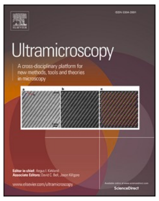
Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic



TEMGYM Advanced: Software for electron lens aberrations and parallelised electron ray tracing

David Landers [a], Ian Clancy [a], Rafal E. Dunin-Borkowski [b], Dieter Weber [b], Andrew Stewart [c,*]

[a] Department of Physics, University of Limerick, Limerick, Munster, V94 T9PX, Ireland
[b] Ernst Ruska-Centre for Microscopy and Spectroscopy with Electrons, Forschungszentrum Jülich, 52425 Jülich, Germany
[c] Department of Chemistry, University College London, 20 Gordon St, London, WC1H 0AJ, United Kingdom

ARTICLE INFO

Keywords:
Ray Tracing
Parallelisation
Differential Algebra
Aberration Integral
NanoMi

ABSTRACT

Characterisation of the electron beams trajectory in an electron microscope is possible in a few select commercial software packages, but these tools and their source code are not available in a free and accessible manner. This paper introduces the free and open-source software TEMGYM Advanced, which implements ray tracing methods that calculate the path of electrons through a magnetic or electrostatic lens and allow evaluation of the first-order properties and third-order geometric aberrations. Validation of the aberration coefficient calculations is performed by implementing two independent methods – the aberration integral and differential algebra (DA) methods and by comparing the results of each. This paper also demonstrates parallelised electron ray tracing through a series of magnetic components, which enables near real-time generation of a physically accurate beam-spot including aberrations and brings closer the realisation of a digital twin of an electron microscope. TEMGYM Advanced represents a valuable resource for the electron microscopy community, providing an accessible and open source means of characterising electron lenses. This software utilises the Python programming language to complement the growing ecosystem of free and open-source software within the electron microscopy community, and to facilitate the application of machine learning to an electron microscope digital twin for instrument automation. The software is available under GNU Public License number Three (GPL 3).

1. Introduction

Over the last decade, an ecosystem of open-source software for electron microscopy has evolved through packages such as LiberTEM, PyXem, HyperSpy & AtomAI [1–4], and these have enabled users to automate data analysis, share results more easily online *via* Google Colab or Jupyter notebooks [5] and facilitate reproducibility and verification of results. There has also been a growing adherence to the FAIR principles [6] for data, requiring scientists to make their data findable, accessible, interoperable, and reproducible. However, an area of electron microscopy which has not yet benefited from this approach is that of electron optics, which includes ray tracing techniques, calculation of aberration coefficients, and generation of spot diagrams. Commercial software such as MEBS & EOD [7,8] are available, but often at cost for researchers, which means they are not ubiquitous within the electron microscopy community. A growing interest within the electron microscopy community to design and modify electron microscope components [9,10] along with funding bodies beginning to require further openness and interoperability amongst electron microscope designs [11], an open-source package capable of producing electron optics calculations would help facilitate cooperation, accelerate discovery and make repetition of experimental results more accessible. This paper introduces TEMGYM Advanced, a Python-based electron optics software for calculating ray tracing paths through electrostatic and electromagnetic lenses for charged particles. Our previous iteration of this software, TEMGYM Basic [12] (under review) is a 3D interactive environment for teaching electron microscope alignments which uses a linear description of the beam path through a series of electron optical components.

In contrast, TEMGYM Advanced correctly calculates the trajectory of electrons through a lens, deflector, or quadrupole by solving the trajectory equations of motion. The development of the software has been aimed primarily at electron optics, although minor modifications would enable it to include other charged particles. Our choice of the Python programming language to develop this software is multifaceted, making use of numerous packages [13–16] to develop the software efficiently, as well as in the future allowing our code to interface smoothly with the

* Corresponding author.
E-mail address: andy.stewart@ucl.ac.uk (A. Stewart).

https://doi.org/10.1016/j.ultramic.2023.113738
Received 19 February 2023; Accepted 3 April 2023
Available online 5 April 2023

growing community of Python-based electron microscopy software and machine learning packages. The TEMGYM Advanced software enables users to perform basic electron optical characterisation routines such as finding the first-order properties (e.g. magnification and focal length) of an axially symmetric lens or finding the aberration coefficients of an analytical lens potential using the aberration integral or the differential algebra method. The differential algebra (DA) method is an alternative method to calculating aberration coefficients and negates the need to derive and calculate aberration integrals by formulating the solution to the electron equations of motion as a Taylor expansion. As far as the authors are aware, the DA method in electron microscopy was only previously available through the academically licensed software COSY INFINITY or the commercially available software MEBS. A further unique aspect of TEMGYM Advanced which separates it from commercial software is the ability to perform ray tracing calculations in parallel and form a spot on the detector in real time, which is discussed further in the section Parallelised Electron Ray Tracing.

2. Analytical electrostatic and magnetic field models with SymPy

Calculating the trajectory of an electron or charged particle through a lens first requires a description of the field near the optical axis. The general expression of a multipole field expansion is given by: [17,18]

$$\varphi(x,y,z) = \left(\sum_{\kappa=0}^{m} \frac{(-1)^{\kappa} x^{-2\kappa+m} y^{2\kappa} m!}{(2^{\kappa})!(-2\kappa+m)!}\right) \sum_{\kappa=i}^{j+i} \frac{(-1)^{\kappa} 4^{-\kappa} (x^2+y^2)^{\kappa} m! U_0^{2\kappa}(z)}{\kappa!(\kappa+m)!} \quad (1)$$

Where U_m is the axial field potential harmonic, j is the number of components to calculate in the series, and m represents the harmonic series to calculate. For $m = 0$, the series calculates the expansion for a round electron optical lens. For $m = 1$, the series calculates the expansion for a dipole harmonic, $m = 2$ is the first quadrupole harmonic etc. The $\kappa = i$ term in the summation determines whether to calculate the expansion for the coordinates x *and* y terms which start at $\kappa = i = 1$, or the z coordinate terms, which start at $\kappa = i = 0$.

Provided the shape of the field produced by the lens along the z-axis is known, expanding equation (1) to j^{th} order generates a function that describes the field beyond the paraxial regime. By paraxial regime, we mean the case where electrons travel close to the centre of the lens with a shallow slope.

Equation (1) is implemented in the software *via* the symbolic computing toolbox SymPy [14], which allows one to compute derivatives & integrals analytically, and its convenient functionality provides a seamless way to expand equation (1) for any required potential. SymPy symbolically computes derivatives for even the most complex axial function representations, enabling a quick and convenient functional representation of the 3D magnetic & electric field surrounding any round lens.

