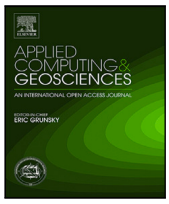
Applied Computing and Geosciences 25 (2025) 100222

Contents lists available at ScienceDirect

Applied Computing and Geosciences

journal homepage: www.sciencedirect.com/journal/applied-computing-and-geosciences



Pymaginverse: A python package for global geomagnetic field modeling

Frenk Out [a,*], Maximilian Schanner [b,c], Liz van Grinsven [a], Monika Korte [c], Lennart V. de Groot [a]

[a] Paleomagnetic laboratory Fort Hoofddijk, Department of Earth Sciences, Utrecht University, Budapestlaan 17, 3584 CD Utrecht, The Netherlands
[b] Institute of applied mathematics, Potsdam University, Karl-Liebknecht-Str. 24-25, D-14476 Potsdam, Germany
[c] Helmholtz Centre Potsdam, Deutsches GeoForschungsZentrum GFZ, Telegrafenberg, D-14473 Potsdam, Germany

ARTICLE INFO

Dataset link: https://github.com/outfrenk/pymaginverse

Keywords:
RLS geomagnetic models
Geomagia
Paleomagnetism
Open research

ABSTRACT

Data-based geomagnetic models are key for mapping the global field, predicting the movement of magnetic poles, understanding the complex processes happening in the outer core, and describing the global expression of magnetic field reversals. There exists a wide range of models, which differ in a priori assumptions and methods for spatio-temporal interpolation. A frequently used modeling procedure is based on regularized least squares (RLS) spherical harmonic analysis, which has been used since the 1980s. The first version of this algorithm has been written in Fortran and inspired many different research groups to produce versions of the algorithm in other programming languages, either published open-access or only accessible within the institute. To open up the research field and allow for reproducibility of results between existing versions, we provide a user-friendly open-source Python version of this popular algorithm. We complement this method with an overview on background literature – concerning Maxwells equations, spherical harmonics, cubic B-Splines, and regularization – that forms the basis for RLS geomagnetic models. We included six spatial and two temporal damping methods from literature to further smooth the magnetic field in space and time. Computational resources are kept to a minimum by employing the banded structure of the normal equations involved and incorporating C-code (with Cython) for matrix formation, enabling a massive speed-up. This ensures that the algorithm can be executed on a simple laptop, and is as fast as its Fortran predecessor. Four tutorials with ample examples show how to employ the new lightweight and quick algorithm. With this properly documented open-source Python algorithm, we have the intention to encourage current and new users to employ and further develop the method.

1. Introduction

The geomagnetic field is generated by currents in the fluid outer core of the Earth. The interaction of all these currents results in a mostly dipolar magnetic field at the Earth's surface. This field protects the atmosphere against charged particles from the sun and shields electric circuits from disturbances by the solar wind. To understand the geomagnetic field and predict its behavior in a global sense, geomagnetic field models are paramount. Geomagnetic models integrate magnetic data from around the world into one time-dependent, global picture. Geomagnetic models are applied for many different purposes: from predicting the movement of the geomagnetic poles to studying the global surface expression of magnetic field reversals (e.g. Finlay et al., 2020; Mahgoub et al., 2023b). Additionally, geomagnetic field models function as a basis for better understanding the Earth's fluid outer core.

Earth's magnetic field is modeled by interpolating geomagnetic data in space and time while applying certain constraints (regularization). It is the type of interpolation used and constrains applied that lead to different models (Bloxham and Jackson, 1992). We focus on modeling only the magnetic core field as function of historical, archeomagnetic, and paleomagnetic datasets. Due to the type of data, we do not include methods for separating crustal or external fields, as is necessary when modeling satellite or observatory data, for example for constructing the IGRF (Alken et al., 2021). The modeling algorithm in focus is based on regularized least squares (RLS) techniques. RLS models require geomagnetic data, preferably evenly distributed around the globe, and spatial and temporal smoothness constraints to invert for a finite set of Gauss coefficients that describe the geometry of the magnetic field by dividing the field into specifically structured field configurations, such as a dipolar and quadrupolar field. Due to the specific numerical structure of the RLS approach, efficient inversion algorithms have been developed. Recently, statistical approaches mostly based on Gaussian processes have been employed to construct distributions over geomagnetic field

* Corresponding author.
E-mail address: f.out@uu.nl (F. Out).

https://doi.org/10.1016/j.acags.2025.100222
Received 29 August 2024; Received in revised form 20 December 2024; Accepted 16 January 2025
Available online 23 January 2025

models (Leonhardt and Fabian, 2007; Hellio and Gillet, 2018; Arneitz et al., 2019; Mauerberger et al., 2020; Nilsson and Suttie, 2021). The motivation behind these statistical approaches is the assessment of uncertainties in the models, due to uncertainties in the data, in the assumptions, and in the modeling itself. Statistical models can be computationally intense, although current developments are speeding up certain types of models (Schanner et al., 2022). Originally, the RLS techniques were developed in a Bayesian framework, and can be considered statistical models as well. In this sense, regularization employed in RLS techniques specifies a prior distribution. It is also possible to construct posterior distributions for the RLS models. However, over the years this perspective has been lost and only the maximum posterior point estimate, which coincides with the RLS solution, has been considered (Bloxham and Jackson, 1992; Jackson et al., 2000; Korte and Constable, 2003). The difference between the recent statistical models and the RLS models is therefore more in the philosophy of how they are used and interpreted than in a strict methodological sense.

RLS models have a widespread use in modeling the geomagnetic field. The method has been established for quite some time (Shure et al., 1982; Bloxham and Jackson, 1992; Holme and Bloxham, 1996; Jackson et al., 2000) and is frequently used for modeling historical-to-millennial scale fields and snapshots thereof (e.g. Bloxham and Jackson, 1992; Constable et al., 2000; Korte and Constable, 2003; Korte et al., 2009; Licht et al., 2013; Pavón-Carrasco et al., 2014; Nilsson et al., 2014; Panovska et al., 2015, 2018; Finlay et al., 2020) and magnetic reversals (e.g. Mahgoub et al., 2023b). Gauss coefficients that vary smoothly in time, expressing the changing geometry of the field, are obtained by means of cubic B-Splines (De Boor, 1978; Constable and Parker, 1988). Unfortunately, the inverse problem to obtain these Gauss coefficients is badly conditioned due to poor data coverage. Therefore, to find a physically reasonable model, the problem is regularized by additional spatial and temporal constraints. These constraints can range from minimizing heat flux through the core-mantle boundary (CMB) to minimizing the acceleration of the vertical component of the magnetic field at the CMB (Gubbins, 1975; Korte and Constable, 2003). These extra constraints are an important reason for RLS models to output unrealistically low uncertainties for Gauss coefficients.

Nevertheless, the relatively easy interpretation of Gauss coefficients in terms of field geometry still make RLS models widely used in the geomagnetic community (e.g. Mahgoub et al., 2023b). The RLS algorithm requires fast numerical calculations to process global geomagnetic datasets (between 10.000 and 100.000 records). This first version of this algorithm has been written in the Fortran programming language, famous for its high speed with numerical calculations, where compiling geomagnetic data into a model is a matter of minutes. Unfortunately, Fortran is infamous for its minimal error handling and, more importantly, these Fortran algorithms have sparse documentation. However, this Fortran algorithm was distributed in many different versions to other groups who changed it into their own version. Since a central version of the algorithms does not exist, it is difficult to reproduce results of a specific version with another version of the same base algorithm (see discussion in Korte et al., 2009). Furthermore, finding, understanding, executing, and checking the different versions can be a time-consuming endeavor, potentially discouraging new users to either generate RLS geomagnetic models or use it as a prior for statistical models.

To tackle these issues, we provide a completely open-source repository for the RLS algorithm and present the accompanying theory in this manuscript. With the help of abundant literature on the RLS technique (Shure et al., 1982; Gubbins and Bloxham, 1985; Bloxham, 1987; Constable and Parker, 1988; Bloxham and Jackson, 1992; Korte and Constable, 2003; Korte et al., 2009; Mahgoub et al., 2023b) and the help of its users, we have rewritten the Fortran version of the RLS geomagnetic algorithm into a Python 3 library. We have chosen Python because it is a modern language that puts emphasis on code readability. The created library (Out and Schanner, 2024) is easy to use for anybody with basic Python skills, while retaining the speed and functionalities of the original Fortran algorithm. For this purpose, we restructured the algorithm in a more logical way; introducing separate classes for loading data and performing calculations. To further help the user, we have included tutorials to introduce all aspects of the algorithm. This enables future users to easily modify and develop the library even further. After providing an overview of the theory and implementation of the algorithm, we will show functionality of the algorithm and compare the efficiency of the library against its Fortran predecessor.

