional effort by Equation (16) in Berger & Prior (2006)

$$T_w = \int_L \frac{\nabla \times B \cdot B}{4\pi B^2} dl,$$

(6)

where the integral range $L$ is a segment of a magnetic field line.

In this work, we take advantage of graphics processing unit (GPU) computing, which is more efficient and economic compared to traditional central processing unit (CPU) computing (Portegies Zwart 2020), to obtain the 3D distribution of $Q$ and $T_w$ with high efficiency. Feng et al. (2013) achieved an acceleration ratio of about 8 times by rewriting the model into a GPU compatible form for the magnetohydrodynamics (MHD) simulation of space weather. Caplan et al. (2019) accelerated the solar MHD code based on OpenAcc; the GPU version has about 3 times the efficiency of the CPU version, with a comparable cost level of hardware. Tassev & Savcheva (2017) have implemented the computation of QSLs with a GPU based on OpenCL, and achieved the efficiency of obtaining a representative 3D QSL map within a few hours.
The rest of this paper is arranged as follows: in Section 2, the algorithm and program structure is presented. In Section 3, we use the extrapolated potential field from a solar active region on 2010 October 16 19:00 UT (AR11112), an analytical quadrpole field, and an analytical field from Titov & Démoulin (1999; TD99 model) to demonstrate the method. In Section 4, we present detailed benchmarks and comparisons for different algorithms and computation architectures. In Section 5, we discuss and summarize the result.

2. Method

FastQSL is developed from the published source code by Liu et al. (2016), hereafter Code2016.\(^8\)

2.1. Calculation of Q

In the computation of Q, the essential and most computational consuming step is numerically deriving magnetic field lines by solving the equation

$$\frac{d \mathbf{r}(s)}{ds} = \frac{\mathbf{B}}{B},$$

where \(s\) is the arc length coordinate of a field line, and \(\mathbf{r}(s)\) is the coordinates function of the field line.

To accurately map the field-line footpoints on a surface, one must solve the field-line equation in high precision. In Code2016, Equation (7) is integrated by the classic RK4. We terminate the integration where it goes outside the data cube, and then move one step back to get the field-line coordinates at the boundary.

\(D\) in Equation (2) is derived from the changes in the mapped footpoint coordinates with respect to the footpoints of neighboring field lines. \(Q\) on a cross section can be calculated with the method 3 introduced in Pariat & Démoulin (2012; hereafter Method I), which introduces an auxiliary cross section \(S_0(x_0, y_0)\) to obtain

$$D_{1,2} = \begin{pmatrix} \frac{\partial x_2}{\partial x_0} & \frac{\partial x_2}{\partial y_0} & \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial y_1} \\ \frac{\partial y_2}{\partial x_0} & \frac{\partial y_2}{\partial y_0} & \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial y_1} \end{pmatrix},$$

and

$$\begin{pmatrix} \frac{\partial x_0}{\partial x_1} & \frac{\partial x_0}{\partial y_1} \\ \frac{\partial y_0}{\partial x_1} & \frac{\partial y_0}{\partial y_1} \end{pmatrix} = \begin{pmatrix} \frac{\partial x_0}{\partial x_1} & \frac{\partial x_0}{\partial y_1} \\ \frac{\partial y_0}{\partial x_1} & \frac{\partial y_0}{\partial y_1} \end{pmatrix}^{-1} = \begin{pmatrix} \frac{\partial x_0}{\partial x_1} & -\frac{\partial x_0}{\partial y_0} \\ -\frac{\partial y_0}{\partial x_1} & \frac{\partial y_0}{\partial y_1} \end{pmatrix} \left| B_{n,0}/B_{n,1} \right|,$$

where \(B_{n,0}\) is the component normal to \(S_0\), which has a similar form to Equation (4). Method I is used in Code2016.

