ARBTools: A tricubic spline interpolator for three-dimensional scalar or vector fields.

Walker, Paul[1,*], Krohn, Ulrich[1], Carty, David[2]

1 Department of Physics, Durham University, South Road, Durham, DH1 3LE, United Kingdom.
2 Department of Chemistry, Durham University, South Road, Durham, DH1 3LE, United Kingdom

*** paul.a.walker@durham.ac.uk**

Abstract

ARBTools is a Python library containing a Lekien-Marsden type tricubic spline method for interpolating three-dimensional scalar or vector fields presented as a set of discrete data points on a regular cuboid grid. ARBTools was developed for simulations of magnetic molecular traps, in which the magnitude, gradient and vector components of a magnetic field are required. Numerical integrators for solving particle trajectories are included, but the core interpolator can be used for any scalar or vector field. The only additional system requirements are NumPy.

Keywords

Python, three-dimensional interpolation, spline, vector field, scalar field, smoothing

Introduction

It is often necessary to smoothly interpolate vector or scalar fields known as a set of discrete data points across a grid. For one- and two-dimensional problems cubic and bicubic spline implementations exist (for example, in the SciPy interpolate library), but three-dimensional problems are more difficult. Solving the three-dimensional motion of neutral particles through a magnetic potential, such as a Zeeman decelerator or magnetic trap, requires knowledge of the derivatives of the potential field. These fields can generally not be analytically solved and are calculated using *e.g.* finite element analysis and interpolated. Commonly available interpolation packages do not give access to these derivatives. Additionally, it may be desired to return the interpolated vector components of the field, for example when calculating the probabilities of non-adiabatic spin transitions, or simulating laser-cooling interactions.

A tricubic spline interpolator of the type described by Lekien and Marsden [1] was implemented using Python, and requires only NumPy to work. Interpolation coefficients are calculated on-the-fly and subsequently reused where required to reduce processor time. For arbitrary points inside the interpolation volume the field magnitude, partial derivatives and vector components are readily accessible from a single query. Separate query methods are included for dealing with interpolation of a single point, or for multiple simultaneous

 doi:10.20944/preprints201903.0029.v1

coordinates. Fourth-order Runge-Kutta and velocity Verlet algorithms are implemented for solving particle motion.

Although produced for the specific application of modelling low-field-seeking neutral particles, this software has been developed to be more general. It can work directly with either a scalar or vector field input, and is suitable for a variety of applications with any field supplied across a regular, paralleliped grid.

An overview of the software, how it was produced, and the research for which it has been used, including references to relevant research articles. A short comparison with software which implements similar functionality should be included in this section.

Implementation and architecture

ARBTools is written in Python [2], with extensive use of NumPy [3]. 'ARBInterp' contains the tricubic interpolator and query methods. 'ARBTraj' contains numerical integrators and acceleration functions for modelling the trajectories of paramagnetic neutral particles in a magnetic field; replacing the acceleration functions as appropriate could allow a different system to be solved, *e.g.* movement of atoms in an optical tweezer.

Installation

To install on Linux run 'sudo python setup.py install'. The interpolator is contained in a file called 'ARBInterp.py' and the command 'from ARBTools.ARBInterp import tricubic' will import the interpolation class.

Usage

To instantiate the class, pass it a source field - *e.g.* 'interp = tricubic(sourcefield)' will create an instance called 'interp'. Input can be either a scalar field U(x,y,z) as an N x 4 (x,y,z,U) array or a vector field B(x,y,z) as an N x 6 (x, y, z, Bx, By, Bz) array. If an N x 4 field is passed, the interpolator will automatically default to return the magnitude and gradient of the field. If an N x 6 field is passed it will accept an optional 'mode' keyword argument to select one of three modes, (*e.g.* interp = tricubic(sourcefield, mode='kw')):

- Norm: takes the norm of the vector field and return the magnitude and gradient (as three partial derivatives)
- Vector: returns the interpolated vector components
- Both: takes vector norm, and returns the magnitude and norm of the vector plus the vector components at the interpolation point

If no keyword is passed, the interpolator defaults to vector mode. Two query modes are implemented: 'sQuery' interpolates a single point within the volume, accepting an input in the form ($[x, y, z]$). 'rQuery' accepts a range of coordinates for simultaneous interpolation, as an array ($[x_1, y_1, z_1]...[x_n, y_n, z_n]$). For multiple queries rQuery is much more efficient than running sQuery in a loop.

