`gauss_decomp`: A Python module for Gauss Decomposition of Cartesian Magnetic Field Maps

Abstract:

In solar physics and planetary sciences, magnetic fields are often measured via remote sensing (e.g., via the Zeeman effect) or *in situ* by satellite magnetometers. These measurements can reveal valuable information about the structure of the observed magnetic configuration. One powerful technique for analyzing magnetic fields measured on a surface (whether a closed surface, or a portion of one) is Gauss's separation method (e.g., Olsen et al. 2010), which decomposes the measured magnetic field on the surface into three separate components, based upon its source currents' locations: (i) interior to the surface; (ii) across the surface; and (iii) exterior to the surface. Here, we describe a Python module, `gauss_decomp`, that can be used to apply Gauss's method in Cartesian geometry to such measurements, focusing on its application to solar photospheric magnetograms. We present the equations of Gauss's method in Cartesian geometry, describing our numerical approach for computing solutions at each step, i.e., which subroutines and code blocks are used with which equations. Gauss's method relies upon Fourier transforms (FTs), and `gauss_decomp` uses fast Fourier transforms (FFTs) from `NumPy` for this, and for computing spatial derivatives and solving Poisson equations. Finally, we provide both example input data and example results from an observed magnetogram, using FITS I/O file utilities in `astropy`. The data used in our example, as well as our example results, are available for download with the module. We also provide a subroutine for simple tests of the method using point dipoles placed at arbitrary locations.

Scientific Motivation:

Electric currents flowing in the solar corona are believed to supply the magnetic energy that is suddenly released in solar flares and coronal mass ejections (CMEs). Although very important, coronal currents are not directly observable, so properties of AR coronal currents are usually inferred via modeling based upon observations. Recently, Shuck et al. (2022) applied Gauss's separation method to photospheric magnetograms to determine which parts of the observed photospheric field are due to currents flowing (i) across the photosphere in the solar interior, (ii) in the solar interior, and (iii) above the photosphere, in the chromosphere and corona. This approach shows tremendous promise for improving our understanding of the nature of currents on the Sun, and their roles in various phenomena, including coronal heating and flares/CMEs.

<u>Terse Review of the Method</u>:

To make version control as transparent as possible, the `gauss_decomp` file available from GitHub contains in its filename the date of its last modification as a YYMMDD suffix. When referring to the module in the text below, we omit this suffix. We recommend creating a copy of the `gauss_decomp` module without this suffix for use in your routines, so they do not refer to the source code from a specific date.

In the `gauss_decomp` module, the Gaussian separation is peformed by the `gauss_sep` subroutine, which takes as its input the three components of the observed photospheric magnetic field: $B_x(x,y,0)$, $B_y(x,y,0)$, $B_z(x,y,0)$. It employs several other subroutines, including 2D Fourier differentiation routines for derivatives in x & y (`xderiv_fft_2d` and `yderiv_fft_2d`), a 2D Fourier Poisson solver (`poisson_fft_2d`), and the `ptd_fft_2d` subroutine, introduced below.

<u>Step 1: Poloidal-Toroidal Decomposition (PTD)</u>

We begin by decomposing the three components of the 2D, horizontal vector photospheric magnetic field into toroidal & poloidal parts (e.g., Chandrasekhar 1947, Fisher et al 2010). This poloidal-toroidal decomposition (PTD) is accomplished within `gauss_decomp` by the `ptd_fft_2d` subroutine.

