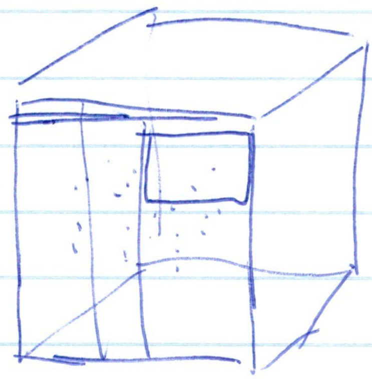
Matrices for coil cubes

$$B_i = \sum_{j=1}^{N_{coils}} M_{ij} I_j \qquad i = 1, \ldots, N_{sensors}.$$

Example 3×3 coils on six faces of cube $\Rightarrow N_{coils} = 54$.

Example 3×3×3 array of 3-axis fluxgates distributed throughout the ROI. $\Rightarrow N_{sensors} = 81$



Matrix notation:

$$B = M I$$

B : column vector of $N_{sensors} = 81$ values (rows)

M : matrix of $N_{sensors} \times N_{coils} = 81 \times 54$ values

I : column vector of $N_{coils} = 54$ values (rows).

Given a certain coil geometry, the magnetic field generated at any fluxgate axis (i) by any given coil (j) is easy to determine from Biot-Savart law (or in this case, the formula of the field generated by a series of straight-line segments). M easy to determine.

Goal: set goal B $=B_{target}$ values and set the coil currents required best to reproduce them.

$$I_{set} = M^{-1} B_{target}.$$

Issue: calculate the inverse of a non-square matrix.

Method: singular value decomposition. (analogous to eigenvalue/eigenvector determination for square matrices)

Write $\quad M = U S V^T$

U: orthogonal matrix $N_{sensors} \times N_{sensors} = 81 \times 81$

V: orthogonal matrix $N_{coils} \times N_{coils} = 54 \times 54$

S: diagonal matrix, same dimension as M.

$N_{sensors} \times N_{coils} = 81 \times 54$

Easy to find the inverse of S, in principle.

Diagonal elements S_{ii} are ordered in size. If $N_{coils} < N_{sensors}$ then there are $N_{coils} = 54$ of them.

Let D be a diagonal matrix of the same dimensions as $S_{(81 \times 54)}$ whose diagonal elements are $D_{ii} = \frac{1}{S_{ii}}$

Then $S^{-1} = D^T$, a matrix of dimension

$N_{coils} \times N_{sensors} = 54 \times 81$. The matrix M^{-1} can

then be calculated from.

$$M^{-1} = V S^{-1} U^T$$

Performing a check of the dimensions:

$V : 54 \times 54$

$S^{-1} : 54 \times 81$

$U^T : 81 \times 81$

and M^{-1} will have the same dimensions as S^{-1}.

For coils with overlapping currents inscribed on a cube, there is always ~~one~~ one mode of applying the same magnitude current to all N_{coils} coils, which gives rise exactly zero net current and hence $B = 0$ in all space. As a result,

$$S_{cc} = 0 \quad \text{where} \quad c = N_{coils} = 54$$

Even in the case of real wires that do not overlap perfectly $S_{cc} << S_{jj}$ for $j < c$.

(Recall, the singular values are ordered in size so that S_{cc} will refer to the smallest one.)

This poses a problem for finding $D_{cc} = \frac{1}{S_{cc}}$.

Even in the case where S_{cc} is not exactly zero, it leads to problems, like:

$$I_{set} = M^{-1} B_{target}$$

if plowing forward and ignoring this problem.

small changes in B_{target} will result in large changes in I_{set}. Ultimately results from having one small singular ~~Previous~~ value, for this system.

See: Gregor Gregorčic example.

Strategies : - regularization (e.g. Tikhonov)

- preventing problematic mode by constraining allowed coil currents (see e.g. S. Ahmed MSc thesis)

- cutting out the problematic mode (truncation) (see e.g. Gregor Gregorčič example, http://github.com/jmartin1454/squares/tet.py and references therein.) We focus on this solution.

Truncation

For coils, S_{cc} can be very small relative to other S_{ii} with $i<c$.

Solution : Let D' be a reduced dimension diagonal matrix, where the last column with the bad singular value has been removed.

$$\text{diagonal elements of } D' = \frac{1}{S_{ii}} \quad \text{up } i=c-1$$

D' will then be dimension $s\times(c-1) = 81\times 53$.

Also cut out the singular vector (corresponding to the bad ~~sv~~ singular value) from V^T — it is in the last row.

$V^{T'}$ is then dimension $(c-1)\times c = 53\times 54$

The modified pseudo-inverse may then be calculated from:

$$M^{-1\,\prime} = (V^{T\prime})^T (D')^T U^T$$

Using our example dimensions:

$(V^{T\prime})^T$ has dimensions 54×53
$(D')^T$ 53×81
U^T 81×81

and so $M^{-1\,\prime}$ has the same dimensions as M^{-1},

$M^{-1\,\prime}$ is 54×81

and

$I_{set} = M^{-1\,\prime} B_{target}$ will be well-behaved under small changes in B_{target}.

Warning:
Codes have built-in "features" to deal with this issue. For example, in python, numpy.linalg.pinv() will automatically truncate singular values/vectors that are zero within the machine precision.