Expansion of equation (1) requires choosing the order of the expansion, m, and the number of terms to calculate j. Calculation of a single component (for example, x) of the series expansion of an axially symmetric electrostatic lens ($m = 0$) with two terms, ($j = 1$) using equation (1) is performed as follows:

$$\sum_{k=0}^{m=0} \frac{(-1)^{\kappa} x^{-2\kappa} y^{2\kappa}}{(2^{\kappa})!(-2\kappa)!} = 1$$

$$\varphi(x,\ y,\ z) = \left(\sum_{\kappa=1}^{j=2} \frac{(-1)^{\kappa} 4^{-\kappa} U_0^{2\kappa}(z)}{\kappa!^2} (x^2+y^2)^{\kappa}\right) \quad (2)$$

$$E_x = -\frac{d\varphi}{dx} = U''(z)\frac{x}{2} - U''''(z)\frac{x(x^2+y^2)}{16}$$

To calculate the x component of the expansion for an axially symmetric magnetic field with two terms, $B(z)$, the potential $\varphi(z)$ must be replaced by the magnetic scalar potential $\omega(z)$, and the axial potential $U(z)$ must be replaced by the axial magnetic potential $\Omega(z)$, where the quantities $B(z)$ and $\Omega(z)$ are connected by:

$$B(z) = -\mu\Omega'(z) \quad (3)$$

where μ is the permeability. thus:

$$\omega(x,\ y,\ z) = \left(\sum_{k=1}^{j=2} \frac{(-1)^{\kappa} 4^{-\kappa} B_0^{2\kappa-1}(z)}{\kappa!^2} (x^2+y^2)^{\kappa}\right) \quad (4)$$

$$B_x = \frac{d\omega}{dx} = -B'(z)\frac{x}{2} + B'''(z)\frac{x(x^2+y^2)}{16}$$

A similar process can be followed to obtain the y and z components of an axially symmetric field distribution. Although usually for an axially symmetric field distribution, cylindrical coordinates are more appropriate, in this work we opt to use the cartesian representation because it enables us to more easily compare the results of our aberration calculations to previous work which also uses the cartesian representation [19,20], and also because it more readily enables us to calculate the trajectory of the electron in 3D as shown in the section parallelised electron ray tracing. Once an axial field function is specified for a given lens configuration, it is possible to reuse this expansion for numerous calculations. For example, this paper shows how to use equation (1) with two terms to trace thousands of rays in parallel through a series of magnetic components and obtain an image of the electron spot that contains third-order geometric aberrations, or use the expansion to calculate the aberration coefficients for a round magnetic or electrostatic lens using the aberration integral or the differential algebra method.

The analytical axial field functions analysed in this paper are Glaser's Bell Shaped Field and Schiske's Electrostatic Lens [21,22]

$$B(z) = \frac{1}{1+\left(\frac{z}{a}\right)^2} \quad (5)$$

$$\phi(z) = \phi_0\left(1 - \frac{k^2}{1+\left(\frac{z}{a}\right)^2}\right) \quad (6)$$

a and k in equation (5) and equation (6) are variables which affect the shape of the axial distribution, and ϕ_0 is the acceleration voltage of the electron.

Although outside of the scope of this paper, it is possible to generate the expanded series for a dipole, quadrupole & octupole and so on, provided one knows the axial harmonic functions. However, in general, this is not a straightforward problem to solve, and the author is only aware of one example in the literature displaying how to calculate the axial harmonics of a dipole system [23].

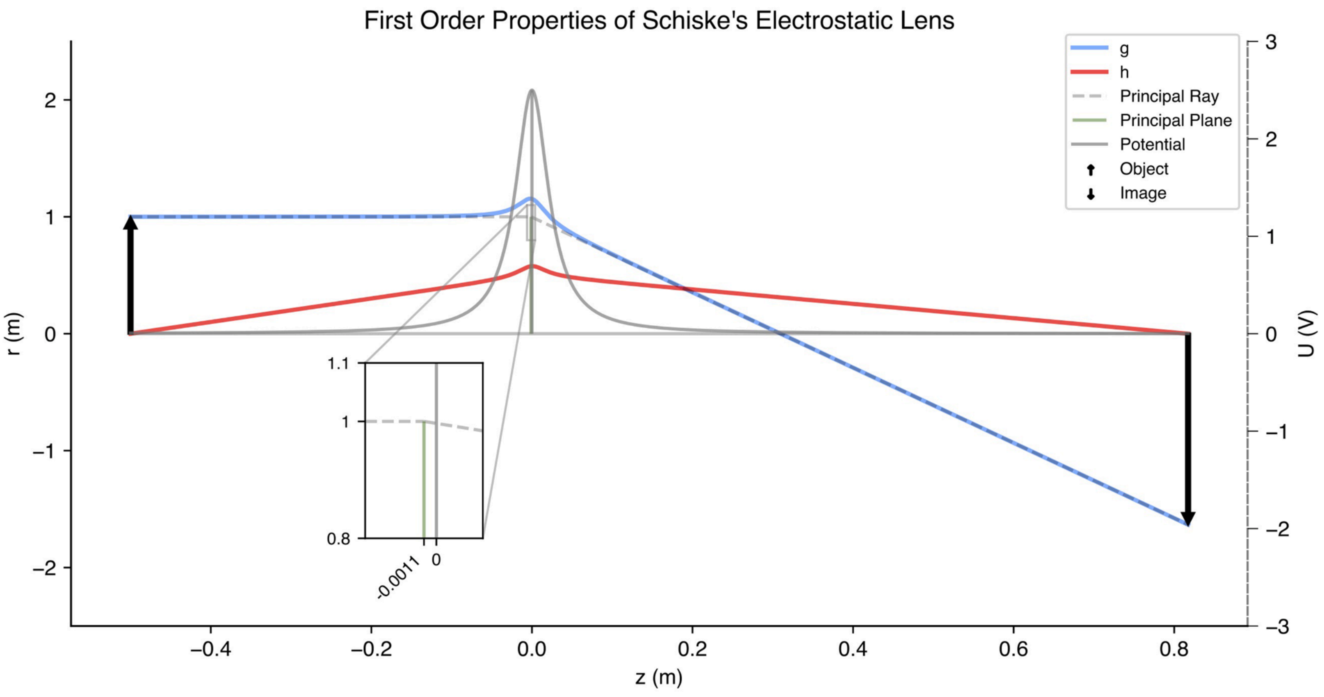
3. Linearised ordinary differential equation solutions of electric and magnetic fields

The series expansion of a round electron optical lens field, as shown by equation (1), enables the calculation of higher-order aberrations, but only the axial field function is needed to calculate the first-order properties of a lens. Combining the axial field function with the equations of motion for an electron in a magnetic or electrostatic field creates a linearised ordinary differential equation (ODE) (equation (7)) which, when solved, will calculate the first-order properties (focal length, magnification, principal image plane) of a lens [17,18] and provide a general solution for any ray path with a different initial position or slope in the plane of the object. The following function represents the non-relativistic linearised ODE for magnetic & electrostatic lenses:

$$x'' + \frac{\phi'(z)}{2\hat{\phi}}x' + \frac{\phi''(z) + \eta^2 B(z)^2}{4\phi_0}x = 0$$
$$y'' + \frac{\phi'(z)}{2\hat{\phi}}y' + \frac{\phi''(z) + \eta^2 B(z)^2}{4\phi_0}y = 0 \quad (7)$$

Where x & y are electron coordinates as a function of z. Primes represent a derivative w.r.t z, i.e. the slope of a ray. $\phi(z)$ and $B(z)$ are axial potential and field functions of a lens, respectively. ϕ_0 is again the acceleration voltage of the electron. η is given by $\sqrt{\frac{|e|}{2m}}$. $e \sim -1.60217663 \times 10^{-19}$ C, and $m \sim 9.1093837 \times 10^{-31}$ kg.