We first isolate the toroidal component of the photospheric magnetic field, $\mathbf{B}_T(x,y,0)$, which is due to currents across the photosphere, i.e., normal to it. Such currents produce a magnetic field that is purely horizontal at the photosphere. This component has non-zero surface curl (or circulation), so can be represented in terms of a stream function,

$$\mathbf{B}_T(x,y,0) = \nabla_h \times T(x, y, 0)\hat{\mathbf{z}} = -\hat{\mathbf{z}} \times \nabla_h T(x, y, 0) \ . \quad (1)$$

In Cartesian geometry, with the photosphere at z = 0, this stream function, T, can be found by solving the 2D Poisson equation derived from Ampère's law,

$$\mu_0 J_z(x,y,0) = [\, \nabla_h \times \mathbf{B}_h(x,y,0)\,]\cdot\hat{\mathbf{z}} = -\nabla_h^2 T(x,y,0)\,, \qquad (2)$$

where $\nabla_h = (\partial_x, \partial_y, 0)$. We refer to the operation $[\nabla_h \times \mathbf{B}_h]\cdot\hat{\mathbf{z}}$ as taking the 2D curl, or horizontal curl, or surface curl of the operand $\mathbf{B}_h$, and $\hat{\mathbf{z}}\cdot[\nabla_h \times \ldots]$ as the surface curl operator.

`gauss_decomp` takes input of the observed $\mathbf{B}(x,y,0)$, computes $J_z(x,y,0)$, and 2D fast Fourier transform (FFTs) in `NumPy` are used to solve equation (2) for T. The toroidal component of the photospheric magnetic field, $\mathbf{B}_T(x,y,0)$, is then computed from equation (1).

Within the `ptd_fft_2d` routine, this is accomplished via

```
# source for T is 2D curl of B_horiz
#------------------------------------
dbxdy = yderiv_fft_2d(bx, yrange)
dbydx = xderiv_fft_2d(by, xrange)
ptd["T"] = poisson_fft_2d( -( dbydx - dbxdy), xrange, yrange)
```

Here, (`xrange,yrange`) are the sizes of the (x, y) dimensions, respectively, of the 2D input arrays, in pixels.

Next, we determine the poloidal part of the photospheric magnetic field, $\mathbf{B}_P(x,y,0)$. In the PTD representation, $\mathbf{B}_P(x,y,0) = \nabla \times (\nabla \times P(x, y, 0)\hat{\mathbf{z}}) = \nabla_h[\partial_z P(x, y, 0)] - \nabla_h^2 P(x, y, 0)\hat{\mathbf{z}}$, where P(x, y, 0) is the poloidal potential and $\partial_z P(x, y, 0)$ is its derivative normal to the surface. These are determined independently from the data, but are not independent of each other. The vertical photospheric field is purely poloidal,

$$B_{P,z}(x,y,0) = B_z(x,y,0) = -\nabla_h^2 P(x, y, 0)\,, \qquad (3)$$

so, by solving another 2D Poisson equation, the observed $B_z(x,y,0)$ determines P(x,y,0) to within a harmonic function. The horizontal part of $\mathbf{B}_P(x,y,0)$ is

$$\mathbf{B}_{P,h}(x,y,0) = \nabla_h[\partial_z P(x, y, 0)]\,. \qquad (4)$$

The horizontal divergence of the horizontal photospheric field determines $\partial_z P(x,y,0)$ to within a harmonic function, again by solving a 2D Poisson equation,

$$\nabla_h \cdot \mathbf{B}_{P,h}(x,y,0) = \nabla^2[\partial_z P(x, y, 0)]. \tag{5}$$

Within the `ptd_fft_2d` routine, solving for $\partial_z P(x,y,0)$ and $P(x,y,0)$ is done via

```
# source for dPdz is 2D divergence of B_horiz
#-----------------------------------------------
dbxdx = xderiv_fft_2d(bx, xrange)
dbydy = yderiv_fft_2d(by, yrange)
ptd["dPdz"] = poisson_fft_2d( ( dbxdx + dbydy), xrange, yrange)

# source for P is B_z
#-----------------------------------------------
ptd["P"] = poisson_fft_2d( -bz, xrange, yrange)
```

Once $T(x,y,0)$ and $\partial_z P(x,y,0)$ have been found, the toroidal and poloidal parts of $\mathbf{B}_h(x,y,0)$ are found via

```
# Use T & P potentials to compute toroidal & poloidal parts of B_hor
#-----------------------------------------------
ptd["btx"] =  yderiv_fft_2d(ptd["T"], yrange)
ptd["bty"] = -xderiv_fft_2d(ptd["T"], xrange)
ptd["bpx"] =  xderiv_fft_2d(ptd["dPdz"], xrange)
ptd["bpy"] =  yderiv_fft_2d(ptd["dPdz"], yrange)
```