Tassev & Savcheva (2017) published their code QSL Squasher\(^9\) that achieved a high efficiency to identify QSLs, and Scott et al. (2017) gave a detailed analysis of the code. Taking \(U, V\) as a pair of orthonormal unit vectors on \(S_0\), Scott et al. (2017) then proposed a method of obtaining \(Q\) without the information of neighboring mapping coordinates by solving

$$\frac{d \mathbf{r}(U, V)}{ds} = \left\{ \frac{\mathbf{B}}{B}, \frac{\mathbf{U} \cdot \nabla \mathbf{B}}{B}, \frac{\mathbf{V} \cdot \nabla \mathbf{B}}{B} \right\},$$

and they proved that

$$Q = \frac{\mathbf{U}_1 \cdot \mathbf{V}_2^2 + \mathbf{U}_2 \cdot \mathbf{V}_1^2 - 2(\mathbf{U}_1 \cdot \mathbf{V}_2)(\mathbf{U}_2 \cdot \mathbf{V}_1)}{(B_{n,0})^2/(B_{n,1} B_{n,2})}$$

is equivalent to Equation (3), where

$$\mathbf{U}_1 = U - \frac{U \cdot n}{B} \mathbf{B} \bigg|_{S_i};$$

the form is similar for \(\mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2\). In this paper, the method of Scott et al. (2017) based on Equation (11) is referred to as Method II.

An alternative set of codes for calculating QSLs has been published\(^10\) (hereafter CodeYang); the first version still followed Method I and was first applied in Yang et al. (2015). As of 2018 October, CodeYang adopted Method II.

Different selections of \(S_1, S_2\) will result in different values of \(Q\) even for the same start point on \(S_0\) (see the example in Section 4.1 of Titov 2007). In Code2016 and FastQSL, \(S_1, S_2\) by Method I are the boundaries where a field line terminates. Therefore Code2016 and FastQSL record these boundaries for every field line. If one locally rotates \(S_1, S_2\) to be perpendicular to the magnetic field line, Equation (3) and Equation (11) will give \(Q_\perp\) (Titov 2007). \(Q_\perp\) removes the projection effect on boundaries, and therefore quantifies the property of volume QSL more precisely than \(Q\) does. The reason that \(Q\) is still often used rather than \(Q_\perp\) is for its numerical simplicity. QSL Squasher and CodeYang set \(S_1, S_2\) to always be perpendicular to \(B\), therefore providing \(Q_\perp\) only.

Method I requires tracing at least four neighboring field lines for the central difference of footpoint coordinates, resulting in a numerical difficulty in cases where four field lines have different \(S_1, S_2\), which gives NaN in Code2016. Method I also has difficulty accurately applying the formula of \(Q_\perp\) of Titov (2007), especially on a polarity inversion line (PIL). Method II can give \(Q\) and \(Q_\perp\) directly without introducing the error of coordinate difference by tracing Equation (10) alone, but solving Equation (10) is less efficient than Equation (7) because of the need of calculating \(\nabla_B\) at every step. In addition, since \(Q\) changes sharply around a separatrix, the high-\(Q\) positions could be inside cells whose surrounding grids still have low values of \(Q\); then, these separatrix segments cannot be captured. In contrast, with Method I, Code2016 traces field lines at refined grids, and implements the central difference of the mapping coordinates in terms of refined grids for Equations (8) and (9); here the grid spacing is denoted as \(\delta\) in Pariat & Démoulin (2012). Consequently, a separatrix can be captured with refined grids, and characterized by an extremely high value of \(Q\).

Briefly speaking, Method I has the advantage of locating the position of thin QSLs, especially separatrices (except that \(S_1, S_2\) are not exactly the same for four neighboring field lines); Method II has the advantage of giving accurate values of \(Q\) and \(Q_\perp\). These characteristics are shown in Section 3 (Figures 2 and 4). Since \(Q\) is mostly used to locate the position of QSLs and separatrices, Method I still has its advantage. FastQSL provides the option of both methods.

---

\(^8\) http://staff.astc.edu.cn/~rfih/qfactor.html
\(^9\) https://bitbucket.org/tassev/qsl_squasher/src/hg/
\(^10\) https://github.com/Kai-E-Yang/QSL/
2.2. Magnetic Field at the Input

For Code2016 and FastQSL, the input magnetic field is assumed to be in Cartesian grids. Code2016 requires uniform grid spacings, while FastQSL additionally supports general stretched (but still rectilinear) grids. CodeYang and QSL Squasher can run in spherical coordinates; QSL Squasher can run in stretched grids.