When solving these equations to obtain a general solution, two rays are traced with specific initial conditions through the electrostatic or magnetic field, denoted *g and h*. The conventional method noted in the literature proceeds as follows [18,24–26]: From a starting object position denoted z_0, trace a ray denoted g that starts at a radial position of 1 m, with a slope of 0, and when traced through the lens, will determine the focal length with respect to the centre of the lens. Calculation of a second ray denoted h, that begins with a slope of 1, and a position of 0 m at the object plane, traced through the lens determines the gaussian



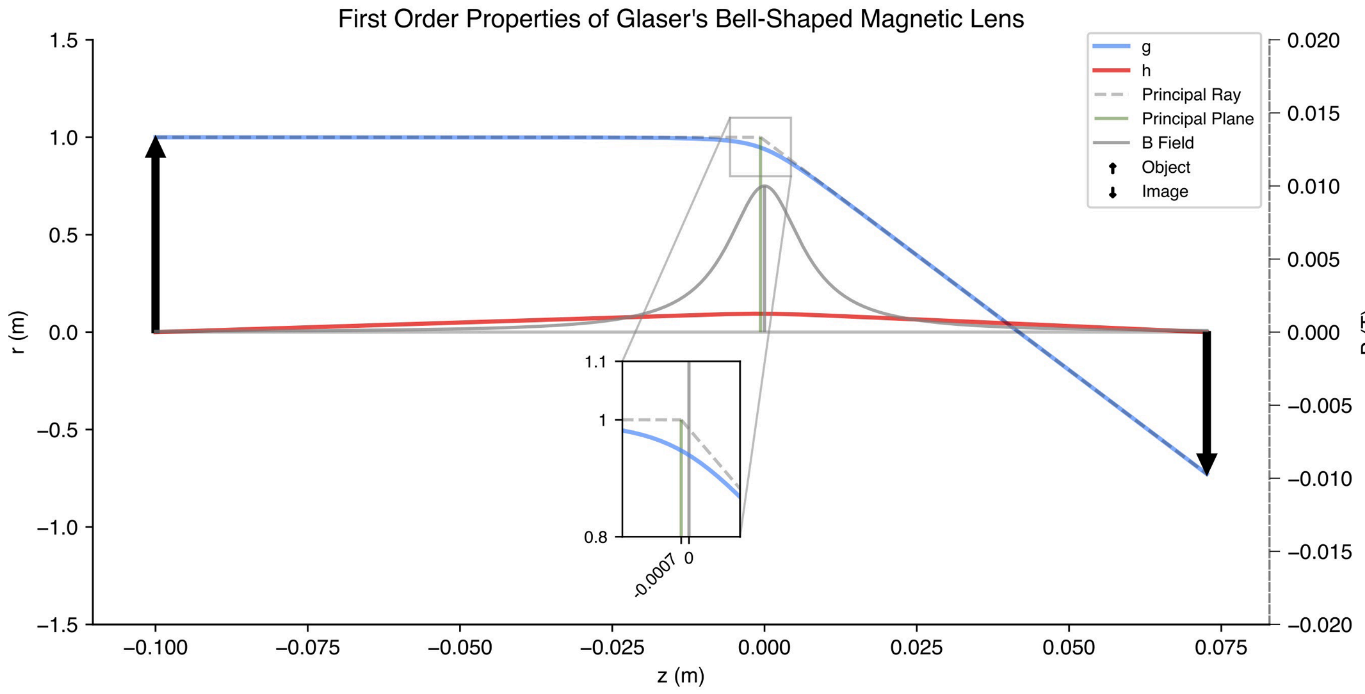


Fig. 1. Linearised ODE solutions to Schiske's Electrostatic Lens & Glaser's Bell Shaped Field. Shown are the rays g (blue) and h (red) and how the object (black up arrow) is imaged (black down arrow) by the lens. The inset shows the principal image plane (where the ray appears to come from) and its location behind the lens.

image plane. When the ray g is traced further to the gaussian image plane, its height determines the magnification of the image (see Fig. 1 for clarification). These two-ray solutions also form any other solution beginning with a different set of initial conditions, which has the practical use of enabling a full first-order description of the beam path with any set of initial conditions for a single starting object position. The solution rays g & h are also necessary when one wishes to calculate the aberration coefficients of an electron optical component, which is discussed further in Calculating aberrations of round lenses in Python.

4. Ordinary differential equation Solvers

The most common method chosen to solve the electron equations of motion through an electron optical lens is usually an adaptive step size Runge Kutta method [20,27]. The method chosen in this paper is the RK5(4) method with Dormand Prince coefficients (RKDP54) [28]. This method is an adaptive step size method which uses a fifth-order step to advance the calculation, and a fourth-order step to estimate the error. If the error is below a user-defined tolerance, the step size h of the algorithm is reduced until the error is below tolerance. The RKDP54 method contains certain drawbacks, as it requires seven force evaluations per step, which are the most expensive calculation when solving the equations of motion, and the method is not symplectic, meaning the kinetic energy of the system is not conserved. Thus, even though the error in the calculation seems low, an unphysical particle trajectory can result when solving the equations of motion. A symplectic integrator is of great importance in the field of charged particle tracing of nuclear fusion reactor designs, which requires tracing particles through magnetic fields for an extended time in a periodic orbit. Fortunately, the distances and time over which the electrons are propagated through electrostatic and magnetic fields (and the accuracy required) within the electron microscope are low enough such that kinetic energy drift does not become an issue. The linearised ODE (equation (7)) & the differential algebra method (see Lens aberration analysis *via* the Differential Algebra Method) presented solve the electron equations of motion *via* the RKDP54 method, which allow the calculation to be sped up significantly due to its adaptive step size feature.

However, when parallelising the ray tracing algorithm for real-time beam visualisation, the code uses the symplectic first-order Euler Cromer method, which only requires one force evaluation per propagation step. As a result, the Euler-Cromer method performs calculations significantly faster per step than the RKDP54 method, with the trade-off that the step size is non-adaptive and the global error rate is proportional to the step size $O(h)$. Furthermore, because the Euler-Cromer method is far easier to implement, the parallelised version of this software opts for the Euler-Cromer method when implementing the parallelised algorithm.

Fig. 1 shows the ray paths obtained when solving equation (7) with the analytical field equations of (5) and (6) for a magnetic and electrostatic lens, and the initial conditions of $g = 1,\ g' = 0$ & $h = 0,\ h' = 1$ to obtain the first order properties of each lens. The inset in each figure displays the principal image plane, where the ray appears to be deflected from if the lens was treated as a thin lens that performs a single deflection.

Table 1 shows the first-order properties of Glaser's Bell-shaped field and Schiske's Electrostatic field that are obtained when solving the linearised ODE *via* the RKDP54 method, and compares these results to the analytical solutions of the first-order properties presented in [19, 29]. Good agreement between both methods is found, which allows us to verify that the non-symplectic nature of the RKDP54 method does not significantly impact the path of the electron ray and the first-order properties. Results in Table 1 are presented with 15 significant figures.

5. Calculating aberrations of round lenses in Python

In a few cases, a straightforward analytic solution to the aberration coefficients of an analytical axial lens function can be found in the literature [29,30]. However, no such method usually exists, and microscopists must resort to other computational methods such as the aberration integral or differential algebra method [18,31]. The differential algebra method formulates the solution to the general electron equations of motion as a Taylor expansion, whose polynomial coefficients represent the aberration coefficients, and this method is available in MEBS & Cosy Infinity [7,32]. Another approach to obtaining aberration coefficients should also be mentioned which is implemented in the software EOD [8], which utilises a polynomial fitting method [33], although this work does not implement this method.

Table 1
Comparison of first-order properties calculated *via* analytical solutions against solving the linearised equations of motion.

	Glaser's Bell-Shaped Field	Schiske's Electrostatic Field
Analytic Focal Length	0.04241592856688	0.31079488270664
ODE Solution Focal Length	0.04241592856914	0.31079488270697
Relative Error	2.26×10^{-12}	3.30×10^{-13}
Analytic Magnification	-0.728140280754384	-1.63329918013632
ODE Solution Magnification	-0.728140280754499	-1.63329918013632
Relative Error	1.58×10^{-13}	0

This software implements both the aberration integral and differential algebra methods into TEMGYM Advanced and recreates aberration calculations found in the literature for the analytical potentials presented by M.Cheng *et al.* and Z.Liu [20,27]. Note that the methods presented here only calculate the real aberration coefficients. In the case of asymptotic aberration coefficients, which occur when either or both the object and image are located inside the lens field, then a different set of aberration integrals must be solved [18]. The differential algebra method implemented can only calculate the real aberration coefficients.