2. Theoretical background

Geomagnetic field modeling allows for interpolation of paleomagnetic data in both space and time, while accounting for measurement errors. The temporal interpolation is obtained by cubic B-splines, which is discussed in Section 2.2; the spatial interpolation is obtained with spherical harmonics, based on two of Maxwell's equations:

$$\begin{aligned} &\vec{\nabla}\cdot\vec{B} = 0 \text{ (divergence free)} \\ &\vec{\nabla}\times\vec{B} = \mu_0(\vec{J} + \epsilon_0\frac{\partial\vec{E}}{\partial t}) = \vec{0} \text{ (source free)}, \end{aligned} \tag{1}$$

with $\vec{B}$ the magnetic field, μ_0 the vacuum permeability ($4\pi\times10^{-7}$ H/m), $\vec{J}$ the electric current density, ϵ_0 the vacuum permittivity, and $\vec{E}$ the electric field. We apply Helmholtz decomposition to the magnetic field $\vec{B}$, separating it into a scalar and a vector field: $\vec{B} = -\vec{\nabla}W + \vec{\nabla}\times\vec{F}$. Since we evaluate the field outside the outer core (source free assumption), we only retain the scalar part. After combining this scalar field with the divergence free assumption, we acquire Laplace's equation ($\nabla^2 W = 0$), and solve that in a spherical coordinates system (derivation can be found in e.g. Lowrie, 2011). Furthermore, external magnetic sources (ionosphere) are ignored, because their contribution to the total magnetic signal is negligible compared to internal sources in the historical and longer time context; the scalar potential W is then given by:

$$W(r,\theta,\phi) = R\sum_{\ell=1}^{\infty}\sum_{m=0}^{\ell}\left(\frac{R}{r}\right)^{\ell+1}(g_\ell^m\cos(m\phi) + h_\ell^m\sin(m\phi))P_\ell^m(\cos(\theta)), \tag{2}$$

with R the radius of the Earth (6371.2 km) and r, θ, and ϕ the radius, co-latitude, and longitude, respectively, of the location where the potential is evaluated. $P_\ell^m(\cos(\theta))$ are the Schmidt semi-normalized associated Legendre polynomials of degree ℓ and order m. Lastly, the variables g_ℓ^m and h_ℓ^m are the Gauss coefficients (g_1^0, g_1^1, h_1^1, g_2^0, g_2^1, h_2^1, etc.), which describe the global geomagnetic field geometry.

2.1. Modeling equations

Paleomagnetic information often include inclination, declination, and intensity data, which are non-linearly related to the magnetic potential. Before going into detail on this non-linearity, we first state how the components of the magnetic field vector relate linearly to the Gauss coefficients. The vector consists of a northern (B_x), eastern (B_y), and vertical (B_z) component in a geocentric dataframe (although most geomagnetic data comes in a geodetic dataframe, see Section 3.1). These three components are obtained by taking the derivative of the scalar potential (Eq. (2)) with respect to the co-latitude, longitude, and

$$\begin{bmatrix} B_x^1 \\ B_y^1 \\ B_z^1 \\ \vdots \\ B_z^Q \end{bmatrix} = \begin{bmatrix} -\sin(\theta_1) & \cos(\phi_1)\cos(\theta_1) & \sin(\phi_1)\cos(\theta_1) & \cdots & \sin(L\phi_1)\frac{d(P_L^L(\cos(\theta)))}{d\theta}\Big|_{\theta=\theta_1} \\ 0 & \sin(\phi_1) & -\cos(\phi_1) & \cdots & -\frac{L\cos(L\phi_1)}{\sin(\theta_1)} P_L^L(\cos(\theta_1)) \\ -2\cos(\theta_1) & -2\cos(\phi_1)\sin(\theta_1) & -2\sin(\phi_1)\sin(\theta_1) & \cdots & -(L+1)\sin(L\phi_1)P_L^L(\cos(\theta_1)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -2\cos(\theta_Q) & -2\cos(\phi_Q)\sin(\theta_Q) & -2\sin(\phi_Q)\sin(\theta_Q) & \cdots & -(L+1)\sin(L\phi_Q)P_L^L(\cos(\theta_Q)) \end{bmatrix} \begin{bmatrix} g_1^0 \\ g_1^1 \\ h_1^1 \\ \vdots \\ h_L^L \end{bmatrix} \tag{4}$$

Box I.

radius, respectively:

$$\begin{aligned} B_x &= -B_\theta = \frac{1}{r}\frac{\partial W}{\partial \theta} \\ &= \sum_{\ell=1}^{\infty}\sum_{m=0}^{\ell}\left(\frac{R}{r}\right)^{\ell+2}(g_\ell^m\cos(m\phi)+h_\ell^m\sin(m\phi))\frac{d(P_\ell^m(\cos(\theta)))}{d\theta} \\ B_y &= B_\phi = \frac{-1}{r\sin(\theta)}\frac{\partial W}{\partial \phi} \\ &= \sum_{\ell=1}^{\infty}\sum_{m=0}^{\ell}\left(\frac{R}{r}\right)^{\ell+2}(g_\ell^m\sin(m\phi)-h_\ell^m\cos(m\phi))\frac{m}{\sin(\theta)}P_\ell^m(\cos(\theta)) \\ B_z &= -B_r = \frac{\partial W}{\partial r} \\ &= \sum_{\ell=1}^{\infty}\sum_{m=0}^{\ell}-(\ell+1)\left(\frac{R}{r}\right)^{\ell+2}(g_\ell^m\cos(m\phi)+h_\ell^m\sin(m\phi))P_\ell^m(\cos(\theta)) \end{aligned} \tag{3}$$

With these equations in place, a system can be set-up that relates observations to Gauss coefficients. For that purpose, the equations need to be truncated at a finite degree L to allow for computation. To ensure stable solution, most energy of the magnetic field should be described by Gauss coefficients with a degree far smaller than the chosen cut-off degree. The forward equation for solving L degree coefficients with Q magnetic measurement locations (B_x, B_y, and B_z are measured) is as in Box I, or $\vec{B} = \mathbf{G}\vec{m}$. If there are more observations than unknown parameters, the vector $\vec{m}$ can be obtained with a least squares inversion.

2.2. Time-dependency

Independently modeling the magnetic field for snapshots of time may result into Gauss coefficients that strongly oscillate in time if no appropriate measures are taken (e.g. Constable et al., 2000). By formulating the problem (Eq. (4)) as function of time, Constable and Parker (1988) show that time-dependence can be approximated by cubic B-Splines. Cubic B-Splines are piece-wise cubic polynomials: they are a set of individual polynomials joined together at knot points in such a way that they are continuous up to the second derivative. This key property of cubic B-Splines allows a range of temporal damping types (see Section 2.4).

Cubic B-Splines enable a geomagnetic field model that varies continuously through time by solving for model coefficients per spline. These spline coefficients are time-independent, and should not be directly interpreted. By combining these spline coefficients of different splines, the usual time-dependent Gauss coefficients $\vec{m}(t)$ are retrieved:

$$\vec{m}(t) = \sum_{k=1}^{k_{max}} \vec{m}^k M_3^k(t), \tag{5}$$

with $\vec{m}(t)$ the Gauss coefficients at time step t, $\vec{m}^k$ all spline coefficients for spline k, $M_3^k(t)$ the factor that represent what part of spline k belongs to the Gauss coefficients at time t, and k_{max} the total amount of splines used. Fig. 1 shows how to interpret Eq. (5) with k_{max} equal to ten. To reconstruct, for example, Gauss coefficients at time $t = 2$, the coefficients of spline 2, 3, and 4 would be required in the ratio 1/6, 4/6, 1/6, respectively. In most cases, four splines are usually needed for reconstruction: Gauss coefficients at $t = 3.5$ are compiled by splines 3, 4, 5, and 6 in the ratio 1/48, 23/48, 23/48, and 1/48, respectively. Hence, to model Gauss coefficients for $1 \le t < 8$, ten splines pinned by fourteen equally spaced knot points are required: eight knots on $t = 1$ to 8, three knot points before $t = 1$, and three knot points after $t = 8$.

Additionally, Fig. 1 shows how paleomagnetic data, observed between $t = 1$ and $t = 8$, translates into the 10 splines used in a model. For example, spline 1 solely consists of paleomagnetic data for $1 \le t < 2$, while spline 5 consists of data for $2 \le t < 6$. This illustrates the reason why we should not interpret Gauss coefficients at both ends of the time series, because these coefficients are based on splines that contain sparse data (i.e. splines 1, 2, 3, 8, 9, and 10) and could therefore introduce unwanted behavior. Gauss coefficients modeled within two points of the ends of the time series are therefore usually not considered.

With the introduction of cubic B-Splines, the equations can be set-up on a per-spline basis by applying Eq. (5) to Eq. (4). Given T time steps, $T + 2$ ($= k_{max}$) spline coefficients are solved at once (see Box II),

where $\vec{B}_T$ stores all magnetic data (B_x, B_y, B_z) at every location between time step $t = T$ and time step $t = T + 1$, $\vec{t}_T$ contains the time of all data point falling between $t = T$ and $t = T+1$, and $\mathbf{G}_{\vec{t}_T}$ represents the Green's matrix relating all magnetic data of that time interval to the spline coefficients $\vec{m}^k$.

To clarify the different matrix and vector dimensions we present the following example for the first row: for $1 \le t < 2$, we assume 20 observations of the magnetic field (any combination of B_x, B_y, and B_z) to infer a model of degree 3. Since there are 20 observations, $\vec{B}_1$ and $\vec{t}_1$ both have dimensions 20×1. These 20 observations determine, via the Green's matrix $\mathbf{G}_{\vec{t}_1}$ and the spline weighing matrix $\mathbf{M}$, the Gauss coefficients for the first four splines (see Fig. 1). Since the degree of the model is 3, $\vec{m}^1$, $\vec{m}^2$, $\vec{m}^3$, and $\vec{m}^4$ each consists of 15 Gauss coefficients. To link these 15 parameters per spline to the observations, $\mathbf{G}_{\vec{t}_1}$ has dimensions of 20×15. M is a diagonal matrix of 20×20 that contains the spline coefficient (see Fig. 1) on its diagonal, coupling each observation to the one of the four splines as a function of time. The total dimension of $M_3^k(\vec{t}_1)\mathbf{G}_{\vec{t}_1}$ is 20×15 and the dimension of each $\mathbf{0}$-matrix is hence 20×15.