We assume that the input magnetic field \( \mathbf{B} \) is known at every 3D Cartesian grid \((x_i, y_j, z_k)\). Then the \( \mathbf{B} \) at \( r = (x, y, z) \) in a cubic unit cell \([x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}]\) is interpolated by

\[
\mathbf{B}_{\text{interp}}(x, y, z) = \sum_{m=0}^{1} \sum_{n=0}^{1} \sum_{p=0}^{1} \omega_{x,m} \omega_{y,n} \omega_{z,p} \mathbf{B}(x_{i+m}, y_{j+n}, z_{k+p}),
\]

(13)

\[
\omega_{x,0} = \frac{x_{i+1} - x}{x_{i+1} - x_i},
\]

(14)

\[
\omega_{x,1} = 1 - \omega_{x,0},
\]

(15)

where \( \omega_{x,m}, \omega_{y,n}, \omega_{z,p} \) have similar forms to Equation (14) and Equation (15). For uniform grids, simply flooring \((x, y, z)\)/spacing is \((i, j, k)\) in Equation (13). While for stretched grids, we apply a binary search for the determination of \(i, j, k\), which is much more time-consuming than the flooring, and the final performance is reduced to 45%–80% (depending on settings at the input) by this determination.

2.3. Tracing Scheme

Code2016 and CodeYang utilizes the classic RK4 to solve Equation (7) and Equation (10). QSL Squasher was updated to version 2.0 from 2019 January, then the option of tracing scheme of Cash–Karp method is removed and only Euler integration is retained. Previous versions are not available online currently. All of them use uniform fixed step-size (hereafter step).

FastQSL updates the tracing scheme with the 3/8-rule RK4 (Kutta 1901), which introduces a smaller step error than the classic RK4, and additionally provides RKF45 (Fehlber 1969) for further acceleration. RKF45 calculates the difference between RK4 and RK5 of each step, \( \text{tol} \) is the maximum tolerated difference, and the unit of \( \text{tol} \) is the original grid spacing. If the difference is larger than \( \text{tol} \), which is a failed-trial step, then the step-size is adjusted to a smaller value and we repeat the tracing step from the same point. If the difference is smaller than \( \text{tol} \), RKF45 accepts this tracing step and then adjusts the step-size to a larger value according to the last difference and \( \text{tol} \). A smaller value of \( \text{tol} \) will result in a more precise output, but takes more computational resources.

If grids are stretched, we adopt a self-adaptive fashion of step and \( \text{tol} \) in cells of different shapes for a better performance. The scaling of step and \( \text{tol} \) in a cell are

\[
\text{scaling} = \sqrt{\left(\frac{B_i}{B_x} \right)^2 + \left(\frac{B_j}{B_y} \right)^2 + \left(\frac{B_k}{B_z} \right)^2},
\]

(16)

\[
\text{tol}_{\text{cell}} = \text{tol} \times \text{scaling},
\]

(17)

\[
\text{step}_{\text{cell}} = \text{step} \times \text{scaling}.
\]

(18)

tol and step here are dimensionless, \( \text{tol}_{\text{cell}} \) and \( \text{step}_{\text{cell}} \) that are actually applied in the cell have the same units as \( x_i, y_j, z_k \).

2.4. Code and Algorithm

Code2016 and FastQSL also provide \( T_w \) and field-line length, which are also extended to 3D by remaining constant along a field line, like Equation (5). Different from the \( Q \) calculation, a low level of (even without) refinement of \( T_w \) is good enough for data analysis. FastQSL additionally provides the coordinates of ending points of field lines, which had been used to locating flaring ribbons in an MHD simulation (Jiang et al. 2021), and can help the calculation of slip-squashing factors (Titov et al. 2009) if the flows at the boundaries are known.

FastQSL provides two sets of code. The first set is directly developed from Code2016 and still runs on IDL (the reliance of SolarSoftWare of Code2016 is removed)+Fortran. This set is optimized in many details. For example, it can be compiled by both gfortran and ifort while Code2016 is designed only for ifort.