6. Aberration Polynomial

The polynomial form of the third-order geometric aberration coefficients of an electron optical system can be obtained *via* a variational method described by Hawkes [34] and can be written as follows:

$$\frac{x_i^{(3)}}{M} = \begin{bmatrix} x_0' & x_0 & -y_0 \end{bmatrix} Ar \tag{8}$$

with

$$A = \begin{bmatrix} B & 2F & D & 2f \\ F & 2C & E & c \\ f & c & e & 0 \end{bmatrix}$$

$$r = \begin{bmatrix} x_0'^2 + y_0'^2 \\ x_0 x_0'^2 + y_0 y_0'^2 \\ x_0^2 + y_0^2 \\ x_0 y'_0 + x_0 y'_0 \end{bmatrix}$$

where x_0 and x_o' are the position and slope of the ray at the object plane, M is the magnification of the image, $x_i^{(3)}$ is the third-order deviation of the ray from the ideal paraxial ray at the image plane, and the matrix A is composed of the individual aberration coefficients where B = Spherical Aberration, C = Isotropic Astigmatism, D = Field Curvature, E = Isotropic Distortion, F = Isotropic Coma, c = Anisotropic Astigmatism, e = anisotropic astigmatism, f = anisotropic coma. B, F, C, D and E are known as the five "Seidel Aberrations" [35], and a graphical representation of their meaning can be found here by J. Savard [36]. c, f and e are the anisotropic aberrations coefficients, and appear because of the rotation applied to the electron as it passes through a magnetic field. The values of each of the aberration coefficients for a particular starting position of the object on the z-axis can be obtained by solving the appropriate aberration integral. See textbooks from P.Hawkes, J.Orloff and M.Szilagyi [17,18,37] also for a more in depth overview. Numerous formulations of aberration integrals for round magnetic & electrostatic lenses can be found in the literature, and this software implements the aberration integrals printed by P. Hawkes [18] for magnetic lenses and those by Z. Liu [38] for electrostatic lenses. See the supplementary section for the specific equations implemented in this software.

Expanding equation (6) creates a polynomial expression which describes the third-order geometric aberrations:

$$\frac{x_i^{(3)}}{M} = Bx_0'^3 + Bx'_0\,y_0'^2 + 2Cx_0^2x'_0 + 2Cx_0y_0y'_0 + Dx_0^2x'_0 + Dy_0^2x'_0 + Ex_0^3 + Ex_0y_0^2 + 3Fx_0x_0'^2 + Fx_0y_0'^2 + 2Fx'_0y_0y'_0 + cx_0^2y'_0 - 2cx_0x'_0y_0 - cy_0^2y'_0 - ex_0^2y_0 - ey_0^3 + 2fx_0x'_0y'_0 - 3fx_0'^2y_0 - fy_0y_0'^2 \tag{9}$$

The software in TEMGYM Advanced combines the linearised ODE solutions, which return the principal rays *g* and *h*, with the symbolic package SymPy that calculates derivatives of the analytic potential or field to obtain the aberration integral. The software then utilises Simpson's rule implemented in the package SciPy [39] to solve the aberration integral numerically. It is important to note also that only the anisotropic coefficients c, f & e appear with a non-zero value for a magnetic lens.

7. Lens aberration analysis *via* the Differential Algebra method

The differential algebra method was introduced to electron optics by M.Berz in the particle accelerator physics community in 1990 [40] as a technique to calculate the aberrations of electron optical systems to machine precision. This implementation can be found in a program entitled "CosyInfinity" [32]. Eric Munro subsequently introduced the differential algebra method into the electron beam design software MEBS [7] and extended the method to calculate the aberration coefficients of electron mirrors [41]. Radlicka has made further use of the DA method to analyse the parasitic aberrations of a hexapole corrector [42].

As far as the authors are aware, our software TEMGYM Advanced represents the first time that the DA method for electron/ion optics has been used in an open-source project and made freely accessible. This is made possible by the freely accessible python wrapper for the C++ DA library, DaceyPy [15].

$$\frac{x_i^{(3)}}{M} = A_{3000}x_0^3 + A_{2010}x_0^2y_0 + A_{1020}x_0y_0^2 + A_{0030}y_0^3 + A_{0300}x_0'^3 + A_{0201}x_0'^2y'_0 + A_{0102}x'_0y_0'^2 + A_{0003}y_0'^3 + A_{1200}x_0x_0'^2 + A_{1101}x_0x'_0y'_0 + A_{1002}x_0y_0'^2 + A_{0210}y_0x_0'^2 + A_{0111}x'_0y_0y'_0 + A_{0012}y_0y_0'^2 + A_{2100}x_0^2x'_0 + A_{1110}x_0y_0x'_0 + A_{2001}y_0^2x'_0 + A_{0201}x_0^2y'_0 + A_{1101}x_0y_0y'_0 + A_{0201}y_0^2y'_0 \tag{12}$$

Differential algebra (DA), also known as truncated power series algebra (TPSA), is a method that can generate Taylor series expansions of an output expression as a function of its input. The underlying mathematical principles of differential algebra called "Dual Numbers" were first introduced by William Clifford [43], which enables the algebraic calculation of derivates and avoids the pitfalls of the finite difference method. Dual numbers are the mathematical backbone of automatic differentiation, enabling backpropagation at scale in many machine-learning packages such as JAX [44] & PyTorch [45]. DA adopts the dual number system such that any function of *n* variables can be expanded into its Taylor series. Expanding a function as a Taylor series *via* the DA method has an interesting application in the realm of aberration analysis in electron microscopy, as when the technique of DA is used to solve the ODE of a particle transmitting through a lens, it readily obtains the final coordinates of the particle as a Taylor expansion of the initial conditions, whose coefficients represent the aberration coefficients of the lens.

When computing a Taylor expansion, the number of coefficients σ associated with each variable *v* as the order of the expansion *n* increases is as follows -

$$\sigma = \frac{(n+v)!}{(n!v!)} \tag{10}$$

To describe a linear solution ($n = 1$) of an optical system *via* a Taylor expansion with four variables ($x, y, x', y', v = 4$), the number of Taylor coefficients, σ is equal to 5 for each variable. In the case of third-order geometric aberration coefficients, $n = 3$, and the number of total coefficients per variable (and thus the number of derivatives) increases to 35 according to (10), hence the importance of automatic differentiation & dual numbers in the DA method, which enables rapid and accurate calculation of functions that contain multiple higher order derivatives.

The relationship between the initial and final coordinate of the electron with Taylor series coefficients takes the following form:

$$_n\begin{bmatrix} x_f \\ x'_f \\ y_f \\ y'_f \end{bmatrix} = \sum_{i,\,j,\,k,\,l=0\sim n}^{i+j+k+l=n} x_0{}^i x_0'{}^j y_0{}^k y_0'{}^l{}_n \begin{bmatrix} A_{ijkl} \\ B_{ijkl} \\ C_{ijkl} \\ D_{ijkl} \end{bmatrix} \tag{11}$$

Expanding the terms of this series with $n = 3$ generates a polynomial whose coefficients are equal to the aberration coefficients obtained by solving the appropriate aberration integral.

For example, A_{0300} *and* A_{0102} in (12) are equivalent to *B* in (9). Coefficients which do not appear in (9) will have a value of zero in (12).

This work employs the Python package DaceyPy [15], a wrapper for the C++ differential algebra library "DACE" [46] that enables us to perform differential algebra in Python.