Note that decomposing $\mathbf{B}_h(x,y,0)$ into $\mathbf{B}_T(x,y,0)$ and $\mathbf{B}_P(x,y,0)$ is not unambiguous. First, any mean field, $\mathbf{B}_{mean}(x,y,0)$, on a finite domain can be toroidal, poloidal, or a mixture of these. Further, any Fourier approach cannot recover such "DC" components of the total field. In the `ptd_fft_2d` routine, the mean components of the input field are computed (by `NumPy as np`) and returned to the user as

```
# Calculating mean field components
# (not recovered by FFT methods)
#---------------------------------
ptd["bx_mean"] = np.mean(bx)
ptd["by_mean"] = np.mean(by)
ptd["bz_mean"] = np.mean(bz)
```

In addition, conjugate harmonic functions can be used to add equal and opposite fields to $\mathbf{B}_T(x,y,0)$ and $\mathbf{B}_P(x,y,0)$, a topic beyond the scope of this document; see the Welsch et al. (2025) for more. The magnitudes of these potentials are limited by magnetic field strengths on the x & y boundaries of the 2D domain, meaning they should be insignificant for an isolated, active-region field surrounded by sizeable areas without a significant, spatially coherent field.

Step 2: Gaussian Separation

The aim of Gaussian Separation is to partition the total magnetic field on a surface into components based upon source currents' locations: (i) $\mathbf{B}^<(x,y,0)$ from horizontal currents $\mathbf{J}_h^<(x,y,0)$ interior to the surface (in $z < 0$); (ii) $\mathbf{B}_T(x,y,0)$ from currents $\mathbf{J}_z(x,y,0)$ across the surface; and (iii) $\mathbf{B}^>(x,y,0)$ from horizontal currents $\mathbf{J}_h^>(x,y,0)$ exterior to the surface (in $z > 0$). (We assume there is not a horizontal current sheet flowing at the photosphere – i.e., no singular current density at $z = 0$. We assume any well-behaved horizontal current density $\mathbf{J}_h$ at $z = 0$ can be partitioned into currents flowing differentially above and differentially below the photosphere.)

Having already found $\mathbf{B}_T(x,y,0)$ via the decomposition of $\mathbf{B}_h(x,y,0)$ into poloidal and toroidal parts, we now proceed to partition the horizontal and vertical parts of $\mathbf{B}_P(x,y,0)$, which are entirely due to currents flowing either below or above the photosphere, into $\mathbf{B}^<(x,y,0)$ and $\mathbf{B}^>(x,y,0)$. In general, $\mathbf{B}_P(x,y,0)$ is comprised of contributions from both. Since, per the Biot-Savart law, the total field is a linear sum of the fields produced by all currents,

$$\mathbf{B}_{P,h}(x,y,0) = \mathbf{B}_h^<(x,y,0) + \mathbf{B}_h^>(x,y,0) \quad (6)$$

$$B_z(x,y,0) = B_z^<(x,y,0) + B_z^>(x,y,0) \quad (7)$$

Because the source currents for $\mathbf{B}^<(x,y,0)$ and $\mathbf{B}^>(x,y,0)$ do not flow *at* the photosphere, both fields are potential there:

$$\mathbf{B}^<(x,y,0) = -\nabla\psi^<(x,y,0) \quad (8)$$

$$\mathbf{B}^>(x,y,0) = -\nabla\psi^>(x,y,0) \quad (9)$$

Because $\mathbf{B}(x,y,z)$ is divergence-free, the divergences of $\nabla\psi^<$ and $\nabla\psi^>$ vanish, i.e., both potentials satisfy 3D Laplace equations, $\nabla^2\psi^< = 0$ and $\nabla^2\psi^> = 0$, in their respective domains ($z \geq 0$ for $\nabla\psi^<$, $z \leq 0$ for $\nabla\psi^>$, respectively). Accordingly, expressed in terms of their Fourier transforms, these potentials obey