The second set is accelerated by an NVIDIA GPU. The flowchart of the GPU program with Method I is shown in Figure 1; the differences from Method II are described in the caption of Figure 1. The program is based on Python 2.6 and CUDA/C13. The major input of this program is the data cube of the 3D magnetic field. The data is loaded with Scipy.io,14 which is capable of reading various file format of data (e.g., .sav, .mat, and unformatted). For the preparation of GPU computing, the field data, the parameters, and the coordinate array of points to be calculated are then transferred to GPU memory. The most computational-power consuming part is to trace magnetic field lines, for which the RKF45 solver is implemented in CUDA/C and compiled as a callable module TraceAllBline (as the green blocks shown in Figure 1) by Cupy.15 The compiled module calculates magnetic field lines and derives the footpoint mapping. Then the footprint mapping results are transferred back to host memory for the calculation of \( Q \). Also, with the magnetic field line computed in TraceAllBline, \( T_w \) can be simply derived with Equation (6). After the calculation, \( Q \) and \( T_w \) are visualized with matplotlib16 (for 2D) and pyvista17 (for 3D).18

3. Results

We apply FastQSL to three common scenarios of magnetic field analyzing: (1) a potential field extrapolated from a solar active region; (2) an analytical quadrupole field; and (3) a flux rope from a TD99 model. These magnetic fields are used to demonstrated and benchmark FastQSL.
Figure 1. The flowchart of GPU program with Method I in FastQSL. For Method II, the words in the upper part of the green blocks should be “TraceBlineScott.cu” and “TraceBlineScott,” and the block of “QCalc” is unnecessary.

Figure 2. Magnetic configuration of NOAA AR 11112 on 2010 October 16 at 19:00 UT. (a) PILs (cyan) and the footprints of bald patch separatrix (purple) superimposed on the magnetogram. (b)–(f) Map of QSL at the photosphere; the method and the factor of refinement of grids (refined grid spacing = 1/factor × original grid spacing) are denoted in the image title. The coordinates are the same as those in Chen et al. (2020). The green and the purple field lines present a double check of the existence of a bald patch in the black box in panel (b).
3.1. An Extrapolated Potential Field

The first test is done with the potential field extrapolated from an active region (NOAA AR 11112) on 2010 October 16 at 19:00 UT, which is the same field for analyzing “dome-plate QSL” in Chen et al. (2020). Strictly speaking, it should be “dome-plate separatrices” with a singular $Q$ since there is a null point (Titov et al. 2011; Scott et al. 2021). As shown by Figure 2(d), $Q_{\perp}$ cannot show any footprints of a bald patch separatrix (Titov et al. 1993; Titov & Démoulin 1999). For example, in the region bounded by the black box in Figure 2(b), there is a footprint of a bald patch separatrix on a PIL between the green and the purple field lines (the neighboring field lines at two sides of a PIL that is the footprints of a bald patch separatrix should depart with each other), which also satisfies ($B_x$, $B_y$, 0)·$\nabla B_z|_{\text{PIL}}>0$ (the purple lines in Figure 2(a)), while such topology is missing in Figure 2(d). As in the discussion above, the footprints of QSLs in Figures 2(d) and (e) are much thinner than those in Figure 2(b), even appearing discontinuous at the thinnest points. Figures 2(c) and (f) are calculated with original grids. Figure 2(c) keeps the continuity of footprints, while Figure 2(f) shows many more discontinuities than Figure 2(e), indicating that Method II requires a higher level of refinement for displaying continuous separatrices.

With the significant improvement in efficiency, the 3D distribution of $Q$ can be obtained with ease within the timescale of minutes. Figure 3 shows the 3D magnetic topologies above the magnetogram. As shown in Figure 3, the dome structure in Figure 4 of Chen et al. (2020) is well represented.

3.2. An Analytical Quadrapole Field

In the first case, $Q$ decays exponentially away from the null point (Pontin et al. 2016). There are still some remnants of the high-$Q$ region around separatrices that can be captured by Method II. An analytical quadrapole field can clearly present the shortage of Method II on capturing separatrices. The field is

$$B_{\text{quadrapole}}(r) = \sum_{i=1}^{4} q_i \frac{r - r_i}{|r - r_i|^3},$$

where $r_1 = (-1.5, 0, -0.5), r_2 = (-0.5, 0, -0.5), r_3 = (0.5, 0, -0.5), r_4 = (1.5, 0, -0.5)$ are the locations of four magnetic charges, and $\{q_1, q_2, q_3, q_4\} = \{1, -1, 1, -1\}$ are the strengths of the magnetic charges. This analytical field is uniformly discretized for FastQSL, and the grid spacing is 0.02. There are four Sun spots on the photosphere (Figure 4(a)), and the length of the field line can sharply jump at some places (Figures 4(b) and (e)), where there must be separatrices. Method I can fully capture all these separatrices (Figures 4(c) and (f)). While in Figure 4(g) that from Method II presents a blank because all calculated $Q$ are below 10 in the region of Figure 4(g); the case is the same if we plot $\log_{10}(Q_i)$. In Figure 4(c), in the area closed by the inner separatrices that are labeled with “in” or in the area outside of the outer separatrices that are labeled with “out,” considering the symmetry of the magnetic field, the field-line mapping (1) should be $x_2 = -x_1, y_2 = y_1$. According to Equation (3), the values of $Q$ in the area “in” and “out” should identically be 2. The real distribution of $Q$ around these separatrices should like the Dirac delta function. As the discussions of Section 2.1, since most refined grids are not on the zero-thickness separatrices, Method II cannot capture these separatrices (Figures 4(d) and (g)).

Another case is that $B = (x, -y, 0)$ has a similar distribution of $Q$. We set $x^2 + y^2 = 1$ as $S_1(\theta, z), S_2(\theta, z)$ of the $Q$ calculation (Figure 5), where $\theta$ is the radian measure. For $0 < \theta < \pi/2$ (the case is similar for $\pi/2 < \theta < \pi$), according to the symmetry
of the magnetic field, the mappings are \( \theta_2 = \pi/2 - \theta_1, z_2 = z_1 \), and all values of \( Q \) are 2 with Equation (3). Separatrices only appear at \( \theta = 0, \pi/2, \pi, 3\pi/2 \).

### 3.3. A TD99 Model

A TD99 model at the setting of \( R = 110 \, \text{Mm}, d = 34 \, \text{Mm}, L = 55 \, \text{Mm}, a = 49.4 \, \text{Mm}, I = 4 \times 10^{12} \, \text{A}, I_0 = 1.66 \times 10^{12} \, \text{A}, q = 10^{14} \, \text{Tm}^{-2} \) is also tested. This analytical field is uniformly discretized and the grid spacing is 3.04 Mm. This setting represents a hyperbolic flux tube (HFT; Titov et al. 2002) topology around the flux rope; the 3D structure of the HFT and the 3D distribution of the twist number are presented by Figure 6.

### 4. Benchmark

A benchmark is presented for comparing the efficiency of FastQSL with published codes (i.e., QSL Squasher, CodeYang, and Code2016). The quality of resultant images and the time consumed depend on the choices of \( \text{step of RK4} \) or \( \text{tol of RKF45} \) and the method. Images with different choices of parameters are shown in Figure 7. There is a marginal value of \( \text{step} \) or \( \text{tol} \) for the quality of the resultant image. For a specific row, comparing horizontally, i.e., comparing with those resultant images with any lower values, the image with the marginal value should not show any recognizable difference. And above the marginal value, the difference is recognizable. In Figure 7, columns are labeled with “A, B, C, D” to mark an image that is the ground truth, distinguishable, and unlikeliness. The marginal value depends on the smoothness of the field. For example, compared to column A and B, most areas in column C are satisfying, but some areas are not; one should check the quality of image to make decision. The image quality is not very sensitive to the value of \( \text{step or tol} \), which the efficiency is sensitive to. For some analysis that do not require a high quality, even column D can provide an acceptable result, and can achieve a very fast performance.