This software uses the trajectory equations of motion in a rotational coordinate system to perform the DA computation and obtain the aberration coefficients, and are given by: [47]

$$x'' = \frac{\rho^2}{2\phi}\left(\frac{d\phi}{dx} - (x' - \theta' y)\frac{d\phi}{dz}\right) + \frac{\eta\rho^2}{\sqrt{\phi}} \times \left(\rho\left(B_y - (y' + \theta' x)B_t\right) + F_x\right.$$
$$y'' = \frac{\rho^2}{2\phi}\left(\frac{d\phi}{dy} - (y' + \theta' x)\frac{d\phi}{dz}\right) + \frac{\eta\rho^2}{\sqrt{\phi}} \times \left(- \rho(B_x + (x' - \theta' y)B_t) + F_y\right. \tag{13}$$

with

$$F_x = 2\theta' y' + \theta' x + \theta'' y$$
$$F_y = -2\theta' x' + \theta' y - \theta'' x$$
$$\rho = \sqrt{1 + (x' - \theta' y)^2 + (y' + \theta' x)^2}$$
$$B_t = \frac{1}{\rho}\left[B_z + (x' - \theta' y)B_x + (y' + \theta' x)B_y\right]$$
$$\theta' = \frac{\eta B(z)}{2\sqrt{\phi_0}}$$
$$\theta'' = \frac{\eta B'(z)}{2\sqrt{\phi_0}}$$

where θ is the rotation angle of the rotating coordinate frame relative to the fixed frame, $B(z)$ is the magnetic axial field distribution, $\phi(z)$ is the electrostatic axial potential distribution equivalent to the following, and again, ϕ_0 is the initial acceleration voltage of the electron, and primes denote derivatives w.r.t. z.

In the case of an electrostatic lens, the rotational coordinate system is equivalent to the fixed coordinate system, as the electron does not rotate about the axis. When calculating aberrations, the rotational coordinate system is only relevant in the case of solving the electron trajectory in a magnetic lens, as it allows direct interpretation of the results without further calculation. If the object's location is inside the field of a magnetic lens, then an effect called Object Magnetic Immersion (OMI) [48] further modifies the aberration coefficient values. To account for the OMI effect, one must solve the electron trajectory in a fixed coordinate system and then transfer it to the rotational coordinate system at the end, which calculates the aberration coefficients with OMI included. TEMGYM Advanced has not included this step, but see Z. Liu [20] for more detail.

The main advantage of the differential algebra method is that it avoids the need to formulate and calculate a series of aberration integrals for an electron optical system to find the aberration coefficients. In the case of electrostatic & magnetic lenses, the aberration integrals are straightforward to formulate and solve up to third order, hence their use as verification of aberration coefficients calculated in our TEMGYM Advanced implementation. However, for more exotic systems involving multiple components and for higher order aberrations, this procedure becomes increasingly complicated [18].

The DA method is also comparatively simple to compute as only a single ray needs to be traced through a system from the object plane to the image plane to obtain the aberration coefficients to any desired order. In previously published works results have been described up to the fifth order [47].

The disadvantages are that the DA method requires an analytical representation of the electric or magnetic field on the optical axis. Requiring an analytical representation is a disadvantage because the finite element method calculates the field of practical electron optical lens designs (i.e. CAD models of a lens), which only returns a discrete set of values that describe the field along the optical axis. Therefore, one must either employ a suitable interpolation routine [49] or a fitting measure of the axial field [50,51] to enable the analytical determination of field values and their derivatives.

A Python-specific disadvantage is that the DA package DaceyPy is incompatible with the standard ODE solvers available in other popular python packages, such as SciPy's odeint. One must implement an ODE solver in pure Python to utilise this software, which will almost always be slower than the highly optimised routines implemented in languages such as Fortran.

Our programme TEMGYM Advanced is tested *via* two analytical axial fields for electrostatic & magnetic lenses, attempting to repeat results from M.Cheng et al. [27] and Z.Liu et al. [20], where the code in this work also calculates the gaussian image plane for each of these lenses *via* the linearised ODE solution (7), enabling us to know with certainty where to trace the ray to obtain the aberration coefficients at the object plane.

Reproducing some of the results found in the literature was challenging, with the necessary details required for a faithful reproduction of the work either missing or implied. For example, in the case of Z. Liu et al. [20], the gaussian image plane's location was omitted; to recreate their results, a user must know this value exactly. The method to calculate it is detailed by P. Hawkes [29], which requires solving an elliptic integral and the details of this calculation can be found on the GitHub repository associated with this publication [52].

Table 2 & Table 3 shows the results of our software which has repeated calculations of the aberration coefficients of Glaser's Bell-Shaped magnetic field and Schiske's Electrostatic field using the differential algebra method and the aberration integral method, with results presented to 15 significant figures. Recreating the results of previous publications to high precision to validate our implementation, and required utilisation of the RKDP54 method with an adaptive step-size. The RKDP54 method's adaptive step size feature significantly reduces the time required to calculate the electron path through Glaser's bell-shaped field, where the object plane location is at $z_0 \sim 0.034$ m,

Table 2
Comparison of aberration coefficients calculated *via* the aberration integral method, and calculated *via* the differential algebra method for Glaser's bell-shaped field.

Aberration Type	Glasers Bell-Shaped Field – Aberration Integral	Glasers Bell-Shaped Field – Differential Algebra	Relative Error
B	-122.772441936280	-122.772441936290	8.14×10^{-14}
F	-3240.95230904754	-3240.95230904682	2.22×10^{-13}
C	-74284.4192292685	-74284.4192292831	1.96×10^{-13}
D	-142976.368793181	-142976.368793201	1.40×10^{-13}
E	-3251938.05175000	-3251938.05199103	7.41×10^{-11}
c	-938.227788619450	-938.227788619490	4.25×10^{-14}
e	-43364.7964718281	-43364.7964718253	6.41×10^{-14}
f	-874382.646193524	-874382.646193516	1.01×10^{-14}

Table 3
Comparison of aberration coefficients calculated *via* the aberration integral method, and calculated *via* the differential algebra method for Schiske's electrostatic field.

Aberration Type	Schiske's Electrostatic Field – Aberration Integral	Schiske's Electrostatic Field – Differential Algebra	Relative Error
B	-122.772441936280	-122.772441936290	8.14×10^{-14}
F	-3240.95230904754	-3240.95230904682	2.22×10^{-13}
C	-74284.4192292685	-74284.4192292831	1.96×10^{-13}
D	-142976.368793181	-142976.368793201	1.40×10^{-13}
E	-3251938.05175000	-3251938.05199103	7.41×10^{-11}

and the gaussian image plane location is at $z_g \sim 34.0$ m. The distance that the code must trace the ray is of the order of 3×10^1 m, yet to achieve the required precision, a step size of $\sim 1 \times 10^{-6}$ m must be used when the electron is in the vicinity of the central field. The calculation will run significantly longer if it utilises a fixed step size because the lens field is located within a small window around 0 m. The higher-order adaptive step size of the RKDP54 method will complete the calculation with the appropriate step size when the field values are large, and as the ray exits the field, the step size will increase, enabling the completion of the calculation in a matter of seconds. It is important to note that the software in its current form can only calculate aberration coefficients of non-relativistic electrons. In future versions, we plan to include this ability in our ray-tracing routines. In the case of magnetic fields, this is trivial, as the kinetic energy of the electron in constant. In the case of electrostatic fields, this becomes slightly more involved, as the gamma factor will change as the electron moves through the electrostatic lens and this must be accounted for correctly.

