The UNIVERSITY of OKLAHOMA Health Sciences Center

Mapping Magnetic Forces

[using the Maxwell stress tensor]

Alisha Shutler, M.S.
Isaac Rutel, Ph.D., DABR
University of Oklahoma Health Sciences Center

[abstract]

By understanding how to manipulate the equivalent-force vectors of a magnetic field space, we can move towards field shaping techniques that apply the force gradient of our choosing to a point in space. We can use this with targeted drug delivery therapies involving superparamagnetic iron oxide nanoparticles (SPION) in order to provide a non-invasive technique to isolate the drug to target tissues.

We demonstrate computation of the force equivalent vector field by applying a simplified Maxwell stress tensor to a 3D magnetic field space (setting electric fields to zero). From this generalized force equation, we then apply the tensor and compute the forward derivative to arrive at the equivalent force. To compute the gradient, we implement a next-next nearest neighbor (NNNN) method to increase computational efficiency while performing computations in all directions. Data handling techniques for large amounts of vector field spaces are also explored. The method is demonstrated by performed calculations across a simulated magnetic field space for uni and multi-directional fields.

[contact]

Alisha Shutler, M.S.
University of Oklahoma Health Sciences Center
Department of Radiological Sciences
College of Medicine
P.O. Box 26901
Garrison Tower, Suite 4G4250
Oklahoma City, Oklahoma 73126-0901
: ashutler@oushc.edu
: (405) 633-3732
: ouhsc.edu

what is the [Maxwell stress tensor]?

Physically, **T** describes pressures and shears on a volume of charges through the fields on the surface:

$$T_{ij} = \epsilon_0\left(E_iE_j - \frac{1}{2}\delta_{ij}E^2\right) + \frac{1}{\mu_0}\left(B_iB_j - \frac{1}{2}\delta_{ij}B^2\right)$$

which can be expanded and simplified by assuming E = 0:

$$T_{xyz} = \frac{1}{2\mu_0}\begin{bmatrix} B_x^2 - B_y^2 - B_z^2 & 2B_xB_y & 2B_xB_z \\ 2B_yB_x & B_y^2 - B_z^2 - B_x^2 & 2B_yB_z \\ 2B_zB_x & 2B_zB_y & B_z^2 - B_x^2 - B_y^2 \end{bmatrix}$$

Force is then given as:

$$\boldsymbol{F} = \nabla \cdot \boldsymbol{T}$$

[implementing] the algorithm

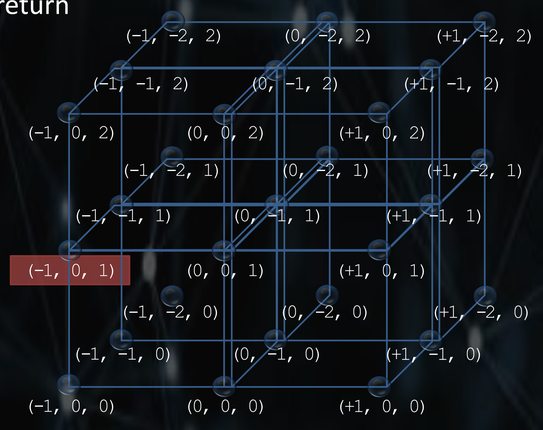
Applying the *tensor* to a collected data sets requires a physical setup with a **3 axis gaussmeter** measurements taken at a known step size between points along each axis. In order to handle this data and compute the contribution to force across each axis, several techniques are implemented:

[defining a coordinate system]

Absolute positions from the stepper motors are converted into an integer-based coordinate system. By converting these experimentally defined spatial dimensions to indexed values, the defined experimental values are selectable and variable, but the code still performs the proper calculations.

[sorting for values]

MATLAB's built-in *linear indexing* feature is one way to easily sort through matrices and recall values (or coordinates). The dimensions have been set to general indices by associating a coordinate set as a single index value. When searching for a particular coordinate set, linear indexing will then return a *single, integer value*, describing the location of the coordinate set within the general matrix. Using this method, the tensor can be applied to any coordinates without re-sorting the integers.