For Method I, if we fix the tracing parameter \( \text{step or tol} \), we find the angle \( \theta = \arcsin(|B_{n,0}|/B) \) between the magnetic field line and the cross section \( S_0 \) will affect image quality, with a smaller \( \theta \) giving poorer results. To mitigate this effect, our empirical formulas are

\[
\text{step}_1 |_{S_0} = \max ((\text{step}_1 \times |B_{n,0}/B|_{\text{step}_1 \text{min}}), \quad (20)
\]

\[
\text{tol}_1 |_{S_0} = \text{tol}_1 \times |B_{n,0}/B|_{1.5}, \quad (21)
\]

where \( \text{step}_1, \text{step}_1 \text{min} \) and \( \text{tol}_1 \) are constants and \( \max() \) is the operation of taking the maximum value. When field lines are tangent to \( S_0, B_{n,0} \to 0 \), introducing spurious high-\( Q \) structures. In order to avoid this artifact, when \( |B_{n,0}/B| < 0.05 \), we locally rotate \( S_0 \) so that it is perpendicular to the field line, and traces four neighboring field lines at the new cross section, and then calculate \( Q \). Therefore, the \( \text{step or tol} \) at the input of FastQSL is for the perpendicular case, then adjusted by Equations (20) or (21) for every field line. Since Code2016 fixes
For Method II, FastQSL always sets $S_0$ to the perpendicular case; this adjustment is not applied.

For Method II, there can be two strategies to calculate $\nabla B/B$. One prepares a 3D array of $\nabla B/B$ at the beginning by a second order finite difference; for example, the formula for uniform grids is

$$\frac{\partial B}{\partial x}(x_i, y_j, z_k) = \frac{1}{(x_{i+1} - x_{i-1})} \left[ \frac{B(x_{i+1}, y_j, z_k) - B(x_{i-1}, y_j, z_k)}{B(x_{i+1}, y_j, z_k) - B(x_{i-1}, y_j, z_k)} \right].$$

(22)

The form is similar for $\frac{\partial B}{\partial y}$, $\frac{\partial B}{\partial z}$, $\frac{\partial B}{\partial x}$, $\frac{\partial B}{\partial y}$, $\frac{\partial B}{\partial z}$. Then one does an interpolation that is similar to Equation (13). The other is to interpolate $B/B$ at the neighboring points of $r$ like $r = \pm (0.001, 0, 0)$, $r = \pm (0, 0.001, 0)$, $r = \pm (0, 0, 0.001)$, and then $\nabla B/B$ is given by central differences. The first way gives a smoother distribution than the second. As shown by the bottom two rows of Figure 7, the marginal tol by the first way is $10^{-2.9}$ while that by the second way is $10^{-4.2}$, which means the image quality by the first way is better with the same tol. The second way needs to calculate an additional six values of $B/B$, therefore, it is slower than the first way. But the first way asks for additional storage for the 3D array of $\nabla B/B$ (3 times the array of $B$ is occupied) while the second does not. CodeYang applies the second way. As the gradient of Equation (13) can be analytically given in every cell, QSL Squasher applies a skill that is mathematically similar to the second way, but the calculation of one value of $\nabla B/B$ consumes the similar duration as the first way. The first set of FastQSL uses the first way, while the second set applies the second way due to the limitation of GPU memory. Compared with Method I (the row of Code2016 of Figure 7), Method II (the top two rows of Figure 7) allows a much larger step.

Providing a completely impartial benchmark to all codes and both methods is extremely difficult. We just show the benchmark at the marginal values (column B of Figure 7), to make a sense of the time consumed. By using the TD99 model as an input, and calculating for the same region (the 3D region of Figure 6) and at the same resolution, the efficiency is measured by the count of calculated values of $Q$ per unit time.

The original CodeYang cannot accept a step > 1, because it extends only one ghost layer to boundaries. We modified the code to have 10 ghost layers for the benchmark.