Figure 1 shows an interpolation example. The quadrupole electric field produced by four point charges was calculated as a grid of 400^3 data points; the left plot is a 2D slice through the central plane. A less dense grid of 40^3 points was then calculated, and the middle image shows a plot through the centre. Lastly, the sparse grid was interpolated to reproduce the 400^3 data, and is shown on the right.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 March 2019 doi:10.20944/preprints201903.0029.v1

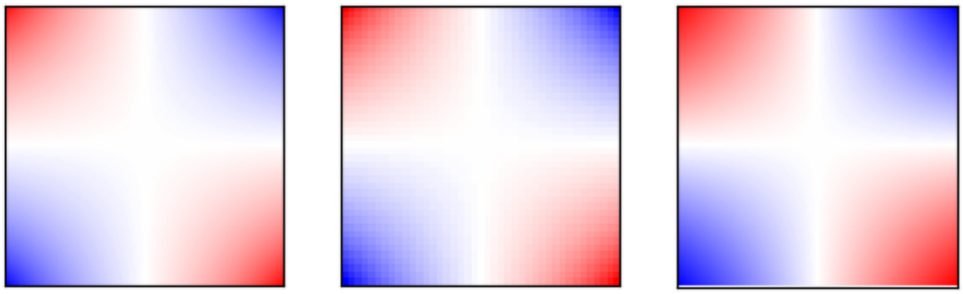

Figure 1. The magnitude of a quadrupole electric field, left, 400 x 400 pixel analytic solution, centre, 40 x 40 pixel exported subset, right, 400 x 400 pixel interpolation of the subset.

Quality control

ARBTools was written with Python 2.7.12 and NumPy 1.13.3 on Linux Mint 18.3, and has been tested with Python 3.5.2 on the same platform. It has also been tested on Enthought Canopy v2.1.9 on Microsoft Windows 7 and 10.

Example input fields and query scripts are available to download from the source repository. Performance benchmarking on 64-bit Linux with an Intel Core i7 CPU shows 100 unique interpolations for a given data set take between 20 and 50 ms, depending on which components are being returned. As expected, there is a linear relationship between number of queries and run time.

The main constraint when using ARBTools is the amount of memory required to load the source file; this is determined by the size of the input grid. For example; a cubic volume 20 mm on a side with a grid spacing of 0.5 mm contains $41^3 = 68921$ grid points, which will load in less than a second with negligible memory usage. The same data sampled at 0.25 mm intervals contains 531441 points, this may take several seconds to load and consume $\approx$ 500 MB memory. At 0.125 mm intervals we have 4173281 points, this may take up to a minute to load and consume over 5 GB of memory. Once loaded, however, querying these different datasets takes almost exactly the same amount of time.

The tricubic interpolation method values smoothness of the interpolated function and its first derivatives over absolute accuracy [1]. In order to quantify the errors in this method two types of model were considered; the quadrupole electric field produced by a series of point charges, which can be solved analytically (figure 1), and a magnetic field produced by a permanent magnet, calculated using finite-element analysis with the 'FEMM' [4] software package (see figure 2). (Of course, if an analytic solution is available there is no need to interpolate - this is simply a useful calibration tool!)

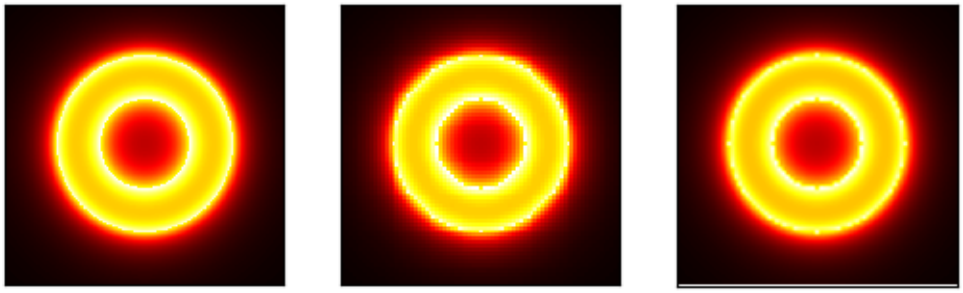

Figure 2. The magnitude of the magnetic field around a ring magnet, left, 400 x 400 pixel finite-element analysis model, centre, 40 x 40 pixel exported subset, right, 400 x 400 pixel interpolation of the subset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 March 2019 doi:10.20944/preprints201903.0029.v1

For both cases a high-resolution source dataset was created, and then a sparse subset of this data was interpolated and compared with it. Figure 3 shows the root-mean-squared errors between the interpolated and 'true' values of the calculated fields for a variety of grid intervals. It can be seen that for a given level of accuracy the analytic solution can tolerate a larger grid spacing - this is due to the high gradients at the interface between two materials in finite-element (or boundary volume integral [5]) analysis. In general, consideration of the nature of the data set being interpolated and its structure will inform the grid spacing chosen, which is a compromise between inaccuracy and unwieldiness. These tests were repeated with the interpolator in the 'EQ Tools' library; although it does not provide the field derivatives, the magnitudes were found to be the same to within 1×10^{-6} %.