8. Parallelised Electron Ray Tracing

TEMGYM Advanced utilises the high-performance python compilation package *numba* [53] to accelerate the time to solve the electron equations of motion through a series of electrostatic or magnetic fields and increases the number of rays that can be calculated simultaneously *via* parallelisation. Parallelisation utilises multiple threads on a CPU or GPU to perform many calculations simultaneously. The GitHub repository [52] associated with this work demonstrates the potential of this feature by presenting an example Jupyter notebook and two interactive models using the visualisation package PyQTGraph [54].

The Jupyter notebook demonstrates the capabilities of running parallelised code on a modern workstation CPU or GPU and gives users who wish to explore further a minimal working example to trace electron rays in parallel for their problem. Fig. 2 shows an example of tracing many rays in parallel through a single lens. Our hardware can trace 1024 rays through 0.02 m with 20000 steps (stepsize $= 1 \times 10^{-5}$) in a time of $\sim$ 84.8 ms $\pm$ 4.83 ms.

Parallelised ray tracing code such as this also enables the creation of basic interactive models which demonstrate some of the basic alignments of a TEM in a physically accurate manner. The first interactive model designed in PyQTGraph mimics the behaviour of a double deflection system and a magnetic lens, recreated *via* two saddle coils and a magnetic lens model *via* Glaser's bell-shaped field (Fig. 3).

Although in a TEM, the electron's velocity is usually so high that a single deflector can model a deflection in the centre of the component *via* the high energy approximation [55], this example illustrates the modelling of multiple magnetic components in succession using TEMGYM Advanced. Magnetic deflection systems can be modelled in many ways. For example, akin to a rotationally symmetric lens, our software can use equation (1) to model the field of a deflector dipole system everywhere in 3D space if the axial fields and their harmonics along the z-axis are known. Although Lencova has shown how to calculate the harmonic fields of a saddle and deflector coil [23], and these results could interface with the SymPy expression to expand equation (1), this code has opted instead to use an alternative representation.

The alternative method chosen recreates the shape of a saddle coil *via* a series of conducting wires and solves the Biot-Savart numerically using the open-source package bfieldtools [16], to obtain a 3D array of the magnetic field value surrounding the saddle coil. A linear interpolation routine then finds the field values at the required location to update the electrons' velocity and position. Once a ray has exited the field of a component and is in field-free space, it is trivial to propagate the distance to the next component in a straight line, thus saving computational cost to transport the electron between components. The arrangement of components in this model mimics a deflector lens system. The GitHub repository [52] contains an interactive model of a double-deflector lens system that demonstrates this software's real-time calculation capabilities.

The next interactive model recreates the astigmatism correction alignment of two quadrupole magnets and a magnetic lens. Akin to a dipole, the software models a quadrupole *via* a series of conducting wires. A quadrupole field focuses rays more strongly on one axis, enabling correction of the astigmatism aberration in a magnetic lens that

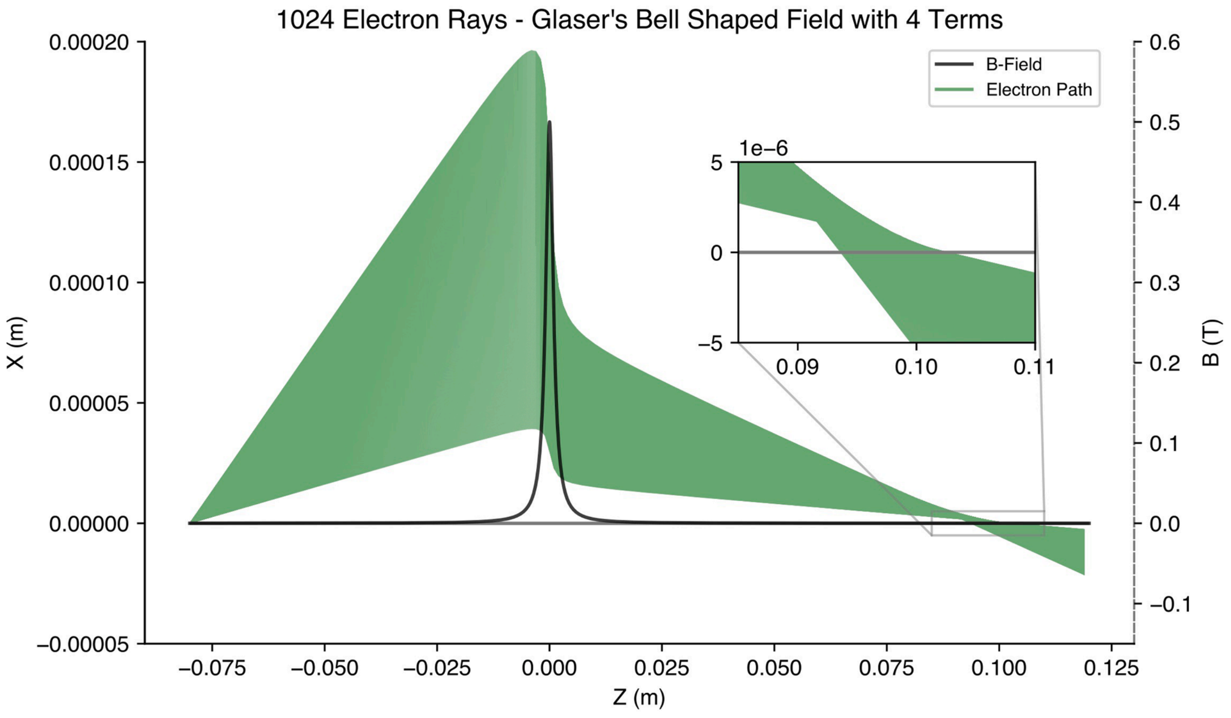
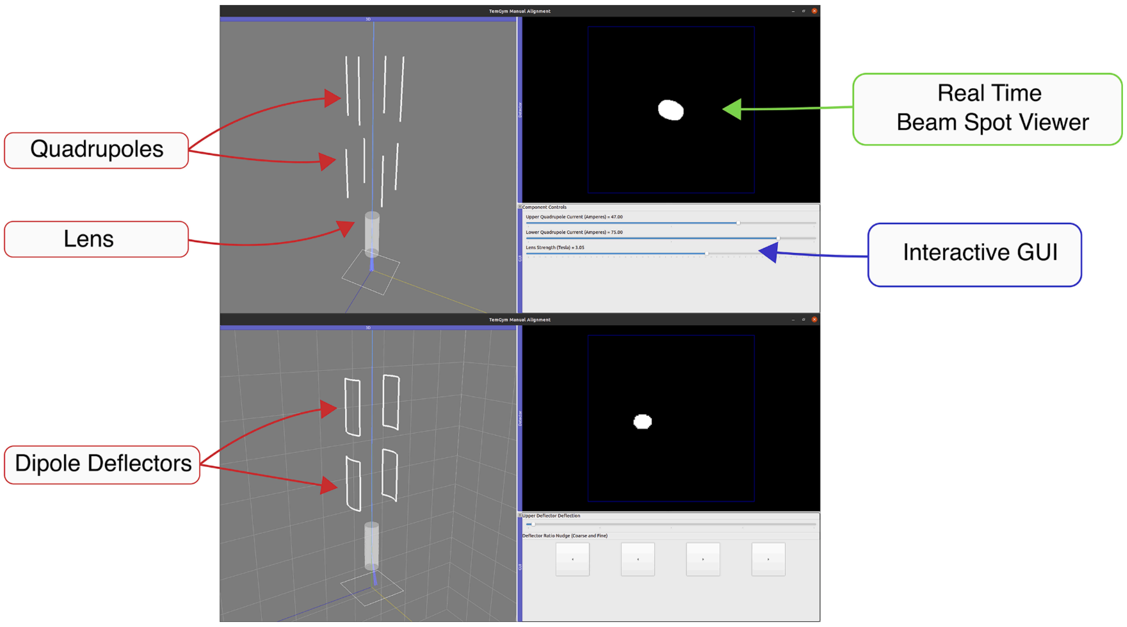


Fig. 2. 1024 electron rays calculated in parallel over 0.02 m and a step size of 10 μm. Inset shows the effect of aberrations on the beam when using multiple terms in Laplace's Expansion while solving the equation of motion.