QSL Squasher allows adaptive mesh refinement; the possible locations of QSL are quickly identified by the Field-line Length Edge (FLEEDGE) map, and then the code only calculates $Q_{\text{est}}$ at these locations. Tassev & Savcheva (2017) claimed an order-of-magnitude speed-up with adaptive refinements. We first do not apply adaptive refinements. We directly set the grid resolution for $Q_{\text{est}}$ that is the same as other panels of Figure 7; then we achieve the marginal step of 0.3, and use this setting for the benchmark (Table 1). A larger step can shrink the resulting QSLs more significantly (Figure 7), and even slightly affects them at the step of 0.3. We infer that this artifact comes from the relatively large step error of Euler integration, because our codes also have such shrinking if we change the tracing scheme to Euler integration. We also try the adaptive refinements. At the proper choices of the threshold of length jump for a refinement and the maximum times of refinements for giving a satisfying image that looks like column B of Figure 7, the performance by GPU is 25 kQ s$^{-1}$, which is surprisingly lower than 189 kQ s$^{-1}$ in Table 1, which is without the adaptive refinements. We guess that the gradient of field-line length in the thick QSL could be small still; then the adaptive refinements cannot improve the performance.

As shown by Table 1, Code2016 is faster than QSL Squasher and CodeYang. QSL Squasher is slowed down by Euler integration, double-precision floating-point format, adaptability for stretched grids, the way of calculating $\nabla B/B$, and other potential aspects. CodeYang is slowed down by double-precision floating-point format, the way of calculating $\nabla B/B$, and other potential aspects. Comparing to Code2016, FastQSL achieves a significant speed-up. For the first set of

Figure 6. 3D structures of the TD99 model. Left: the purple semitransparent surface is the contour of $Q = 5000$, different types of magnetic field line are rendered in different colors. Right: the distribution of the twist number.
FastQSL, compiling by ifort is slightly faster than by gfortran. Tracing by RKF45 is slightly faster than by RK4, but this comparison may be not true for all kinds of magnetic field. For a highly twisted field, many failed-trial steps could happen in that by RKF45, and then that by RK4 could be even faster. If traced by RK4, the efficiency by Method II is slightly faster than that by Method I; the comparison is reversed if traced by RKF45. For the second set of FastQSL that is optimized with GPU, it achieves the best efficiency by Method I. If $Q$ is calculated by Method II, since the second set uses the second way to calculate $\nabla_B^2$, it is even slower than the first set.

5. Discussion and Summary

Compared with Code2016, the computational efficiency increased by 24 times in this work. The increase in computational efficiency majorly comes from two aspects: new computing architecture and the improvement of algorithm.
Table 1
The Computational Efficiency of Different Methods, Measured as the Capability of Calculating $Q$ per Second

| Code          | Processor | Compiler | Method | Tracing scheme | Parameter | Performance |
|---------------|-----------|----------|--------|----------------|-----------|-------------|
| QSL Squasher  | CPU       | gfortran | II, 2  | Euler integration | step = 0.3 | 29 kQ s$^{-1}$ |
|               | GPU       | ifort    | I      | classic RK4     | step = 2.1 | 27 kQ s$^{-1}$ |
| CodeYang      | CPU       | ifort    | II, 1  | 3/8-rule RK4    | step = 0.2 | 191 kQ s$^{-1}$ |
| Code2016      | GPU       | CUDA/C   | II, 1  | RKF45           | tol = 10$^{-2.9}$ | 970 kQ s$^{-1}$ |
| FastQSL       | GPU       | ifort    | II      |                 | tol = 10$^{-3.2}$ | 1.1 MQ s$^{-1}$ |

Note. In this test, the CPU is Intel Core i9 10900K and the GPU is RTX 3070 OC. The version of gfortran is 9.4.0 and the version of ifort is 2021.6.0. The columns of “Processor,” “Compiler,” and “Performance” are only for the computation of QSL; others like input and output, preprocessing, and visualization are not involved. The numbers following II indicate the first or the second way of calculating $\sqrt{B}$ in $Q$. The column of “Parameter” shows the marginal value of a satisfied image in column B of Figure 7. QSL Squasher is tested with a smaller data cube of the same TD99 due to its high requirement of memory.

In the computational tasks of data inspection like QSL identification and quantification, GPU acceleration can improve the efficiency of interactive inspection and help to discover new features in data. In the computational tasks of simulation (Feng et al. 2013), the massive parallel computation with GPU can help explore more parameter space with a given amount of computational resource. Also, GPU computing is more environmentally friendly by reducing carbon emission (Stevens et al. 2020; Portegies Zwart 2020).