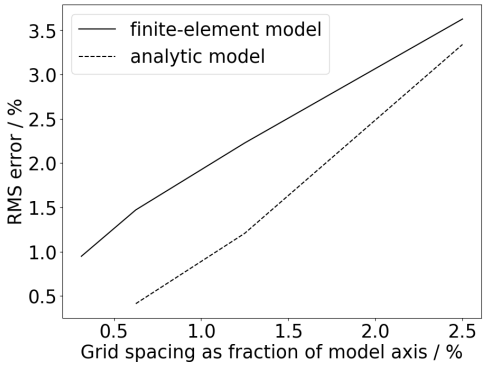

Figure 3. Root-mean-square error in interpolated data as compared to 'true' values from either a finite-element analysis model or an analytic solution.

(2) Availability

Operating system

ARBTools was developed on Linux Mint 18.3, and has been tested on Windows 7 and 10.

Programming language

ARBTools was developed in Python 2.7.12 and has been tested on 3.5.2. Any version of Python from 2.7 upwards should be suitable.

Additional system requirements

Several GB of RAM should be suitable for most applications. ARBTools has been used with large datasets on the Durham university supercomputer.

Dependencies

Written using NumPy 1.13.3. Earlier versions may work.

Software location:

- Name: ARBTools
- Persistent identifier: https://doi.org/10.5281/zenodo.2548609
- Publisher: Paul A. Walker
- Version published: v1.3
- Repository: GitHub
- Persistent identifier: https://github.com/DurhamDecLab/ARBInterp
- Licence: GPL-3.0
- Date published: 15/02/2019

Language

English

(3) Reuse potential

The core of ARBTools is the tricubic interpolator, which can be used with any suitably-formatted input field, for many possible tasks - for example, visualising the shape of a three-dimensional potential or extracting coherent Lagrangian structures from a time-dependent two-dimensional flow.

As is, ARBTools can be used to model the trajectories of low-field-seeking argon atoms in a magnetic field. Simply altering the mass and magnetic moment parameters would allow other species to be modelled. If the functions defining the acceleration due to a potential are replaced, trajectories in alternative systems could easily be modelled, for example, the motion of charges in electric fields, or masses moving under gravity.

Support may be requested through the project GitHub page:

(https://github.com/DurhamDecLab/ARBInterp). The source code is available and may be reused or modified at will subject to the details of the GPL-3.0 licence.

Acknowledgements

Many thanks to Dr. Lewis McArd for his invaluable advice on this and other projects.

Funding statement

This software was developed as part of research funded by EPSRC grant number EP/N509462/1.

Competing interests

“The authors declare that they have no competing interests.”

References

1. Lekien, F. and Marsden, J. (2005) 'Tricubic interpolation in three dimensions', *International Journal for Numerical Methods in Engineering*, Vol. 63, No. 3, pp. 455–471 . DOI: 10.1002/nme.1296

2. van Rossum, G. (1995) Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam

3. van der Walt, S., Colbert, S. C., and Varoquaux, G. (2011) 'The NumPy array: a structure for efficient numerical computation', *Computing in Science and Engineering*, Vol. 13, pp. 22-30, DOI:10.1109/MCSE.2011.37

4. 'Finite Element Method Magnetics' (2019), *Meeker, D.* . Available at: http://www.femm.info/wiki/HomePage

5. Elleaume, P., Chubar, O., Chavanne, J. (1997) 'Computing 3D Magnetic Field from Insertion Devices', *proc. of the PAC97 Conference*, pp. 3509-3511.

6. Chilenski, M.A., Faust, I.C. and Walk, J.R. (2017) 'eqtools: Modular, extensible, open-source, cross-machine Python tools for working with magnetic equilibria', *Computer Physics Communications*, Vol. 210, pp. 155-162, DOI:10.1016/j.cpc.2016.09.011.