Author Response RC2

Tensorweave 1.0: Interpolating geophysical tensor fields using spatial neural networks

Akshay V. Kamath[1], Samuel T. Thiele[1], Hernan Ugalde[2], Bill Morris[3], Raimon Tolosana-Delgado[1], Moritz Kirsch[1], and Richard Gloaguen[1]

[1]Helmholtz-Zentrum Dresden-Rossendorf, Helmholtz Institute Freiberg for Resource Technology, 09599 Freiberg, Germany.

[2]DIP Geosciences, Hamilton, ON Canada

[3]Morris Magnetics Inc., Fonthill, ON Canada

Correspondence: Akshay Kamath (a.kamath@hzdr.de)

Dear Reviewer,

We thank you for your time and effort reviewing the submitted manuscript, and are pleased that you appreciated our results. We have incorporated your suggestions into the revised manuscript, as detailed in the following pages. Please note that to facilitate the evaluation of our revision, the line numbers of the reviewers' comments refer to the originally submitted manuscript while line numbers of our responses refer to our revised manuscript.

Kindest regards,
Akshay Kamath (on behalf of the authors)

Q1) First: the comparison is done with fairly simple methods (e.g. RBFs), but it is common to use equivalent sources for processing (e.g. this would be standard with a Falcon AGG survey). At a minimum, equivalent sources should be discussed as an approach, but ideally, including a comparison with equivalent-source-based interpolation would be valuable.

We agree that equivalent-source (EQL) methods are widely used in operational gravity and gravity-gradiometry processing, and we have expanded the manuscript to include a focused discussion of EQL approaches and their relevance to full-tensor gravity gradiometry (FTG) [Section 2.2, L76]. We have added the following:

"Another widely used approach for interpolating and transforming potential-field (and gradient) data is the equivalent-source/equivalent-layer method: one replaces the true 3D distribution of sources by a 2D layer of hypothetical monopoles or dipoles beneath the observation surface whose field exactly reproduces the measured data on that surface (Dampney, 1969; Blakely,1995).

In practice the infinite layer is discretised into a finite set of sources and the corresponding dense sensitivity matrix is solved—often with regularisation—to obtain source strengths that honour the physical constraints of potential fields and enable stable continuation and derivative transforms (Hansen and Miyazaki, 1984; Blakely, 1995; Oliveira Junior et al., 2023). This formulation is powerful but computationally demanding for large surveys. Consequently, a substantial literature focuses on accelerating the method by exploiting data geometry and matrix structure: scattered equivalent-source gridding (Cordell, 1992); the "equivalent data" concept to reduce system size (Mendonça and Silva, 1994); wavelet compression and adaptive meshing to sparsify sensitivities (Li and Oldenburg, 2010; Davis and Li, 2011); fast iterative schemes in the space/wavenumber domains (Xia and Sprowl, 1991; Siqueira et al., 2017); and scalable algorithms that leverage the block-Toeplitz Toeplitz-block (BTTB) structure of the sensitivity matrix (Piauilino et al., 2025). Recent machine-learning–inspired variants (e.g., gradient-boosted equivalent sources) further cut memory and runtime for continental-scale datasets (Soler and Uieda, 2021). Open-source implementations, notably Harmonica, provide practical tools for gravity and magnetic datasets using these ideas (Fatiando a Terra Project et al., 2024)."

Regarding a direct comparison in our paper: at present we could not identify a maintained, open-source, out-of-the-box implementation for joint FTG equivalent-source interpolation that we could apply reproducibly to our dataset. The widely used open-source Harmonica library provides EQL and a gradient-boosted variant, but its public API does not support joint inversion of multiple data types (e.g., simultaneously fitting the FTG tensor components), which is required for a fair apples-to-apples FTG comparison. Industrial implementations used in Falcon-style processing are proprietary. Recent academic work on fast EQL for AGG and on convolutional/FFT-based EQL for gravity/magnetics (and a 2025 multi-component FTG variant) does not, to our knowledge, provide open code we can reuse directly. In the interest of reproducibility, we therefore limited our quantitative baselines to well-established, openly available interpolators (RBF, minimum curvature) and provided a detailed discussion of EQL (and its limitations) instead.

Q2) In practice, the full tensor may not be measured in a survey. For example, it is common for gravity gradiometry to only measure two components, e.g. the Falcon system only measures Gne and Guv, and the rest of the tensor is computed from these values. How would you handle this in your method?

We agree that the Falcon AGG system directly measures two horizontal curvature components—G_{ne} and G_{uv} with $G_{uv} = (G_{nn} - G_{ee})/2$—with a system called the Horizontal Partial Tensor Gradiometer (HPTG) and that the remaining tensor components are reconstructed in processing. However our contribution focuses on generic tensor gradiometry, and many sensors (including the Lockheed Martin FTG System flown by Bell Geospace, and the SQUID Magnetic Gradiometers) do measure all five independent components of the tensor to avoid additional assumptions that reduce the degrees of freedom within the system. Both our synthetic and real examples use this kind of dataset.

Q3) As a comment throughout, it would be helpful to have equation numbers to be able to reference.

Equation numbers have been added to all equations throughout the manuscript.

Q4) The equations in sections 3.1 & 3.2 are confusing. The notation is mixed: I understand lowercase bold as vectors, should v_i not be a vector? or is it meant to be an entry of v (in which case the first term W*r is still a vector, so that would need to be indexed? What is the size of r? is it 3N X 1 or N X 3 ?

We agree with the reviewer in that the equations jumping between vector notation and indexed notation were reducing clarity and hindering a clear understanding of the mathematics underlying our approach. Therefore, we have switched all the notations within the manuscript to a fully vector based notation. Starting in Section 3.1, the RFF Mapping equation has been changed to:

$$\boldsymbol{\nu}_s = \left[\sin\left(2\pi\,\mathbf{W}_s\boldsymbol{r}\right),\, \cos\left(2\pi\,\mathbf{W}_s\boldsymbol{r}\right)\right],\ \text{where}\ \mathbf{W}_s = \ell_s^{-1}\mathbf{W}^{M\times 3}$$

For each individual length scale, with the subsequent line (L128) explaining the terms in the equation. The size of **r** is 3×1, and the matrix **W** has a size of *M*×3. This acts on the position vector to return a feature vector of size *M*×1 (2*M*×1 after it passes through both the element-wise sine and cosine, and gets concatenated along the feature axis, which has now been clarified at L128). Section 3.2 has been remedied in a similar way (see Q6).

Q5) Also, it would be helpful to clarify that the number of Fourier features (M) is the same as the number of frequencies.

We agree with the reviewer, and have added the clarification in Section 3.1, L123.

Q6) These questions follow from the point about equations in sections 3.1 and 3.2. I don't understand how the sizes in the equation at line 127 work. Ws is size M x N, but now r is in R2, does r only have 2 entries, or is it N x 2 ? I suspect you are treating it as N x 2. Do you then add these together or take a dot product to collapse it to a vector? is N the total number of points? or is N just 3 because it is a 3D vector. Clarifying this would help with the rest of the math in this section.

The equations shown in the manuscript for the mapping correspond to the mapping acting on a single position column vector, of dimensions *N*×1 where *N* corresponds to the spatial dimension. In section 3.1, since the input is 3D and the mapping is applied to all the dimensions, **r** is 3×1 and the **W** matrix is *M*×3, resulting in a *M*×1 feature vector. In Section 3.2, we split the input position vector into the horizontal position $\boldsymbol{r}_{xy}$ (which is now a 2×1 column vector) and a scalar vertical coordinate (*z*). The **W** matrix now acts only on the horizontal coordinate vector. This has now been clarified in Section 3.2, L151. Furthermore, to avoid any miscommunication about the norm of the **W** matrix and the problems with the application of the decay to the features, we have defined a new vector $\boldsymbol{K}_s$, which is a *M*×1 vector, which is multiplied by the scalar *z* before being exponentiated to acquire a decay vector of size *M*. The new equation for the harmonic embedding is given at L158

$$\boldsymbol{\nu}_s = \left[\sin\left(2\pi\,\mathbf{W}_s\boldsymbol{r}_{xy}\right)\odot e^{-\boldsymbol{\kappa}_s\,z},\ \cos\left(2\pi\,\mathbf{W}_s\boldsymbol{r}_{xy}\right)\odot e^{-\boldsymbol{\kappa}_s\,z}\right]$$

Furthermore, we have also added a paragraph on the logic behind our embedding, from L139 - L150. We hope that these improvements help in clarifying the underlying mathematics. We have also upgraded the notations for all other equations, hopefully increasing readability and clarity.

Q7) Section 3.4 - Network architecture: are you using activation functions between the MLP layers? If so, what are they?

We have added a paragraph about the activation functions used in Section 3.4, L193:

"The MLP block in our model uses non-linear activations for all layers except the output layer. As our framework involves computing second derivatives with AD, using activation functions like ReLU (which do not satisfy the C^2 differentiability criterion) resulted in abrupt edges within the resultant interpolation. Notably, even within the activations that satisfy the aforementioned criterion, some functions performed better than the others. For example, the Hyperbolic Tangent activation function has extremely small second order derivatives which tend to get saturated, impeding convergence. These activations are stable, but not ideal for our models. Among the various activation functions tested, Swish (Sigmoid Linear Unit, SiLU; Ramachandran et al., 2017) and Mish (Misra, 2019) activations provided the best results."

Q6) It would be helpful to state the full training problem in section 3.4 – e.g. what are you minimizing over, presumably the weights in W, and you are summing this over all available data points.

We agree with the reviewer and have upgraded Section 3.4 with the necessary details about our training regimen. Furthermore, we have also explained our hyperparameter tuning choice (starting at L227), along with updates to the Loss equations (Eq 10, 11) to explain the logic we use to train our models. We train the weights within the MLP block of our model, and keep the **W** matrix frozen. This is because **W** acts as a projection matrix for our input vector and encodes the input into the higher dimensional feature space, which then acts as the input for the neural network part of our model. Optimising over the weights matrix **W** would amount to finding the perfect projection that minimises the error between the output field and the measured hessians, which is a much more difficult problem to solve. This has been clarified in the manuscript starting at L238, along with the rest of the parameters used for training, as well as details about the learning rate scheduler, and early stopping criterion.

"We train the MLP parameters (weights and biases) with Adam (Kingma and Ba, 2014), while keeping the random Fourier feature (RFF) encoder fixed after initialisation. The initial learning rate is set to 10^{-3} (occasionally 10^{-2} when the initial loss scale is large). We apply a plateau scheduler that reduces the learning rate by a factor of 0.8 whenever the loss fails to improve for 20 epochs. Optionally, we also optimise a set of learnable length-scale parameters that modulate the Fourier features; the log-values of these scales are stored as parameters and updated jointly with the MLP. However, the learnable nature of these length scales did not help the model convergence greatly, reproducing results explored by Tancik et al. (2020), which suggested that neural fields fail to suitably optimise these length scale parameters."

Q6) In section 4, it would be useful to show the loss curves and provide some discussion of the training process – e.g. how many iterations? What do you use for an optimization algorithm? What was the stopping criterion used? Was it the number of iterations or a threshold value for the loss? What are the final values for each component of the loss (e.g. the data fit vs. Laplacian loss – it would also be nice to see how these evolve as a function of iteration)

We agree with the reviewer in that the loss curves provide a visual understanding of the model's training process. Therefore, we have updated Figures 3 and 7 by adding an additional panel showing all the components of our loss as a function of epochs of training. We trained the models for a specified number of epochs (added in the text at L278 for the synthetic, L322 for the data from Geyer) with an early stopping criterion based on the exponential moving average of the total loss. This has been explained in the manuscript starting at L245. The *Adam* optimisation algorithm was used (and has been specified in the text at L238), and the rest of the information about the loss can be seen in the updated Figures as well.

Other details:

1. **line 25: define Random Fourier Features before the acronym (RFF)** Rectified, we removed the mention of the acronym and now define it directly in Section 3.1.
2. **The definition of equation 110 is a bit of abuse of notation in defining Ws; it would be cleaner to state Ws = 1/l_s W.** We agree and have updated the equations accordingly.
3. **Equation at line 123, if you want to stick with vectors as boldface, I suggest bolding k-hat.**

This equation is a bit odd, because rxy is in R2, and it implies that rz = [0, 0, rz] when you add them together, so the sizes don't match. I get what you mean, but you might be better off stating r = [rxy, rz] or similar. We completely agree and hope that the updates showcased earlier regarding the notation used solve this problem while increasing clarity and readability.