We now extend the complete matrix beyond the first row; in total we model 30 time steps, from $t \ge 1$ to $t < 31$, to infer a degree 3 model. This means that we need at least 32 splines and 15 Gauss coefficients per spline. If we also assume that every time step contains 20 observations, then the vector containing all observations has a dimension of 600×1, the matrix has 600 rows and ($15 \cdot 32$=) 480 columns, and the vector containing all Gauss coefficients in Eq. (6) has a dimension of 480×1.

2.3. Non-linearity

Most paleomagnetic data consist of a combination of inclination I, declination D, and intensity F magnetic data. They are related as follows to the north, east, and vertical component, with the extra

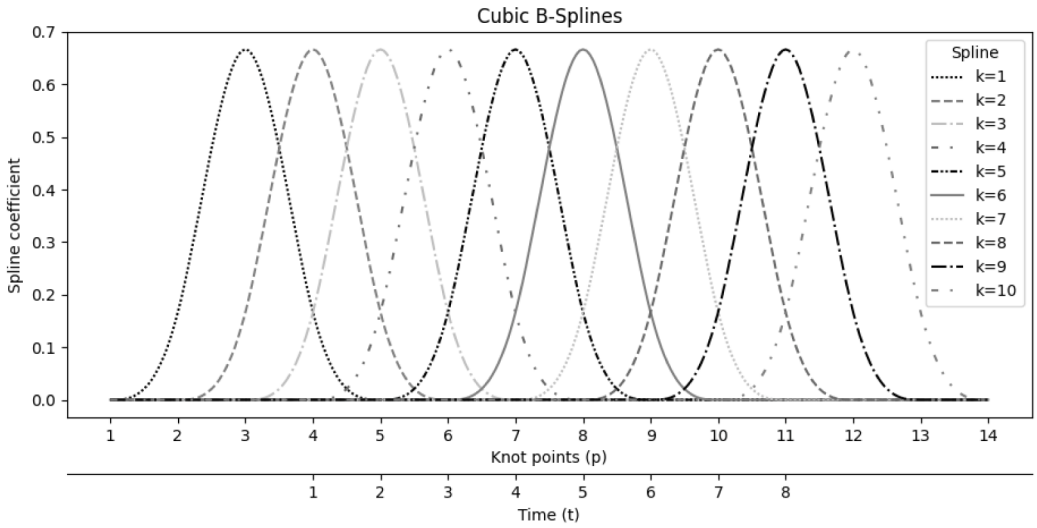


Fig. 1. Example of 10 Cubic B-Splines on 14 knot points covering $t = 1$ till $t = 8$.

$$\begin{bmatrix} \vec{B}_1 \\ \vec{B}_2 \\ \vec{B}_3 \\ \vdots \\ \vec{B}_T \end{bmatrix} = \begin{bmatrix} \mathbf{M}_3^1(\vec{t}_1)\mathbf{G}_{\vec{t}_1} & \mathbf{M}_3^2(\vec{t}_1)\mathbf{G}_{\vec{t}_1} & \mathbf{M}_3^3(\vec{t}_1)\mathbf{G}_{\vec{t}_1} & \mathbf{M}_3^4(\vec{t}_1)\mathbf{G}_{\vec{t}_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_3^2(\vec{t}_2)\mathbf{G}_{\vec{t}_2} & \mathbf{M}_3^3(\vec{t}_2)\mathbf{G}_{\vec{t}_2} & \mathbf{M}_3^4(\vec{t}_2)\mathbf{G}_{\vec{t}_2} & \mathbf{M}_3^5(\vec{t}_2)\mathbf{G}_{\vec{t}_2} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}_3^3(\vec{t}_3)\mathbf{G}_{\vec{t}_3} & \mathbf{M}_3^4(\vec{t}_3)\mathbf{G}_{\vec{t}_3} & \mathbf{M}_3^5(\vec{t}_3)\mathbf{G}_{\vec{t}_3} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M}_3^{k_{max}}(\vec{t}_T)\mathbf{G}_{\vec{t}_T} \end{bmatrix} \begin{bmatrix} \vec{m}^1 \\ \vec{m}^2 \\ \vec{m}^3 \\ \vec{m}^4 \\ \vec{m}^5 \\ \vdots \\ \vec{m}^{k_{max}} \end{bmatrix}, \tag{6}$$

Box II.

horizontal component B_h given for convenience:

$$\begin{aligned} B_h &= \sqrt{B_x^2 + B_y^2} \\ F &= \sqrt{B_x^2 + B_y^2 + B_z^2} \\ I &= \arctan\left(\frac{B_z}{B_h}\right) \\ D &= \arctan\left(\frac{B_y}{B_x}\right) \end{aligned} \tag{7}$$

These components depend non-linearly on the Gauss coefficients, requiring functions that are no longer solvable by standard least-squares inversion techniques. One way of solving the resulting non-linear equations is by applying the iterative inversion scheme introduced by Bloxham and Jackson (1992), based on work of Lünberger (1969). This iterative inversion scheme updates an estimate of the model coefficients until a required level of convergence has been reached. For this convergence, the algorithm relies on the first derivative of forward observations with respect to the model coefficients $\vec{m}$:

$$\vec{m}_{i+1} = \vec{m}_i + ((\mathbf{M}_3\mathbf{A}_i)^T \mathbf{C_e}^{-1} \mathbf{M}_3\mathbf{A}_i + \mathbf{C_m}^{-1})^{-1}((\mathbf{M}_3\mathbf{A}_i)^T \mathbf{C_e}^{-1} \vec{e}_i - \mathbf{C_m}^{-1}\vec{m}_i), \tag{8}$$

with $\vec{m}_{i+1}$ containing the updated spline coefficients after iteration i for all splines and $\mathbf{C_e}$ the data covariance matrix containing the variances of the measurements on its diagonal. These variances should be chosen based on the quality of the input data. $\mathbf{C_m}$ is the regularization matrix (discussed in Section 2.4) and $\vec{e}_i$ the difference (residual) between the measurements (data) $\vec{d}$ and the forward problem $f(\vec{m})$. The forward problem provides predictions of the magnetic field (B^*) based on evaluating the model at times and locations of the respective measurements. $\mathbf{A_i}$ is the matrix build of the partial derivatives (Fréchet derivatives) of the forward observation with respect to the model parameters, evaluated at the current model $\vec{m}_i$ (see matrix in Eq. (6)). $\mathbf{M}_3$ ports the $\mathbf{A_i}$-matrix, or Fréchet matrix, into cubic B-Splines (see Figure S1).

The partial derivatives of the northward, eastward, and vertical magnetic component form the basis for the partial derivatives of the four non-linear magnetic data types (B_h, I, D, and F) with respect to $\vec{m}$. The partial derivative vector for one measurement ($\vec{\mathbf{A}}$, a vector in this case) is:

$$\begin{aligned} \vec{\mathbf{A}}_{B_h} &= \frac{\partial f_{B_h}(\vec{m})}{\partial \vec{m}} = \frac{\partial}{\partial \vec{m}}\sqrt{B_x^{*2} + B_y^{*2}} = \frac{\partial}{\partial \vec{m}}\sqrt{(\vec{\mathbf{A}}_{B_x}\cdot\vec{m})^2 + (\vec{\mathbf{A}}_{B_y}\cdot\vec{m})^2} \\ &= \frac{2(\vec{\mathbf{A}}_{B_x}\cdot\vec{m})\vec{\mathbf{A}}_{B_x} + 2(\vec{\mathbf{A}}_{B_y}\cdot\vec{m})\vec{\mathbf{A}}_{B_y}}{2\sqrt{B_x^{*2} + B_y^{*2}}} = \frac{B_x^*\vec{\mathbf{A}}_{B_x} + B_y^*\vec{\mathbf{A}}_{B_y}}{B_h^*} \\ \vec{\mathbf{A}}_F &= \frac{\partial f_F}{\partial \vec{m}} \\ &= \frac{\partial}{\partial \vec{m}}\sqrt{B_x^{*2} + B_y^{*2} + B_z^{*2}} = \frac{\partial}{\partial \vec{m}}\sqrt{(\vec{\mathbf{A}}_{B_x}\cdot\vec{m})^2 + (\vec{\mathbf{A}}_{B_y}\cdot\vec{m})^2 + (\vec{\mathbf{A}}_{B_z}\cdot\vec{m})^2} \\ &= \frac{2(\vec{\mathbf{A}}_{B_x}\cdot\vec{m})\vec{\mathbf{A}}_{B_x} + 2(\vec{\mathbf{A}}_{B_y}\cdot\vec{m})\vec{\mathbf{A}}_{B_y} + 2(\vec{\mathbf{A}}_{B_z}\cdot\vec{m})\vec{\mathbf{A}}_{B_z}}{2\sqrt{B_x^{*2} + B_y^{*2} + B_z^{*2}}} \\ &= \frac{B_x^*\vec{\mathbf{A}}_{B_x} + B_y^*\vec{\mathbf{A}}_{B_y} + B_z^*\vec{\mathbf{A}}_{B_z}}{F^*} \\ \vec{\mathbf{A}}_I &= \frac{\partial f_I(\vec{m})}{\partial \vec{m}} = \frac{\partial}{\partial \vec{m}}\arctan\left(\frac{B_z^*}{B_h^*}\right) = \frac{\partial}{\partial \vec{m}}\arctan\left(\frac{\vec{\mathbf{A}}_{B_z}\cdot\vec{m}}{\vec{\mathbf{A}}_{B_h}\cdot\vec{m}}\right) \\ &= \frac{1}{1 + \frac{B_z^{*2}}{B_h^{*2}}} \times \frac{B_h^*\vec{\mathbf{A}}_{B_z} - \vec{\mathbf{A}}_{B_h}B_z^*}{B_h^{*2}} = \frac{B_h^*\vec{\mathbf{A}}_{B_z} - B_z^*\vec{\mathbf{A}}_{B_h}}{F^{*2}} \\ \vec{\mathbf{A}}_D &= \frac{\partial f_D(\vec{m})}{\partial \vec{m}} = \frac{\partial}{\partial \vec{m}}\arctan\left(\frac{B_y^*}{B_x^*}\right) = \frac{\partial}{\partial \vec{m}}\arctan\left(\frac{\vec{\mathbf{A}}_{B_y}\cdot\vec{m}}{\vec{\mathbf{A}}_{B_x}\cdot\vec{m}}\right) \\ &= \frac{1}{1 + \frac{B_y^{*2}}{B_x^{*2}}} \times \frac{B_x^*\vec{\mathbf{A}}_{B_y} - B_y^*\vec{\mathbf{A}}_{B_x}}{B_x^{*2}} = \frac{B_x^*\vec{\mathbf{A}}_{B_y} - B_y^*\vec{\mathbf{A}}_{B_x}}{B_h^{*2}}, \end{aligned} \tag{9}$$