To summarize, we developed a reliable method of calculating the squashing factor $Q$ and twist number in data cubes of magnetic fields on Cartesian grids. This method can achieve unprecedented computation efficiency, with which one can obtain maps of $Q$ and the twist number within a few seconds for 2D input and a few minutes for 3D input. The high efficiency can benefit the analysis of magnetic topology, especially for the analysis of MHD simulations, which may require the computation of 3D $Q$ and the twist number in a time series.

J.C. thanks the constructive discussions with Guo, Yang, and Roger B. Scott, and acknowledges the support by the China Scholarship Council (No. 201706340140). The research was supported by the National Natural Science Foundation of China (42188810, 41974199, 41574167, 41774150, and 11925302), the B-type Strategic Priority Program of the Chinese Academy of Sciences (XDB41000000), and STELLAR project (952439).

ORCID iDs

PeiJin Zhang @ https://orcid.org/0000-0001-6855-5799
Jun Chen @ https://orcid.org/0000-0003-3060-0480
Rui Liu @ https://orcid.org/0000-0003-4618-4979
ChuanBing Wang @ https://orcid.org/0000-0001-6252-5580

References

Aulanier, G., Pariat, E., & Démoulin, P. 2005, A&A, 444, 961
Berger, M. A., & Prior, C. 2006, JPhA, 39, 8321
Caplan, R. M., Linker, J. A., Mikić, Z., et al. 2019, JPhCS, 1225, 012012
Chen, J., Liu, R., Liu, K., et al. 2020, ApJ, 890, 158
Démoulin, P., Bagala, L. G., Mandrini, C. H., Henoux, J. C., & Rovira, M. G. 1997, A&A, 325, 305
Démoulin, P., Priest, E., & Long, D. 1996, JGR, 101, 7631
Feng, X., Zhong, D., Xiang, C., & Zhang, Y. 2013, in ASP Conf. Ser. 474, Numerical Modeling of Space Plasma Flows (ASTRONUM2012), ed. N. V. Pogorelov et al. (San Francisco, CA: ASP), 131
Jiang, C., Chen, J., Duan, A., et al. 2021, FrP, 9, 575
Kutta, W. 1901, Z. Math. Phys., 46, 435
Liu, R., Kliem, B., Titov, V. S., et al. 2016, ApJ, 818, 148
Pariat, E., & Démoulin, P. 2012, A&A, 541, A78
Pontin, D., Galsgaard, K., & Démoulin, P. 2016, SoPh, 291, 1739
Pontin, D. I. 2011, AdSpR, 9, 575
Portegies Zwart, S. 2020, NatAs, 4, 819
Priest, E., & Forbes, T. 2000, Magnetic Reconnection: MHD Theory and Applications (Cambridge: Cambridge University Press)
Priest, E., & Démoulin, P. 1995, JGR, 100, 23443
Scott, R. B., Pontin, D. I., Antiochos, S. K., DeVore, C. R., & Wyper, P. F. 2021, AAS Meeting, 53, 2332.18
Scott, R. B., Pontin, D. I., & Hornig, G. 2017, ApJ, 848, 117
Scott, R. B., Pontin, D. I., & Hornig, G. 2017, ApJ, 848, 117
Stevens, A. R., Bellstedt, S., Elahi, P. J., & Murphy, M. T. 2020, NatAs, 4, 843
Tassev, S., & Savcheva, A. 2017, ApJ, 840, 89
Titov, V., & Démoulin, P. 1999, A&A, 351, 707
Titov, V. S. 2007, ApJ, 660, 863
Titov, V. S., Forbes, T. G., Priest, E. R., Mikić, Z., & Linker, J. A. 2009, ApJ, 693, 1029
Titov, V. S., Horning, G., & Démoulin, P. 2002, JGRA, 107, 1164
Titov, V. S., Mikić, Z., Linker, J. A., Lionello, R., & Antiochos, S. K. 2011, ApJ, 731, 111
Titov, V. S., Priest, E. R., & Démoulin, P. 1993, A&A, 276, 564
Yang, K., Guo, Y., & Ding, M. D. 2015, ApJ, 806, 171