Component Controls
Upper Quadrupole Current (Amperes) = 47.00
Lower Quadrupole Current (Amperes) = 75.00
Lens Strength (Tesla) = 3.05

Upper Deflector Deflection
Deflector Ratio Nudge (Coarse and Fine)

Fig. 3. Screenshot of interactive TEM models that perform real-time ray tracing in parallel through a series of components. Top – Double deflector system and lens, with the beam spot and GUI displayed on the right-hand side and the 3D model on the left. Bottom – Astigmatism correction system with two quadrupoles and a lens. The beam spot and GUI are displayed on the right-hand side, and the 3D model is on the left.

suffers from astigmatism. To create a toy model of astigmatism *via* Glaser's bell-shaped lens, the response of the field includes an angular component, which results in a different focal length on each axis, recreating an elliptical beam spot. Users can adjust the strength of each quadrupole to correct for the astigmatic Glaser's bell-shaped lens. This interactive demo calculates 2048 electron rays in parallel through two deflectors and a lens and includes a coarse and fine adjustment of the deflector ratio that enables the user to find the beam tilt/beam shift alignment in this example. The beam spot window displays the final location of all 2048 ray paths, but note that in the 3D window, the software only displays a subset of ray paths (512 electron rays) to keep the calculation fast. Further improvements to the software could enable it to display more electron ray paths.

Although these models are simplified representations, they demonstrate how to apply parallelisation with a modern CPU or GPU to electron microscopy. Typically, commercial microscope designs are unavailable to users; however, users can easily implement an axial field function and interface it with our software or use designs from an open-source microscope, such as the NanoMi project [56], that could pair with this code. Access to the designs of an electron microscope would more easily enable the creation of a complete digital twin of a microscope *via* our software. A digital twin of a TEM that can accurately calculate the electron beam path and generate a realistic beam spot in near real-time has the potential for many exciting applications in the future. For example, it could enable exploration of automated control of the base alignment steps *via* machine learning, akin to how many researchers are using digital twin models of many complicated systems such as Tokamak reactors to achieve state-of-the-art automated control in complicated environments where classic control methods struggle to perform [57–60]. A digital twin simulation also has value as an interactive teaching tool, helping to train the next generation of microscopists by providing them with a behind-the-scenes view of how an electron microscope behaves. Furthermore, the methods presented in this paper can be used to calculate the first order properties of real electron optical lenses, which can then be used to create a first-order microscope model *via* the code presented in TEMGYMBasic [12].

9. Conclusion

A suite of open-source tools that model and characterise analytic fields of electron microscope lenses has been presented, which enables the performance analysis of lens designs and the visualisation of a beam spot in real time. The software demonstrates how users can use the expansion of an electron lens potential/field with methods to solve the electron equations of motion to calculate the first-order lens properties or third geometric aberration coefficients. Two methods have been implemented to calculate aberration coefficients *via* the DA and the aberration integral method, enabling us to verify each algorithm's results. The software has also reproduced the aberration coefficient results of Glaser's Bell-Shaped Field and Schiske's Electrostatic lens from previously published results. Furthermore, this paper presents a method to achieve parallelised ray tracing of an electron beam through a series of electron optical components, generating an electron beam spot with third-order aberrations in a model electron microscope in near real-time. These models help to move us closer towards a realisation of a digital twin of the electron microscope while doing so with an open-source ethos that enables reproducible and verifiable results.

TEMGYM Advanced demonstrates how the rich ecosystem of packages available in Python enable rapid prototyping and development. For example, the packages available in Python enable symbolic equation manipulation *via* SymPy [14], hardware acceleration of functions that parallelise and speed up the code *via* Numba [53], calculation of magnetic fields surrounding a wire *via* bfieldtools [16], fast calculation of Taylor expansions *via DaceyPy* [15], and development of a basic GUI *via* PyQTGraph [54]. In the current iteration of this software, we have focused on applying our methods to the most basic of microscope components operating in TEM mode, and there is a rich number of ways to improve and expand upon the code presented in this paper. The software could also easily be adapted to calculate aberrations and electron paths through a microscope operating in STEM mode. TEMGYM Advanced could also calculate the aberration coefficients of practical lens designs whose field was calculated *via* FEM methods and enable the calculation of aberration coefficients for relativistic electrons, and future work will focus on completing this goal. It is also possible to calculate fifth-order geometric and chromatic aberration coefficients with relatively few changes. With more effort, the software will be able to model more exotic microscope components such as Wien filters, electron mirrors, aberration correctors and an electron gun, and we plan to also include these features in future versions of the software.

Code availability

The code associated with this paper is available in the corresponding GitHub repository [52]: https://github.com/AMCLab/TEMGYM Advanced

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.

Data availability

No data was used for the research described in the article.

Acknowledgements

The research was supported by the Irish Research Council EPSPG/2019/509 and the Ernst Ruska-Centre for Microscopy and Spectroscopy with Electrons (ER-C).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ultramic.2023.113738.