with · the dot-product. The Fréchet matrix is assembled by adding the respective rows for each datum together. The matrix requires field predictions (B^*) to calculate partial derivatives. Therefore, a user should input an initial model to calculate these field predictions. Usually an axial dipole is a safe initial guess (e.g. Korte and Constable, 2003). After the first iteration, these forward field predictions originate from the model itself. For the first couple of iterations, the forward calculations are very crude approximations of reality, that change strongly with every iteration step. Eventually, the changes should get smaller and converge, resulting into a relatively stable matrix **A**. Depending on number and quality of data, regularization might be required to further stabilize the solution.

2.4. Regularization

Constructing geomagnetic field models from noisy and sparse paleomagnetic data is an ill-posed inverse problem, with unstable solutions that can show fast oscillations and large overswingings. With the available archeo- and paleomagnetic data we cannot expect to resolve small-scale structures and rapid variations. These unresolved cases leads to unstable solutions that typically have too much power at high spherical harmonic degrees. Field models are therefore derived using regularization to reduce the energy stored in Gauss coefficients of higher degree and to smooth out irregularities by constraining the inversion scheme with an additional term, the regularization or model covariance matrix $\mathbf{C_m}$ (Eq. (8)). This term is composed of both a spatial and a temporal damping component:

$$\mathbf{C_m}^{-1} = \varsigma \mathbf{S}^{-1} + \tau \mathbf{T}^{-1} \tag{10}$$

$\mathbf{S}^{-1}$ and $\mathbf{T}^{-1}$ are symmetric matrices containing the spatial and temporal damping, and ς and τ are weighing factors determining how much emphasis is applied to damping. The desired damping parameters can be obtained by e.g. inspection of powerspectra of the Gauss coefficients, by finding the knee-point in residual-energy plots, by comparison of spectral expectations of known historical and modern fields, or through the cross-validation method (Korte et al., 2009; Panovska et al., 2012; Licht et al., 2013). However, setting a good temporal and spatial damping parameter is not a straightforward task. Setting both damping parameters too high will result into a smoothly varying dipolar field with high data residuals, while a too low setting will result into physically unreasonable models. It is therefore paramount to always compare modeled observations to input data, and consider whether the model is representative for the data it is provided with.

In our algorithm we have included six different constraints for the spatial damping matrix $\mathbf{S}^{-1}$, and two constraints for the temporal damping matrix $\mathbf{T}^{-1}$. All damping methods rely on minimizing the relevant property of the magnetic field at the core-mantle boundary (CMB) at 3485 km distance from the center of the Earth. The complexity of the relevant property is constrained by minimizing the integral of that magnetic field property over the CMB. This minimization translates into a degree-dependent damping function ($S^{-1}(\ell)$ or $T^{-1}(\ell)$) of the Gauss coefficients in the spectral domain (see Table 1). First, the six spatial damping methods are described here:

- (1) Uniform damping: every model parameter receives the same amount of damping.
- (2) Constraining surface heat flux through the core mantle boundary (Gubbins, 1975): through minimization of the curl of the magnetic field, the heat flux through the CMB is constrained. If actual values are used, this method gives a relatively weak constraint to the Gauss coefficients, because only the lower bound of the heat flux out of the core is known. We provide a simplified implementation in the spectral domain that does not include all natural constants, resulting into an underestimation of realistic heat flux values (see Korte et al., 2009). The full derivation is presented in Backus et al. (1996).

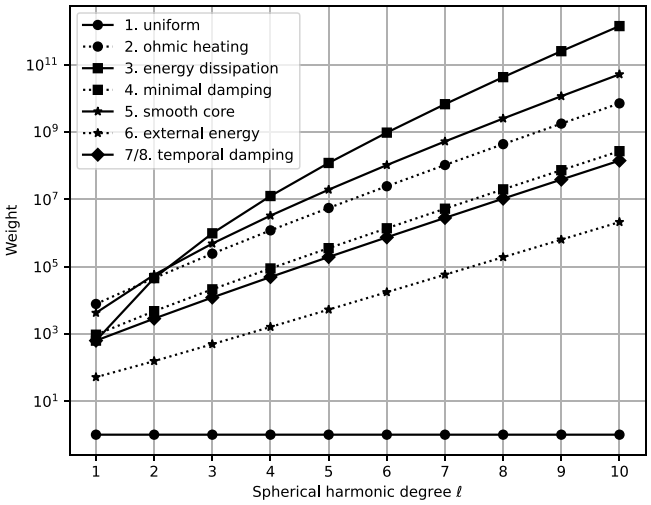


Fig. 2. Weight of all damping types on Gauss coefficients per degree. Physical basis of all damping is summarized in Table 1. Note that 'temporal damping' represents both minimizing velocity and minimizing acceleration of the magnetic field, since their spectral representation is the same.

- (3) Minimizing the energy dissipation through the CMB (Gubbins and Bloxham, 1985; Bloxham, 1987): fluid motions in the outer core can increase small scale feature complexity through advection, which should be smoothed out by diffusion, dictated by the diffusion equation. Geomagnetic models deal with small scale complexity by storing energy in higher degree Gauss coefficients. These high degree coefficients are most of the time not necessary to explain the data, making these coefficients susceptible to data contamination. To reduce model complexity, diffusion and advection are minimized. The full derivation is presented by Bloxham (1987).
- (4) Minimal damping: This damping method provides minimal constraints to reduce magnetic energy in the higher degrees (Lowes, 1974; Shure et al., 1982).
- (5) Smooth radial component: by minimizing the horizontal derivative of the radial component at the CMB, smoothed radial magnetic components are obtained (Shure et al., 1982; Holme and Bloxham, 1996).
- (6) Minimizing external magnetic energy: Holme and Bloxham (1996) present a damping function that minimizes ionospheric influences in satellite magnetic data on Uranus and Neptune. In the context of archeomagnetic and paleomagnetic data this method has no physical meaning.

Effectively, the different methods modify the slope of the geomagnetic power spectrum (see Fig. 2): the energy stored in the Gauss coefficients summed per degree (Lowes, 1974). The Ohmic heating constraint (Gubbins, 1975) is the most widely used in paleomagnetic context.

There are two methods implemented to damp the Gauss coefficients temporally. One form of temporal damping is applied by minimizing the rate (7. in Table 1) at which the radial magnetic field changes at the CMB. The other form involves minimizing the second derivative (8. in Table 1) at which the radial magnetic field changes at the CMB (Korte and Constable, 2003; Korte et al., 2009). The first temporal method promotes a constant magnetic field, while the second method leads to a linearly changing magnetic field through time. Note that both damping methods only work because cubic B-Splines have a second time derivative. All damping types are summarized in Table 1 and their weight on the Gauss coefficients per degree are plotted in Fig. 2.

Table 1
Overview of spatial and temporal damping types applied at the core-mantle boundary (CMB). A damping term function is obtained by translating a physical constraint into the spectral domain. This damping function returns a factor per model parameter depending on degree (see Fig. 2). $R_\oplus$ indicates the radius of the Earth, R_{CMB} is the radius of the core-mantle boundary (CMB), and Δ indicates the angular component of $r^2\nabla^2$.