The single linear index value can point to both the location of the coordinates and their corresponding magnetic field values. This is due to the values being linked via a [cell] structure.

index (P(Bx, By, Bz)) = 993
P(Bx, By, Bz) = (-1, 0, 1)

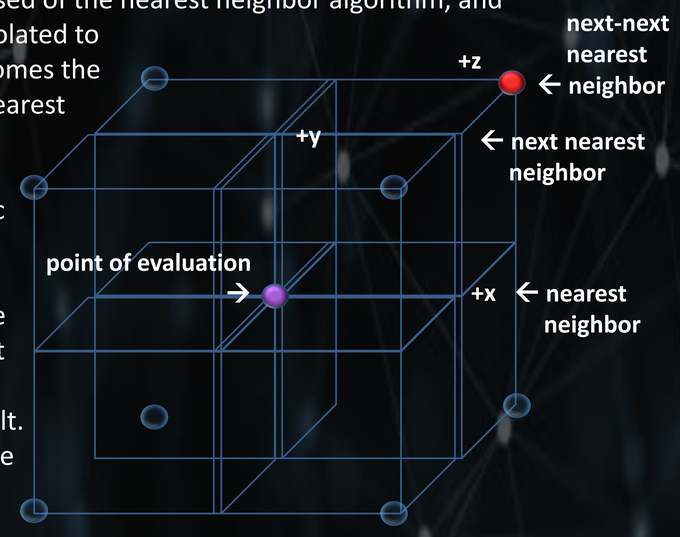
[linking positions with magnitudes]

Utilizing MATLAB's [cell] structure, the x, y, z coordinates are linked with their corresponding Bx, By, Bz magnitudes (analogous to a pointer in other programming languages). Consequently, when linear indexing is used to return the position of a coordinate, that same value points to the corresponding magnitude within the [cell] structure.

This allows a single value to point to both the position and magnitude when searching for coordinates to perform the derivatives across, which makes handling the data sets much easier.

[next-next nearest neighbor]

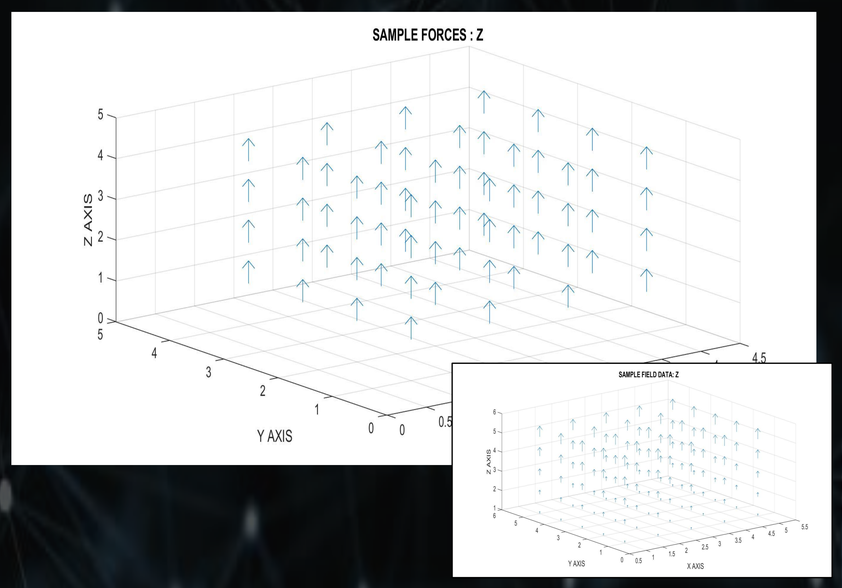
While computing the gradient of the tensor across the 3-dimensional field space, sometimes a 0 value will occur in the derivative, resulting in a undefined result. As a way of preventing this, the algorithm can be implemented such that it only looks at the force contribution from the *8 next-next nearest neighbors* across the **3-axes**: ±x, ±y, and ±z. This is based of the nearest neighbor algorithm, and when extrapolated to 3 space, becomes the "next-next nearest neighbor" method.



The magnetic field varies adiabatically, so this can be done without detriment to the final result. Therefore, the *force* on the point under evaluation becomes the average of the force contribution from the eight surrounding neighbors.

results on a [sample space]

For a magnetic field that increases in strength along the same axes in increments of the same value, we can plot the magnetic field and the resulting equivalent-force vector field:



And for the same field increasing in strength multi-directionally across a 5x5x field space: