Eigenstrain Calculations Using an Isotropic and Ellipsoidal Eshelby Approximation

Carter Cocke

ME EN 6530

December 12, 2020

Contents

1 Introduction and Background

As a quick introduction, this work was motivated by challenge 4 of the AFRL (Air Force Research Laboratory) Additive Manufacturing Modeling Challenge Series. This challenge gave participants an initial 3D microstructure image of an additively manufactured Inconel 625 specimen, a macroscopic stress-strain response of that same specimen, and finally the grain-averaged elastic strain tensors for 28 target "challenge" grains at the unloaded initial state. The AFRL utilized synchotron x-ray methods such as near-field and far-field high energy x-ray diffraction microscopic (nf and ff-HEDM) to measure the grain-averaged elastic strain tensors at six states during in-situ loading of the IN625 specimen. The challenge participants were then asked to blindly predict the grain-averaged elastic strain tensors for these 28 challenge grains at six macroscopic loading states. Fig. 1 visually shows these loading states. My team (consisting of Drs. Ashley Spear, Anthony Rollett, and Ricardo Lebensohn) received the "Top Performer Award" for best overall blind predictions utilizing the elasto-viscoplastic fast Fourier transform (EVPFFT) crystal plasticity modeling method. As such, we are planning on publishing our work and we decided to complete a "post-mortem" analysis and try to improve our predictions from the challenge. This project aimed to improve the methods used to incorporate the initial elastic strain tensors into our EVPFFT model.

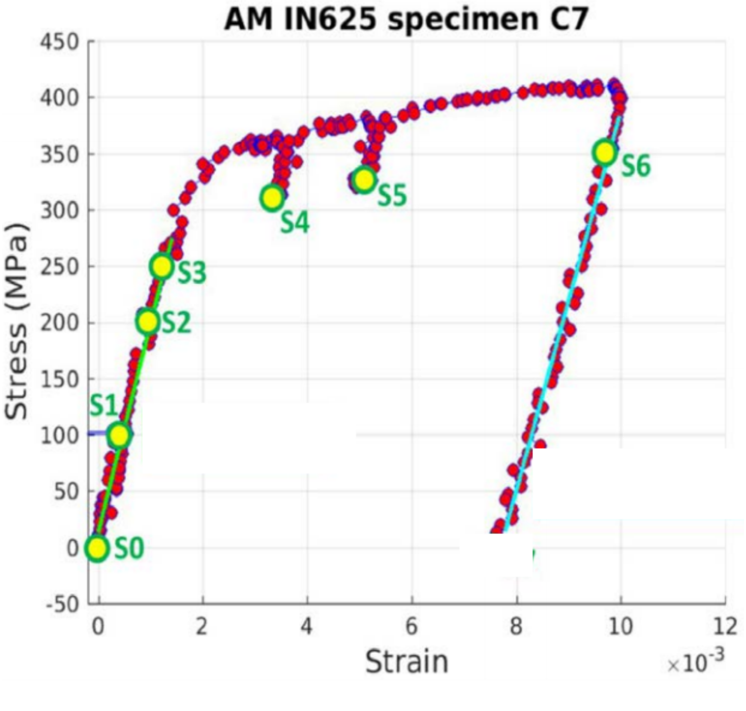

Figure 1: Macroscopic stress-strain response of the IN625 specimen with the initial stress state as well as the six challenge stress states marked [1].

1.1 EVPFFT Method

The EVPFFT modeling method is a crystal plasticity modeling method that utilizes the fast Fourier transform (FFT) algorithm to compute convolution integrals in the context of resolving the micromechanical fields resulting from some applied macroscopic strain. The method itself is not explicitly important to this project, but there is one important constitutive equation that relates stress and strain that forms the basis of the method. Eq. (1) below shows this constitutive relationship written in small strains

$$\varepsilon_{ij}(x) = \varepsilon^{e}_{ij}(x) + \varepsilon^{p}_{ij}(x) = C^{-1}_{ijkl}(x)\sigma_{kl}(x) + \varepsilon^{p,t}_{ij}(x) + \dot{\varepsilon}^{p}_{ij}(x,\sigma)\Delta t + \varepsilon^{*}_{ij}(x) \tag{1}$$

where x represents the localized value of any given parameter, ε_{ij} is the total strain tensor, ε^{e}_{ij} is the elastic strain tensor, ε^{p}_{ij} is the plastic strain tensor, C^{-1}_{ijkl} is the stiffness tensor, σ_{kl} is the stress tensor, $\dot{\varepsilon}^{p}_{ij}$ is the plastic strain rate tensor, and ε^{*}_{ij} is the eigenstrain tensor [2]. There are extremely complicated calculations behind all of these tensors, but the basic EVPFFT scheme revolves around iteratively solving the stress and strain fields such that they satisfy compatibility, static equilibrium, and the constitutive equation given in Eq. (1) [3]. In terms of this project, the most important portion of the EVPFFT to consider is that an initial elastic strain field cannot be directly input as an initial condition to the microstructure. Instead, the initial strains need to be converted to eigenstrains which is why the eigenstrain tensor ε^{*}_{ij} appears in the formulation. These eigenstrain fields are then equilibrated to get the elastic strain fields in sort of roundabout way.

1.2 Eigenstrains

“Eigenstrain” is a term coined by Toshio Mura [4] to represent any nonelastic strains that have no stresses associated with them. These eigenstrains can come from many sources such as thermal expansion, phase transformation, plasticity, etc., however the source of an eigenstrain is irrelevant to determination of an eigenstrain. It is well known that metal-based additive manufacturing causes large thermal gradients that cause residual thermal strains to form inside a material. As the test sample given in the AFRL challenge was produced via additive manufacturing, such residual strains exist and were given by AFRL as part of the provided dataset. These residual or initial strains were provided as elastic strain tensors for the 28 challenge grains. However, as seen previously in Eq. (1), the EVPFFT formulation requires initial eigenstrain tensors and not initial elastic strain tensors. In the example of a permanently deformed grain, the eigenstrain is the strain state the grain would have if the grain were removed from the microstucture and allowed to relax. In this case, there would be no applied stress to the grain but there would be some associated strain state in the grain, i.e. an eigenstrain state. In order to calculate this eigenstrain state, we utilize Eshelby's tensor.