Damping type	Physical constraint	Damping term function
1. Uniform	–	$S^{-1}(\ell) = 1$
2. Ohmic heating	$\int_{\mathrm{core}} (\nabla \times \mathbf{B})^2 dV$	$S^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+3} \frac{(\ell+1)(2\ell+1)(2\ell+3)}{\ell}$
3. Energy dissipation	$\oint_{\mathrm{CMB}} B_r \frac{\Delta B_r}{r^2} dS$	$S^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+4} \frac{(\ell+1)^2\ell^4}{2\ell+1}$
4. Minimal damping	$\int_{\mathrm{core}} \lvert\mathbf{B}\rvert^2 dV$	$S^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+4} (\ell+1)$
5. Smooth core	$\oint_{\mathrm{CMB}} \lvert\nabla(B_r)\rvert^2 dS$	$S^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+6} \frac{\ell(\ell+1)^3}{2\ell+1}$
6. External energy	$\int_{R_{\mathrm{CMB}}}^{\infty} \lvert\mathbf{B}\rvert^2 dV$	$S^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+1} \frac{\ell+1}{2\ell+1}$
7. Minimize rate	$\oint_{\mathrm{CMB}} \left(\frac{\partial B_r}{\partial t}\right)^2 dS$	$T^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+4} \frac{(\ell+1)^2}{2\ell+1}$
8. Minimize 2nd derivative	$\oint_{\mathrm{CMB}} \left(\frac{\partial^2 B_r}{(\partial t)^2}\right)^2 dS$	$T^{-1}(\ell) = 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+4} \frac{(\ell+1)^2}{2\ell+1}$

After choosing a spatial or temporal damping type and determining the damping terms per spherical harmonic degree, the damping matrix is constructed. The individual elements of this matrix are obtained by integrating the specific function (see Table 1) over time for a specific degree (ℓ) and spline combination. For all damping types this elements are constructed in a similar way, e.g. for minimal damping we obtain for degree ℓ and splines 3 and 4 (covering knot points 4–7 together, see Fig. 1):

$$\mathbf{S}^{-1}(\ell, 3, 4) = \int_t \int_{\mathrm{core}} |\mathbf{B}|^2 dV\, dt \simeq 4\pi \frac{R_\oplus}{R_{\mathrm{CMB}}}^{2\ell+4} (\ell+1) \int_{p=4}^{p=7} M_3^3(p) M_3^4(p) dp, \qquad (11)$$

By performing this integration for all degrees and possible spline combination we end up with the symmetric damping matrix, $\mathbf{C_m}^{-1}$. Depending on the type of damping, we either have to integrate the cubic B-Spline, its derivative (minimal velocity), or its second time derivative (minimal acceleration). The derivatives of cubic B-Splines are obtained with recursive formulae, described in De Boor (1978). A graphical overview of the constructed matrix and vector is found in the supplementary information (Figure S1).

3. Code implementation

The *pymaginverse*-library for creating geomagnetic models is written in Python 3 and is separated in two parts: a class for preparing data-input *InputData*, and a Python class for creating geomagnetic models *FieldInversion*. Calculations for creating forward observations, damping matrices, and (banded) Fréchet matrices have been moved to separate modules located in sub-folders (see Out and Schanner, 2024). Wherever possible, intense matrix formation and manipulation have been optimized using C-extensions for Python (Cython). The main code (*FieldInversion*) requires at least three fixed parameters: a timevector, a maximum spherical harmonic degree, and a starting model. Besides the main code, four tutorials are included to guide the user in generating geomagnetic models and plotting relevant results.

3.1. Loading data

The first step for generating a geomagnetic model is having a database. These data should be accompanied by measurement uncertainties; if no uncertainties are given either these data should be manually assigned realistic error values or these data should be excluded. Furthermore, the user can adjust provided errors, remove data outliers, or create different dataset subsets based on confidence in the data (e.g. Korte et al., 2005; Pavón-Carrasco et al., 2014; Mahgoub et al., 2023a). Paleomagnetic data forms the basis for geomagnetic models, so great care should be taken to actually provide a solid database to the model.

After the database is ready, it can be loaded into the model. *InputData* accepts geomagnetic data in the form of a CSV-file or a Geomagia dataset (Brown et al., 2015a,b). The algorithm expects magnetic data in the geodetic dataframe, in which the coordinate system is based on WGS84 (ellipsoid fitted to Earth). Additionally, the algorithm accepts geocentric data, where the coordinate system is based on a spherically shaped Earth. When providing a CSV-file, the data should be stored row-wise, where each row contains at least time, latitude, longitude, and corresponding geomagnetic data. Hence, declination, inclination, and intensity with respective errors can be stored in the same row (see Tutorials for further clarification).

3.2. Obtaining a model

After the data is loaded into the *InputData*-class, a geomagnetic model can be generated with the *FieldInversion*-class. A code snippet showing basic usage of the algorithm is presented in Fig. 3. The *FieldInversion*-class is initiated by providing a time span with specific time steps to be covered by the model. These time steps should be considered carefully as sufficient paleomagnetic information should be present per time step. It is tempting to create very small time steps to obtain a 'high-resolution' model, but without enough data a smeared model based on the data of nearby time steps is created. This model might seem very good, but it holds little value for those specific time steps. Additionally, changing the size of the time step involves changing the temporal damping parameter to keep temporal fluctuations in order. Lastly, a maximum spherical degree should be provided. In our example, we want a degree 10 model covering −2000 to 1990 with steps of ten years.

After the class is created, the geomagnetic data, prepared by *InputData*, is loaded with the *prepare_inversion*-method. Additionally, spatial and temporal damping matrices are calculated based on user input. Besides damping, this method also initializes the B-Spline basis and Fréchet matrices. The iterative inversion is initiated by making a call to the *run_inversion*-method. The method requires a starting model, damping factors, and a maximum number of iterations. This method is optimized to use the sparse structure of the involved matrices to the fullest (see Eq. (6) and Figure S1), with individual matrix elements being composed with Cython. The resulting matrices are banded and positive definite, so they are efficiently inverted using Cholesky factorization. A realistic starting model should consist of $L(L+2)$ elements, representing the Gauss coefficients in a spherical harmonic model of degree L. The coefficients are first ordered by degree, then by order: g_1^0, g_1^1, h_1^1, g_2^0, g_2^1, h_2^1, etc. The number of iterations should be set high enough to allow for convergence. The user can input a threshold for the change in RMS residual between iterations (r_i and r_{i+1}); when

```python
import numpy as np
import pandas as pd
from pathlib import Path
from pymaginverse import InputData, FieldInversion

# load csv-file into InputData class
path = Path().absolute()
dataset = pd.read_csv(path / 'example_data.csv', index_col=0)
inputdata = InputData(dataset)

##### start geomagnetic field inversion #####
# set time array and maximum spherical degree
test_inv = FieldInversion(t_min=-2000, t_max=1990, t_step=10, maxdegree=10)
# load data-class and set damping types: Ohmic heating and minimum acceleration
test_inv.prepare_inversion(inputdata, spat_type='ohmic_heating',
                           temp_type='min_acc')
# set starting model, should have 120 elements
x0 = np.zeros(test_inv._nr_coeffs)
x0[0] = -30000
# run inversion by setting start model, damp factors, and max # iterations
test_inv.run_inversion(x0, spat_damp=1.0e-13, temp_damp=1.0e-3, max_iter=5)
# save gauss coefficients results
test_inv.save_coefficients(path / 'output', file_name='example')
```

Fig. 3. Code snippet for loading data and generating a geomagnetic field model, using both the *InputData* and *FieldInversion*-class.

the threshold is larger than $\frac{r_{i+1}-r_i}{r_i}$, the iterative inversion is halted. However, best practice is to observe the residuals of the data between iterations; if these are not changing significantly, the model has likely converged.

It is not possible to decide which damping parameters retrieve the 'best' model up front. Optimal damping factors can be obtained by sweeping through models with different damping factors and comparing data residuals against the damping norm ($\vec{m}^T \mathbf{C_m}^{-1} \vec{m}$, Korte and Constable, 2003; Korte et al., 2009). We have included a tool for this task: *sweep_damping*; other methods for finding damping parameters can be found in e.g. Panovska et al. (2012). We would like to stress here that selecting parameters by this criterion does not necessarily lead to the best numerically stable model, i.e. a model that changes minimally for a small data perturbation. Stability should be further investigated by e.g. comparing geomagnetic models based on subsets of the original dataset (Mahgoub et al., 2023b).

After damping parameters have been set, and the inversion has run, the resulting Gauss coefficients can be saved in *npy*-format with the *save_coefficients*-method or in *csv*-format with the *coefficients_csv*-method. Besides the coefficients, data residuals and spatial+temporal damping norms can be saved. Since a lot of geomagnetic field modeling has been executed with Fortran libraries, Gauss coefficients can also be saved in legacy Fortran-format. Furthermore, a method is included to save the generated coefficients as a *pymagglobal*-model (Schanner et al., 2020), allowing for further processing and advanced plotting.

Lastly, basic plotting tools are included in the Tutorials. These tools allow users to plot Gauss coefficients through time, magnetic field maps, residuals, powerspectra and secular variation (see Fig. 4). To asses the influence of the age uncertainties, it is possible to run the provided algorithm multiple times while bootstrapping the data (see e.g. Korte et al., 2009, 2011; Constable et al., 1987). However, this is not implemented in the presented version.

3.3. Benchmark

In order to ensure consistency and usability of the novel code, we compare its results against an original Fortran implementation. Using a reference dataset, the results of multiple inversions of both codes are compared. Tests included in the codebase ensure that the output of the Python 3 implementation agrees with the Fortran results. Initially, we were hoping to achieve agreement within the machine precision limit. However, we found that for some coefficients the output of the Python implementation deviates up to 20 nT from the output of the Fortran version. This slight difference may be due to different precision in the code (Fortran uses 32-bit double, while the numpy library uses 64-bit) or due to different implementations of the matrix solving algorithm. Fig. 5 shows the mean and maximum difference between all coefficients of the Fortran and Python solution. The mean difference is about 0.3 nT, while the maximal difference is at around 17 nT. Fig. 6 shows the solutions for the two codes for multiple iterations. Gray lines indicate the solutions after the final (9th) iteration. Both solutions are visually indistinguishable. After the second iteration, they are also visually indistinguishable from the final solution, which agrees for both implementations. The picture is similar for the higher order coefficients.

To benchmark the code and prevent regressions, we include tests that ensure that the Python solution does not deviate more than 25 nT from the Fortran solution, both after one and ten iterations. The level of 25 nT is well below the precision that can be expected from paleomagnetic data. Benchmark dataset and coefficients are provided with the git-repository of the code (Out and Schanner, 2024).