$$\psi^{<(>)} = \int \Psi\,(k_x, k_y) \exp[i\,(k_x\, x + k_y\, y) +(-)\,(k_h\, z)]\, dk_x\, dk_y\,, \quad (10)$$

with $k_h = (k_x^2 + k_y^2)^{1/2}$ and $\Psi^{<(>)}(k_x, k_y)$ being the Fourier transform (FT) of $\psi^{<(>)}$. (The differing signs on the $(k_h\, z)$ term in the exponential arise from the different asymptotic behaviors of $\psi^{<(>)}$

away from their source currents: $\psi^<$ decays as $z \to +\infty$, and $\psi^>$ decays as $z \to -\infty$.) Equations (6) and (7) provide Dirichlet and Neumann boundary conditions (BCs) for $\psi^<$ and $\psi^>$,

$$\nabla_h \cdot \mathbf{B}_h = -\nabla_h^2 \psi^< - \nabla_h^2 \psi^> \qquad (11)$$

$$B_z = -\partial_z \psi^< - \partial_z \psi^> , \qquad (12)$$

respectively, which constrain the FTs $\Psi^{<(>)}$. In the Fourier domain, with the spatial operators $\nabla_h \to \mathbf{ik}_h$ and $\partial_z \to \pm k_h$, and $[\mathcal{B}_x(k_x, k_y), \mathcal{B}_y(k_x, k_y), \mathcal{B}_z(k_x, k_y)]$ denoting the three FT's of the three respective components of the photospheric vector magnetic field $\mathbf{B}(x, y, 0)$, the Dirichlet and Neumann BCs imply

$$\mathrm{i}\mathbf{k}_h \cdot \boldsymbol{\mathcal{B}}_h(k_x, k_y) = + k_h^2 (\Psi^< + \Psi^>) \qquad (13)$$

$$\mathcal{B}_z(k_x, k_y) = + k_h (\Psi^< - \Psi^>), \qquad (14)$$

which implies

$$\Psi^{<(>)}(k_x, k_y) = \tfrac{1}{2} [(\mathrm{i}\mathbf{k}_h \cdot \boldsymbol{\mathcal{B}}_h(k_x, k_y)) k_h^{-2} +(-) \mathcal{B}_z(k_x, k_y) k_h^{-1}]. \qquad (15)$$

In the `gauss_sep` routine, equation (13) is not used directly; instead, equation (11) is first solved for the sum $(\psi^< + \psi^>)$, which is then Fourier transformed to find the sum $(\Psi^< + \Psi^>)$. This is done via

```
phi_dirichlet = poisson_fft_2d(-(dbxdx + dbydy), xrange,
 yrange)

dirispect = np.fft.fft2(phi_dirichlet)
```

Solving (14), to get $(\Psi^< - \Psi^>)$, requires defining 2D wavenumber arrays, which are computed from the size of the 2D input magnetic field arrays,

```
dx = xrange/nxbx
dy = yrange/nybx

xfreqs = np.fft.fftfreq(nxbx, d = dx)
yfreqs = np.fft.fftfreq(nybx, d = dy)

# Construct wave number arrays
xfreqs_2d,yfreqs_2d = np.meshgrid(xfreqs,yfreqs)

# Radius in wavenumber space
hfreqs = np.sqrt(xfreqs_2d**2 + yfreqs_2d**2)
```

Note that these arrays contain no information about length scales in physical units, i.e., wavenumbers have units of inverse pixels. Using these, the Fourier spectrum needed for the Neumann boundary condition in (14) is found via

```
# source for P is B_z: P(kx,ky) = B_z(kx,ky)/(2*pi*hfreqs)
#----------------------------------------------------------

# Compute 1/hfreqs -- BEWARE: element [0,0] is zero
inv_hfreqs = np.zeros(hfreqs.shape)
nz_indices = np.nonzero(hfreqs)
inv_hfreqs[nz_indices] =   1./hfreqs[nz_indices]


bzspect = np.fft.fft2(bz)
neumspect = inv_hfreqs * bzspect/(2*np.pi)
```