References

[1] A. Clausen, et al., LiberTEM: Software platform for scalable multidimensional data processing in transmission electron microscopy, J. Open Source Softw. 5 (2020) 2006.
[2] Johnstone, D. N. et al. pyxem/pyxem: pyxem 0.10.0. (2019). 10.5281/ZENODO.3533653.
[3] F. De La Peña, et al., hyperspy/hyperspy: Release v1.6.5, zndo (2021), https://doi.org/10.5281/ZENODO.5608741.
[4] M. Ziatdinov, A. Ghosh, C.Y.(Tommy) Wong, S.V. Kalinin, AtomAI framework for deep learning analysis of image and spectroscopy data in electron and scanning probe microscopy, Nat. Mach. Intell. 2022 (2022) 1–12, https://doi.org/10.1038/s42256-022-00555-8.
[5] T. Kluyver, et al., Jupyter Notebooks – a publishing format for reproducible computational workflows, in: Position. Power Acad. Publ. Play. Agents Agendas - Proc. 20th Int. Conf. Electron. Publ. ELPUB 2016, 2016, pp. 87–90, https://doi.org/10.3233/978-1-61499-649-1-87.
[6] M.D. Wilkinson, et al., The FAIR Guiding Principles for scientific data management and stewardship, Sci. Data 3 (2016) 1–9, 2016 31.
[7] E. Munro, Numerical simulation methods for electron and ion optics, Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip. 645 (2011) 266–272.
[8] B. Lencová, J. Zlámal, A new program for the design of electron microscopes, Phys. Procedia 1 (2008) 315–324.
[9] P. McBean, P. Murphy, R. Sagawa, L. Jones, The User Adjustable Pole-piece: Expanding TEM Functionality Without Compromise, Microsc. Microanal. 28 (2022) 2636–2638.
[10] T. Harvey, C. Ophus, Automated Design of Electron Mirrors for Multipass Electron Microscopy and 4D-STEM+EELS, Microsc. Microanal 28 (1927) 2022.
[11] R. Ciancio, et al., e-DREAM: the European Distributed Research Infrastructure for Advanced Electron Microscopy, Microsc. Microanal 28 (1927) 2022.
[12] Landers, D., Clancy, I., Dunin-Borkowski, R. E., Weber, D. & Stewart, A. TEMGYM Basic: A Transmission Electron Microscopy ray tracing software for training and progress towards a digital twin. Submitted.
[13] C.R. Harris, et al., Array programming with NumPy, Nat. 585 (2020) 357–362, 2020 5857825.
[14] A. Meurer, et al., SymPy: Symbolic computing in python, PeerJ Comput. Sci. 2017 (2017) e103.
[15] giovannipurpura/daceypy: Python wrapper of DACE, the Differential Algebra Computational Toolbox. Available at: https://github.com/giovannipurpura/daceypy. (Accessed: 21st December 2022).
[16] A.J. Mäkinen, et al., Magnetic-field modeling with surface currents. Part I. Physical and computational principles of bfieldtools, J. Appl. Phys. 128 (2020), 063906.
[17] M. Szilagyi, Motion of Charged Particles in Electric and Magnetic Fields, Electron Ion Opt. (1988) 13–50, https://doi.org/10.1007/978-1-4613-0923-9_2.
[18] H. Peter, K. Erwin, Principles of Electron Optics, Basic Geometrical Optics, Elsevier, 2018.
[19] M. Cheng, T. Tang, Y. Zhenhua, Study on differential algebraic aberration method for electrostatic electron lenses, Optik (Stuttg) 112 (2001) 250–254.
[20] Z. Liu, On the map method for electron optics, Institute of Physics Conference Series 175 (2005) 185–192.
[21] W. Glaser, Strenge Berechnung magnetischer Linsen der Feldform, Zeitschrift für Phys 117 (1941) 285–315.
[22] P. Schiske, An Electrostatic Single Lens permitting Rigorous Calculation, Nat. 171 (1953) 443–444, 1953 1714349.
[23] M. Lenc, B. Lencovà, Analytical and numerical computation of multipole components of magnetic deflectors, Rev. Sci. Instrum. 68 (1998) 4409.
[24] M.De Graef, Introduction to Conventional Transmission Electron Microscopy, Introd. to Conv. Transm. Electron Microsc. (2003), https://doi.org/10.1017/CBO9780511615092.
[25] M. Szilagyi, Electron and Ion Optics, Electron Ion Opt (1988), https://doi.org/10.1007/978-1-4613-0923-9.
[26] El-Kareh, A. *Electron beams, lenses, and optics*. (2012).
[27] M. Cheng, T. Tang, Z Yao, Study on differential algebraic chromatic aberration method for Glaser's bell-shaped magnetic lenses, Opt 112 (2001) 483–486.
[28] J.R. Dormand, P.J Prince, A family of embedded Runge-Kutta formulae, J. Comput. Appl. Math. 6 (1980) 19–26.
[29] P.W. Hawkes, E.(Erwin) Kasper, Principles of electron optics, Applied geometrical optics Volume 2 (1989).
[30] P.W. Hawkes, The relation between the spherical aberration and distortion coefficients of electron probe-forming and projector lenses, J. Phys. D. Appl. Phys. 1 (1968) 1549–1558.
[31] A.J. Dragt, E Forest, Lie Algebraic Theory of Charged-Particle Optics and Electron Microscopes, Adv. Electron. Electron Phys. 67 (1986) 65–120.
[32] K. Makino, M. Berz, COSY INFINITY Version 9, Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip. 558 (2006) 346–350.
[33] M.A.R. Krielaart, P. Kruit, Flat electron mirror, Ultramicroscopy 220 (2021).
[34] Hawkes, P. W. Magnetic electron lenses. 462 (1982).
[35] L. Seidel, Seidel, L. Zur Dioptrik, Über die Entwicklung der Glieder 3ter Ordnung welche den Weg eines ausserhalb der Ebene der Axe gelegene Lichtstrahles durch ein System brechender Medien bestimmen, vo Herrn Dr. L. Seidel, AN 43 (1856) 289.
[36] Savard, J. J. G. The Five Seidel Aberrations. Available at: http://www.quadibloc.com/science/opt0505.htm. (Accessed: 3rd February 2023).
[37] J. Orloff, Handbook of charged particle optics, CRC Press/Taylor & Francis, 2009.
[38] Z. Liu, Differential algebraic method for aberration analysis of typical electrostatic lenses, Ultramicroscopy 106 (2006) 220–232.
[39] P. Virtanen, et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (2020) 261–272, 2020 173.
[40] M. Berz, Computational aspects of optics design and simulation: COSY INFINITY, Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip. 298 (1990) 473–479.
[41] L. Wang, J. Rouse, E. Munro, H. Liu, X Zhu, Aberration analysis of electron mirrors by a differential algebraic method, Opt. - Int. J. Light Electron Opt. 119 (2008) 90–96.
[42] T. Radlička, Correction of parasitic aberrations of hexapole corrector using differential algebra method, Ultramicroscopy 204 (2019) 81–90.
[43] C. Clifford, Preliminary Sketch of Biquaternions, Proc. London Math. Soc. s1-4 (1871) 381–395.
[44] google/jax: Composable transformations of Python+NumPy programs: differentiate, vectorize, JIT to GPU/TPU, and more. Available at: https://github.com/google/jax. (Accessed: 21st December 2022).
[45] Paszke, A. et al. Automatic differentiation in PyTorch. (2017).
[46] dacelib/dace: Differential Algebra Computational Toolbox. Available at: https://github.com/dacelib/dace. (Accessed: 21st December 2022).
[47] Z. Liu, Differential algebraic description for third- and fifth-order aberrations of electromagnetic lenses, Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip. 519 (2004) 154–161.
[48] J. Ximen, Z Liu, Third-order geometric aberrations in Glaser's bell-shaped magnetic lens for object magnetic immersion, Opt 111 (2000) 355–358.
[49] Y. Kang, T. Tang, Y. Ren, X. Guo, Differential algebraic method for computing the high order aberrations of practical electron lenses, Optik (Stuttg) 118 (2007) 158–162.
[50] M. Berz, Modern map methods for charged particle optics, Nucl. Inst. Methods Phys. Res. A 363 (1995) 100–104.

[51] L. WANG, Simulation of electron optical systems by differential algebraic method combined with Hermite fitting for practical lens fields, Microelectron. Eng. 73–74 (2004) 90–96.
[52] AMCLab/TEMGYMAdvanced. Available at: https://github.com/AMCLab/TEMGYMAdvanced. (Accessed: 3rd February 2023).
[53] Lam, S. K., Pitrou, A. & Seibert, S. Numba: A LLVM-based Python JIT Compiler. *Proc. Second Work. LLVM Compil. Infrastruct. HPC - LLVM '15* 10.1145/2833157.
[54] Moore, O. & Campagnola, L. Introduction — pyqtgraph 0.12.4.dev0 documentation. Available at: https://pyqtgraph.readthedocs.io/en/latest/introduction.html. (Accessed: 13th September 2022).
[55] G. Pozzi, Particles and Waves in Electron Optics and Microscopy, Adv. Imaging Electron Phys. 194 (2016) 1–354.
[56] M. Malac, et al., NanoMi: An open source electron microscope hardware and software platform, Micron 163 (2022), 103362.
[57] J. Degrave, et al., Magnetic control of tokamak plasmas through deep reinforcement learning, 414 | Nat 602 (2022).
[58] P.R. Wurman, et al., Outracing champion Gran Turismo drivers with deep reinforcement learning, Nat. 602 (2022) 223–228, 2022 6027896.
[59] M.G. Bellemare, et al., Autonomous navigation of stratospheric balloons using reinforcement learning, Nat. 588 (2020) 77–82, 2020 5887836.
[60] O. Vinyals, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 575 (2019) 350–354.