The most prominent critique brought forward against the Python language is its slowness. Nowadays, many scientific libraries like Numpy and Scipy include a C, C++, or Fortran backend to lift heavy computational weight. By using inversion routines from the scipy

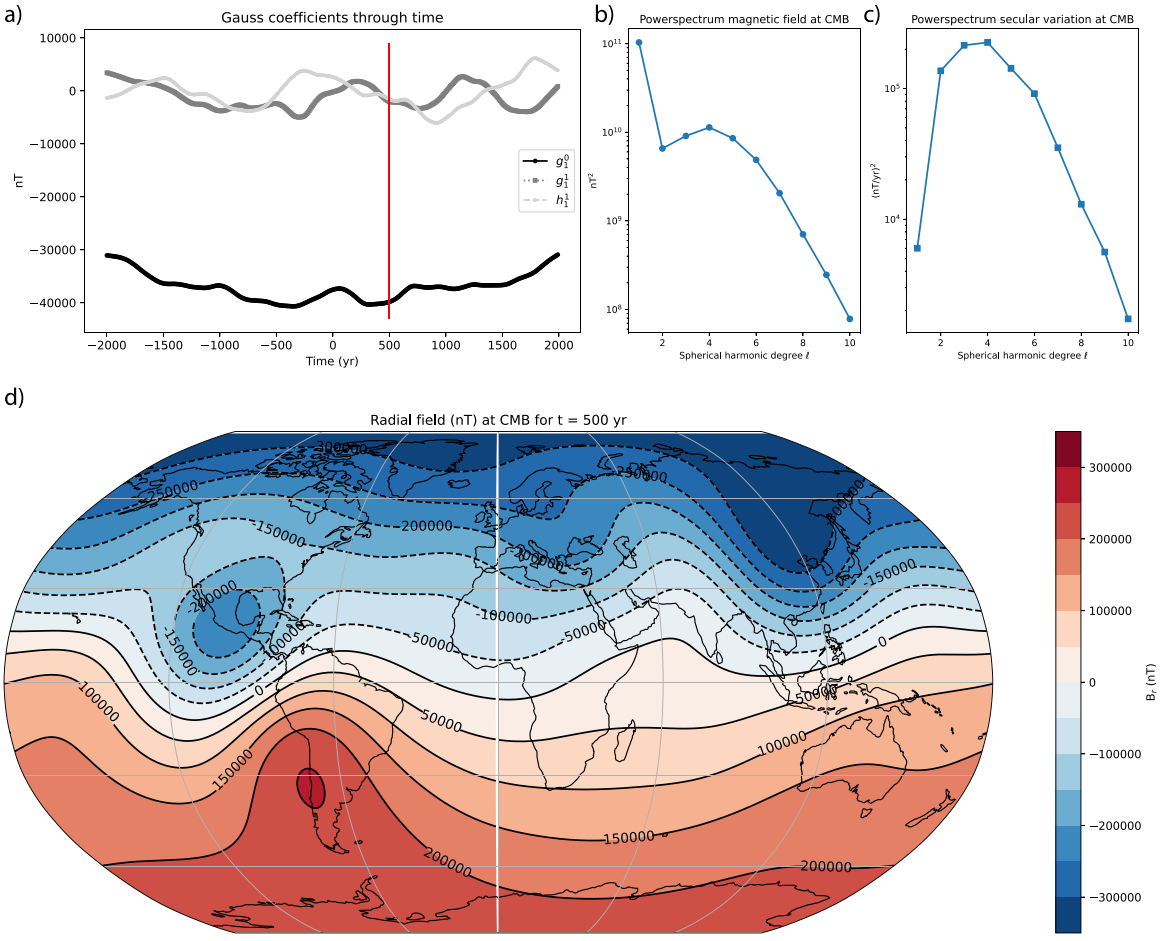


Fig. 4. Example of plotting tools available in Tutorials. All plots are based on generated Holocene data using *CALS10k.2* (Constable et al., 2016) with *pymagglobal* (Schanner et al., 2020). (a) Degree 1 Gauss coefficients; red line indicates t = 500. (b–c) Powerspectra of magnetic field and secular variation per spherical harmonic degree at the CMB; these plots are used to determine stability of damping parameters. (d) Radial magnetic field at the CMB plotted for t = 500.

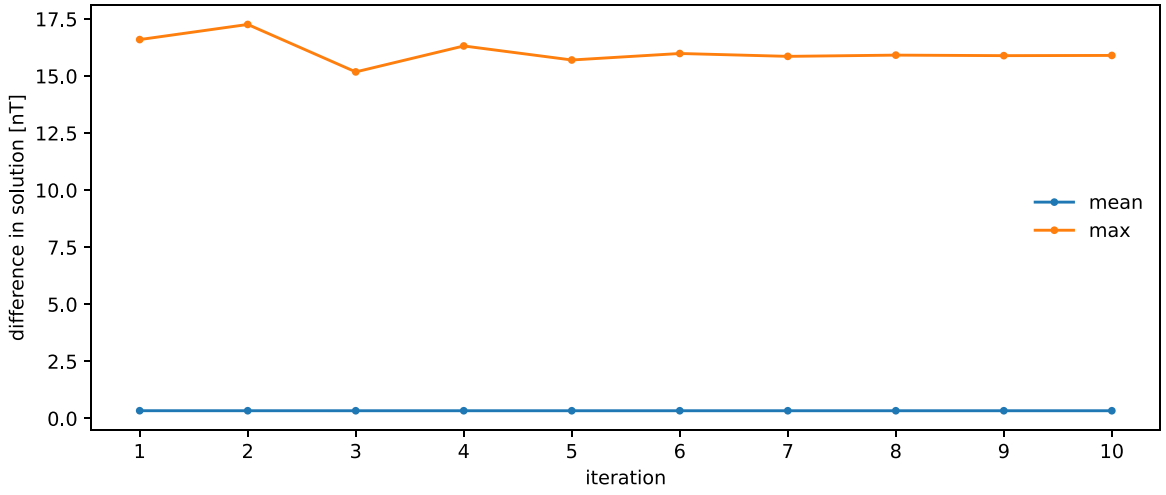


Fig. 5. Absolute difference in the coefficient solution between the Fortran and Python implementation per iteration. The orange line shows the maximal difference while the blue line shows the mean difference. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

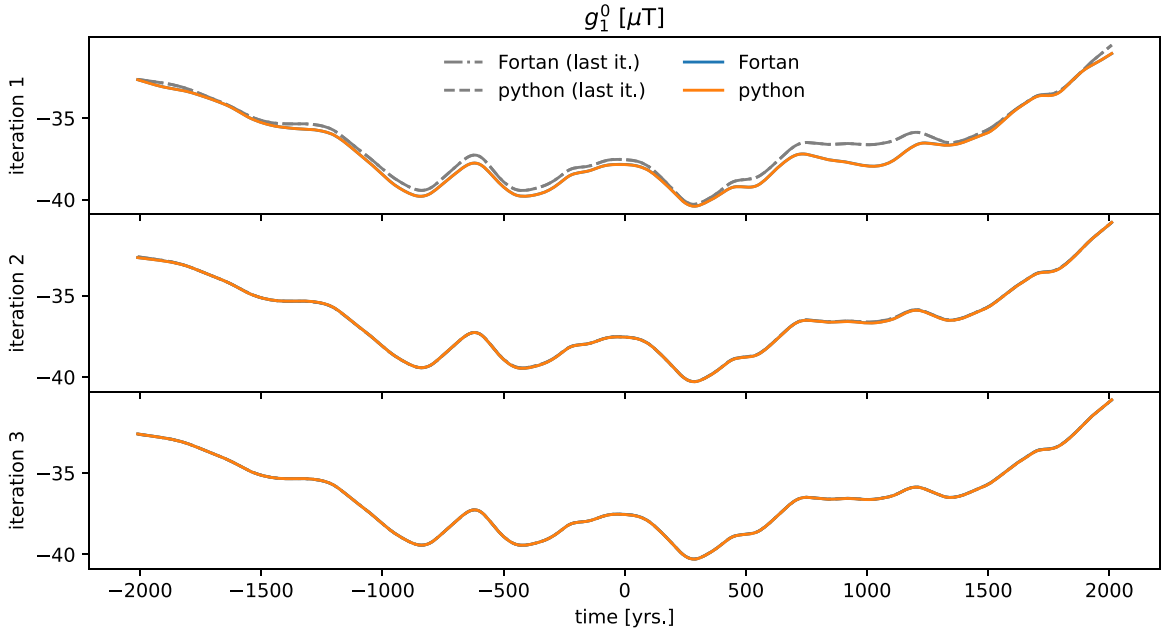


Fig. 6. Evolution of the solution for the axial dipole after multiple iterations of the inversion. The dashed line shows the solution of the Python implementation after 10 iterations. The dash-dotted line shows the same for the Fortran code, but is visually indistinguishable from the Python solution. The Fortran (blue line) and Python (orange line) solutions agree so well that no difference is visible. Both solutions have converged to the final solution already after the second iteration and the dashed line disappears behind the orange one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

package, our Python implementation is able to keep up with its Fortran predecessor. The bottleneck of our code was the construction of the banded normal equation matrices. To circumvent this bottleneck, we changed the loop order from the original Fortran version. Our outer loop runs over the matrix bands, while in Fortran the outer loop is over the data. We found our approach to be faster and implemented this part of the algorithm in Cython, maintaining Python readability but gaining a massive speedup compared to pure Python.