Having found ($\Psi^{<} + \Psi^{>}$, `dirispect`) and ($\Psi^{<} - \Psi^{>}$, `neumspect`), the independent spectra of $\Psi^{<}$ and $\Psi^{>}$ are then found,

```
psigtspect = 0.5*(dirispect-neumspect)
psiltspect = 0.5*(dirispect+neumspect)
```

Remark: Because `gauss_decomp` is primarily used with 2D image data, subroutines in the `gauss_decomp` module typically accept input in column-major order (column, row; or x, y) order, not the row-major order that is the default for many Python routines. Consequently, users should exercise care when interfacing between other Python routines and `gauss_decomp` routines.

<u>**Example Results from Solar Observations:**</u>

For a user to verify the performance of the `gauss_decomp` routines on their computer, we have included sample input data, in FITS format, and the corresponding output Gaussian decomposition from `gauss_decomp`. These sample data, in the file `Hinode_SOT_SP_20061212_2030.fits`, were used in a heavily cited study of coronal magnetic field extrapolation by Schrjiver et al. (2008), and are described in detail there.

The Python routine `GaussSeperationExample.py`, available for download with the `gauss_decomp` software, uses FITS file I/O utilities from `astropy` to read in the Hinode sample magnetogram, perform the Gaussian decomposition, and write the decomposed magnetic field components to the file `gauss_decomp_trial_output.fits`. This file's "trial" contents should be numerically equivalent, to machine precision, with the contents of example data we provide in the file `gauss_decomp_example_output.fits`, which must be present in the same directory as `GaussSeperationExample.py`. By default, `GaussSeperationExample.py` prints out the maximum absolute differences between the contents the user-derived field components and the field components in `gauss_decomp_example_output.fits`.

Numerical Tests with Analytic Sources:

Users can also explore the nature of the Gaussian separation using another test routine, `HdipExample.py`, supplied with `gauss_decomp` package. `HdipExample` is a main-level Python program that creates a synthetic magnetogram containing vector magnetic fields due to currents above and below the magnetogram surface by calling the `hdip` subroutine in the `gauss_decomp` package. Each call to `hdip` computes the vector magnetic field in a plane a distance z_0 above or below horizontal a dipole (i.e., a dipole with its axis parallel to the plane), on a grid defined by two 1D, input coordinate arrays. Gaussian separation is then applied to the synthetic magnetogram, and the fields separated by `gauss_decomp` are then graphically compared to the input dipoles' fields in several images and scatter plots. By editing `HdipExample`, users can vary the input dipoles' amplitudes, locations, and orientations, or even add more dipoles.

Feedback & Bug Reports

If you have comments, especially suggestions for improvements or bug reports, please email Brian Welsch at welschb "at" uwgb.edu.

References:

Olsen N., Glassmeier, K. H., Jia, X. 2010, "Separation of the Magnetic Field into External and Internal Parts," Space Science Reviews, Volume 152, Issue 1-4, pp. 135-157
https://link.springer.com/article/10.1007/s11214-009-9563-0

Schrijver *et al.*, 2008, “Nonlinear Force-free Field Modeling of a Solar Active Region around the Time of a Major Flare and Coronal Mass Ejection,” ApJ, v. 675, Issue 2, pp. 1637-1644, https://arxiv.org/abs/0712.0023

Schuck, P. W., Linton, M. G., Knizhnik, K. J., & Leake, J. E. 2022, “On the Origin of the Photospheric Magnetic Field,” ApJ, v. 936, id.94, https://iopscience.iop.org/article/10.3847/1538-4357/ac739a

Welsch, B. T., Schuck, P. W., Linton, M. G. 2025, “The Photospheric Imprints of Coronal Currents,” in preparation, https://arxiv.org/abs/2211.01911