Another benefit of the Cython implementation lies on its low memory usage. Numpy and other python libraries rely on array operations, but the construction of the normal equations matrices is not well suited for formulation on array basis, as the banded structure cannot be utilized efficiently. This leads to large memory requirements when running the code. An early version in array formulation required more than 15 GB of computer memory to build the matrices for a dataset of about 10.000 observations, 401 spline knots and a cutoff degree order of 10. With the loop-based Cython implementation, due to exploiting the banded structure, the code now requires about 400 MB for the same dataset and can therefore be run comfortably on standard notebooks.

The top panel of Fig. 7 provides a runtime comparison of the Python implementation against the original Fortran version. As evident, the runtimes are comparable, with our Python version being slightly faster. The runtime benchmark was conducted on synthetic (non-linear) declination, inclination, and intensity data, generated using *pymagglobal* (Schanner et al., 2020). For each implementation, data loading routines, damping matrix construction, and one iteration were timed using the CPU clock. Three runs were performed and the best of these three was noted for evaluation. The model cutoff degree was set to 10. When decreasing the cutoff degree, the Fortran code becomes slightly faster than the Python version, but runtimes become so small that this advantage is negligible. On our testing machine, increasing the cutoff degree beyond 14 is not possible with the Fortran reference we were provided due to memory issues. The Python version maintains practicable runtimes up to cutoff degree 20 and possibly beyond. A runtime comparison for different cutoff degrees is shown in the bottom panel of Fig. 7.

4. Conclusions

In this article we have introduced a modern Python 3 algorithm for creating a time-dependent spherical harmonic model of the geomagnetic core field. This algorithm is based on a frequently used Fortran version, which circulates in the geomagnetic modeling community in many different versions. To make this new algorithm appealing, we ensured that the Python 3 algorithm is easy to install, user-friendly given basic Python skills, and as fast as its Fortran predecessor. By making use of the well developed banded structure of the involved matrices, we have created a memory-efficient algorithm that can be executed on a regular laptop. Additionally, the algorithm contains methods to export the data to csv, a Fortran legacy format, and *pymagglobal* format (Schanner et al., 2020), allowing further advanced processing and plotting with established scripts and tools.

We have included four tutorials and numerous comments throughout the algorithm to showcase all functionalities and basic plotting options. Combining this with a concise overview of background literature, we aim to centralize the RLS method. With this central repository, we aim that (future) users can easily compare results of their geomagnetic model and further build on this, ensuring that this useful method remains accessible to future generations of geomagnetic field modelers.

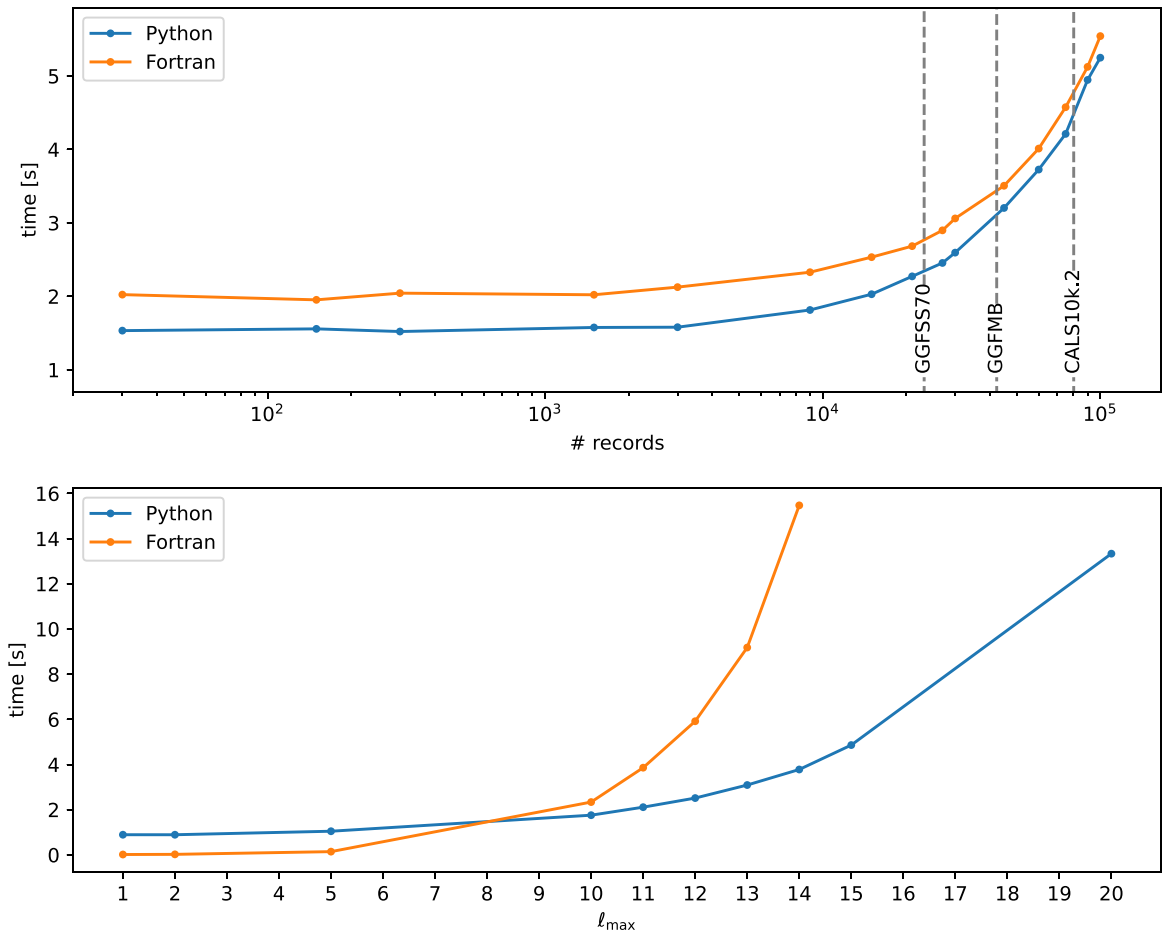


Fig. 7. Runtime benchmark of the Python 3 implementation (blue) against its Fortran predecessor (orange). Data and matrix setup and a single iteration of the inversion were timed using the CPU clock. For each marker, three runs were performed per implementation and the best value is reported here. The top panel shows the runtime dependency on datasets of varying size. The gray dashed lines indicate the number of data used to create exemplary models. The bottom panel shows the runtime dependency on the cutoff degree. See the text for additional information. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Code availability section

The algorithm is publicly available and hosted in the Github repository at https://github.com/outfrenk/pymaginverse under a MIT License. The algorithm is written in the Python, Cython, and C programming languages. The codes utilize Python numerical libraries which include Numpy (Harris et al., 2020), Scipy (Virtanen et al., 2020), matplotlib (Hunter, 2007), Jupyter notebooks (Kluyver et al., 2016), Cython (Behnel et al., 2011), pandas (WesMcKinney, 2010), and pyshtools (Wieczorek and Meschede, 2018).

All the codes and benchmarks from this study are published in the public GitHub repository (Out and Schanner, 2024), version 1.1.

CRediT authorship contribution statement

Frenk Out: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Maximilian Schanner:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis. **Liz van Grinsven:** Writing – review & editing, Validation, Conceptualization. **Monika Korte:** Writing – review & editing, Resources, Funding acquisition. **Lennart V. de Groot:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The presented Python code is based on a variety of Fortran codes that have been spread within the geomagnetic community by personal communication and in its original version were mainly written by David Gubbins, Kathryn Whaler, Jeremy Bloxham, and Andrew Jackson. The authors want to express their gratitude to Sanja Panovska for the fruitful discussions on the algorithm and its spatial and temporal damping options. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 851460 to L.V. de Groot). This project is also funded by the Dutch Science Foundation (NWO) VIDI grant VI.Vidi.192.047 to L.V. de Groot. M.S. received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Grant 388291411.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.acags.2025.100222.

Data availability

The algorithm and data are publicly available and hosted in the Github repository at https://github.com/outfrenk/pymaginverse under a MIT License.

References

Alken, P., Thébault, E., Beggan, C.D., Amit, H., Aubert, J., Baerenzung, J., Bondar, T., Brown, W., Califf, S., Chambodut, A., et al., 2021. International geomagnetic reference field: The thirteenth generation. Earth, Planets Space 73, 1–25. http://dx.doi.org/10.1186/s40623-020-01288-x.

Arneitz, P., Egli, R., Leonhardt, R., Fabian, K., 2019. A Bayesian iterative geomagnetic model with universal data input: Self-consistent spherical harmonic evolution for the geomagnetic field over the last 4000 years. Phys. Earth Planet. Inter. 290, 57–75. http://dx.doi.org/10.1016/j.pepi.2019.03.008.

Backus, G., Parker, R.L., Constable, C., 1996. Foundations of geomagnetism. Cambridge University Press.

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D.S., Smith, K., 2011. Cython: The best of both worlds. Comput. Sci. Eng. 13 (2), 31–39. http://dx.doi.org/10.1109/MCSE.2010.118.

Bloxham, J., 1987. Simultaneous stochastic inversion for geomagnetic main field and secular variation: 1. A large-scale inverse problem. J. Geophys. Research: Solid Earth 92 (B11), 11597–11608. http://dx.doi.org/10.1029/JB092iB11p11597.

Bloxham, J., Jackson, A., 1992. Time-dependent mapping of the magnetic field at the core-mantle boundary. J. Geophys. Res. 97 (B13), 19537. http://dx.doi.org/10.1029/92JB01591.

Brown, M.C., Donadini, F., Korte, M., Nilsson, A., Korhonen, K., Lodge, A., Lengyel, S.N., Constable, C.G., 2015a. GEOMAGIA50. v3: 1. General structure and modifications to the archeological and volcanic database. Earth, Planets Space 67, 1–31. http://dx.doi.org/10.1186/s40623-015-0232-0.

Brown, M.C., Donadini, F., Nilsson, A., Panovska, S., Frank, U., Korhonen, K., Schuberth, M., Korte, M., Constable, C.G., 2015b. GEOMAGIA50. v3: 2. A new paleomagnetic database for lake and marine sediments. Earth, Planets Space 67, 1–19. http://dx.doi.org/10.1186/s40623-015-0233-z.

Constable, C.G., Johnson, C.L., Lund, S.P., 2000. Global geomagnetic field models for the past 3000 years: Transient or permanent flux lobes?. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 358 (1768), 991–1008. http://dx.doi.org/10.1098/rsta.2000.0570.

Constable, C., Korte, M., Panovska, S., 2016. Persistent high paleosecular variation activity in southern hemisphere for at least 10 000 years. Earth Planet. Sci. Lett. 453, 78–86. http://dx.doi.org/10.1016/j.epsl.2016.08.015.

Constable, C., Parker, R., 1988. Smoothing, splines and smoothing splines; their application in geomagnetism. J. Comput. Phys. 78 (2), 493–508. http://dx.doi.org/10.1016/0021-9991(88)90062-9.

Constable, S.C., Parker, R.L., Constable, C.G., 1987. Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data. Geophysics 52 (3), 289–300. http://dx.doi.org/10.1190/1.1442303.

De Boor, C., 1978. A practical guide to splines. vol. 27, springer-verlag New York.

Finlay, C.C., Kloss, C., Olsen, N., Hammer, M.D., Tøffner-Clausen, L., Grayver, A., Kuvshinov, A., 2020. The CHAOS-7 geomagnetic field model and observed changes in the south atlantic anomaly. Earth, Planets Space 72 (1), 156. http://dx.doi.org/10.1186/s40623-020-01252-9.

Gubbins, D., 1975. Can the earth's magnetic field be sustained by core oscillations?. Geophys. Res. Lett. 2 (9), 409–412. http://dx.doi.org/10.1029/GL002i009p00409.

Gubbins, D., Bloxham, J., 1985. Geomagnetic field analysis—III. Magnetic fields on the core—mantle boundary. Geophys. J. Int. 80 (3), 695–713. http://dx.doi.org/10.1111/j.1365-246X.1985.tb05119.x.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585 (7825), 357–362. http://dx.doi.org/10.1038/s41586-020-2649-2.

Hellio, G., Gillet, N., 2018. Time-correlation-based regression of the geomagnetic field from archeological and sediment records. Geophys. J. Int. 214 (3), 1585–1607. http://dx.doi.org/10.1093/gji/ggy214.

Holme, R., Bloxham, J., 1996. The magnetic fields of Uranus and Neptune: Methods and models. J. Geophys. Research: Planets 101 (E1), 2177–2200. http://dx.doi.org/10.1029/95JE03437.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9 (3), 90–95. http://dx.doi.org/10.1109/MCSE.2007.55.

Jackson, A., Jonkers, A.R., Walker, M.R., 2000. Four centuries of geomagnetic secular variation from historical records. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 358 (1768), 957–990. http://dx.doi.org/10.1098/rsta.2000.0569.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., development team, J., 2016. Jupyter notebooks - a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, Netherlands, pp. 87–90. http://dx.doi.org/10.3233/978-1-61499-649-1-87.

Korte, M., Constable, C., 2003. Continuous global geomagnetic field models for the past 3000 years. Phys. Earth Planet. Inter. 140 (1–3), 73–89. http://dx.doi.org/10.1016/j.pepi.2003.07.013.

Korte, M., Constable, C., Donadini, F., Holme, R., 2011. Reconstructing the Holocene geomagnetic field. Earth Planet. Sci. Lett. 312 (3), 497–505. http://dx.doi.org/10.1016/j.epsl.2011.10.031.

Korte, M., Donadini, F., Constable, C., 2009. Geomagnetic field for 0–3 ka: 2. A new series of time-varying global models. Geochem. Geophys. Geosyst. 10 (6), http://dx.doi.org/10.1029/2008GC002297.

Korte, M., Genevey, A., Constable, C., Frank, U., Schnepp, E., 2005. Continuous geomagnetic field models for the past 7 millennia: 1. A new global data compilation. Geochem. Geophys. Geosyst. 6 (2), http://dx.doi.org/10.1029/2004GC000800.

Leonhardt, R., Fabian, K., 2007. Paleomagnetic reconstruction of the global geomagnetic field evolution during the Matuyama/Brunhes transition: Iterative Bayesian inversion and independent verification. Earth Planet. Sci. Lett. 253 (1–2), 172–195. http://dx.doi.org/10.1016/j.epsl.2006.10.025.

Licht, A., Hulot, G., Gallet, Y., Thébault, E., 2013. Ensembles of low degree archeomagnetic field models for the past three millennia. Phys. Earth Planet. Inter. 224, 38–67. http://dx.doi.org/10.1016/j.pepi.2013.08.007.

Lowes, F.J., 1974. Spatial power spectrum of the main geomagnetic field, and extrapolation to the core. Geophys. J. Int. 36 (3), 717–730. http://dx.doi.org/10.1111/j.1365-246X.1974.tb00622.x.

Lowrie, W., 2011. A student's guide to geophysical equations. Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511794438.

Lünberger, D.G., 1969. Optimization by vector space methods. John Wiley & Sons.

Mahgoub, A.N., Korte, M., Panovska, S., 2023a. Characteristics of the Matuyama-Brunhes magnetic field reversal based on a global data compilation. J. Geophys. Research: Solid Earth 128 (2), http://dx.doi.org/10.1029/2022JB025286, e2022JB025286.

Mahgoub, A.N., Korte, M., Panovska, S., 2023b. Global geomagnetic field evolution from 900 to 700 ka including the Matuyama-Brunhes reversal. J. Geophys. Research: Solid Earth http://dx.doi.org/10.1029/2023JB026593.

Mauerberger, S., Schanner, M., Korte, M., Holschneider, M., 2020. Correlation based snapshot models of the archeomagnetic field. Geophys. J. Int. http://dx.doi.org/10.1093/gji/ggaa336, arXiv:https://academic.oup.com/gji/article-pdf/doi/10.1093/gji/ggaa336/33490503/ggaa336.pdf.

Nilsson, A., Holme, R., Korte, M., Suttie, N., Hill, M., 2014. Reconstructing Holocene geomagnetic field variation: New methods, models and implications. Geophys. J. Int. 198 (1), 229–248. http://dx.doi.org/10.1093/gji/ggu120.

Nilsson, A., Suttie, N., 2021. Probabilistic approach to geomagnetic field modelling of data with age uncertainties and post-depositional magnetisations. Phys. Earth Planet. Inter. 317, 106737. http://dx.doi.org/10.1016/j.pepi.2021.106737.

Out, F., Schanner, M.A., 2024. Least squares spherical harmonic (paleo)geomagnetic field modeling with pymaginverse. http://dx.doi.org/10.5281/zenodo.11098494, Github: https://github.com/outfrenk/pymaginverse.

Panovska, S., Constable, C., Korte, M., 2018. Extending global continuous geomagnetic field reconstructions on timescales beyond human civilization. Geochem. Geophys. Geosyst. 19 (12), 4757–4772. http://dx.doi.org/10.1029/2018GC007966.

Panovska, S., Finlay, C.C., Donadini, F., Hirt, A.M., 2012. Spline analysis of Holocene sediment magnetic records: Uncertainty estimates for field modeling. J. Geophys. Research: Solid Earth 117 (B2), http://dx.doi.org/10.1029/2011JB008813.

Panovska, S., Korte, M., Finlay, C.C., Constable, C.G., 2015. Limitations in paleomagnetic data and modelling techniques and their impact on Holocene geomagnetic field models. Geophys. J. Int. 202 (1), 402–418. http://dx.doi.org/10.1093/gji/ggv137.

Pavón-Carrasco, F.J., Osete, M.L., Torta, J.M., De Santis, A., 2014. A geomagnetic field model for the Holocene based on archaeomagnetic and lava flow data. Earth Planet. Sci. Lett. 388, 98–109. http://dx.doi.org/10.1016/j.epsl.2013.11.046.

Schanner, M., Korte, M., Holschneider, M., 2022. ArchKalmag14k: a Kalman-filter based global geomagnetic model for the Holocene. J. Geophys. Research: Solid Earth 127 (2), http://dx.doi.org/10.1029/2021JB023166, e2021JB023166.

Schanner, M.A., Mauerberger, S., Korte, M., 2020. pymagglobal - Python interface for global geomagnetic field models. GFZ Dataservices, http://dx.doi.org/10.5880/GFZ.2.3.2020.005.

Shure, L., Parker, R.L., Backus, G.E., 1982. Harmonic splines for geomagnetic modelling. Phys. Earth Planet. Inter. 28 (3), 215–229. http://dx.doi.org/10.1016/0031-9201(82)90003-6.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17, 261–272. http://dx.doi.org/10.1038/s41592-019-0686-2.

WesMcKinney, 2010. Data structures for statistical computing in python. In: van der Walt, S., Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference. pp. 56–61. http://dx.doi.org/10.25080/Majora-92bf1922-00a.

Wieczorek, M.A., Meschede, M., 2018. SHTools: Tools for working with spherical harmonics. Geochem. Geophys. Geosyst. 19 (8), 2574–2592. http://dx.doi.org/10.1029/2018GC007529.