1.3 Eshelby's tensor

In 1957, J.D. Eshelby came up with a theoretical problem wherein an inelastically deformed ellipsoidal inclusion is suspended inside an infinite homogeneous isotropic continuum [5]. Eshelby was able to show that the stress and strain fields inside the inclusion are constant and have a constant tensor S_{ijkl} relating them. This tensor is called Eshelby's tensor and it defines the main tensorial relationship between eigenstrains and elastic strains via Eq. (2)

$$\varepsilon^{\mathrm{e}}_{ij} = S_{ijkl}\varepsilon^{*}_{kl} \tag{2}$$

where $\varepsilon_{ij}^{\mathrm{e}}$ is the elastic strain tensor, S_{ijkl} is the fourth-rank Eshelby tensor, and ε_{kl}^{*} is the eigenstrain tensor. Eq. (2) allows us to calculate the elastic strain tensor from an eigenstrain tensor, but we are trying to calculate the eigenstrain tensor given an elastic strain tensor. Thus, we are solving the inverse problem and we can rewrite Eq. (2) to form Eq. (3)

$$\varepsilon_{ij}^{*} = (S_{ijkl} - I_{ijkl})^{-1}\varepsilon_{kl}^{e} \tag{3}$$

where I_{ijkl} is the fourth-rank identity matrix [2]. The Eshelby tensor is calculated from the shape and elastic properties of the inclusion and has the functional form of Eq. (4)

$$S_{ijkl} = f(a, b, c, \nu) \tag{4}$$

where a, b, and c are the ellipsoid semi-axes and ν is Poisson's ratio. The following sections will cover the calculations of the Eshelby tensor in more detail. In previous work with the EVPFFT modeling method, the eigenstrains calculated through Eq. (3) had some offset issues and it was proposed that a correctional factor β_{ij} should be added to get Eq. (5) [2, 6].

$$\varepsilon_{ij}^{*} = \beta_{ij}(S_{ijkl} - I_{ijkl})^{-1}\varepsilon_{kl}^{e} \quad \text{no sum over } ij \tag{5}$$

This correctional matrix β_{ij} is calculated as follows:

- Calculate the eigenstrain field from the initial elastic strain field of a microstructure volume
- Initialize an EVPFFT simulation with that eigenstrain field and equilibrate the micromechanical fields
- Plot the equilibrated elastic strains $\varepsilon_{ij}^{\mathrm{e,eq}}$ vs. the initial elastic strains $\varepsilon_{ij}^{\mathrm{e,init}}$ on a per grain basis
- Perform linear regression of these plots and calculate the slope m_{ij} and R-squared R_{ij}^2 matrices. These matrices correspond to the regression between the six unique components of $\varepsilon_{ij}^{\mathrm{e,eq}}$ and $\varepsilon_{ij}^{\mathrm{e,init}}$.
- Calculate $\beta_{ij} = \frac{1}{m_{ij}}$ in order to "force" the slope to be 1 (which would imply a 1:1 relationship)

Understanding this correction factor is not very important in the context of this project. However, it is important to note that having larger R_{ij}^2 values is better (implying a better eigenstrain correlation) and having β_{ij} values closer to 1.00 is better (implying less empirical correction is required).

In previous research with the EVPFFT model, the Eshelby tensor has never been calculated on a per grain basis. Instead, there is existing Fortran code that calculates eigenstrains using a global assumption of grain shape (i.e. assuming all grains have the exact same shape). As such, previous work has always assumed spherical grains for the purposes of eigenstrain calculations. Therefore, this project aimed to solve the eigenstrain fields using the shape of each individual grain.

2 Methodology

2.1 Best fit ellipsoids

As the full microstructure image of the challenge microstructure is known, the grain ellipsoidal shapes can be calculated. The dataset provided by AFRL was given as a full 3D voxelized DREAM.3D file. DREAM.3D is an open-source software tool developed to greatly simplify dealing with multidimensional microstructural datasets [7]. The grain ellipsoidal shapes were calculated using the "Find Feature Shapes" filter in DREAM.3D, which calculates grain best fit ellipsoids through calculating the moment of inertia of each grain, calculating the principal moments, and then finally calculating the principal axis lengths of an ellipsoid with those principal moments. DREAM.3D provides the three principal axis lengths (a, b, c) of the best fit ellipsoid and the Euler angles (ϕ, θ, ψ) in $z - x' - z''$ notation of the best fit ellipsoid's orientation for each grain in the microstructure.

2.2 Eshelby tensor calculations

Before jumping into the Eshelby tensor calculations, it is important to note the assumptions made with these calculations. Here are the main assumptions made in the following models:

- Grains have some inherent shape to them (ellipsoid, spherical, cylindrical, etc.).
- An individual grain is surrounded by a homogeneous isotropic matrix with the same material properties as the grain.
- The eigenstrain field outside of a given grain is zero.

2.2.1 Spherical assumption

There are simplified solutions for the Eshelby tensor for special cases of an ellipsoid. For example, a sphere is simply an ellipsoid with equal semi-axis lengths $a = b = c$. With the assumption of perfectly spherical grains, the Eshelby tensor calculation is greatly simplified and the grain shape does not need to be computed. Calculating the best-fit ellipsoids of a grain is not a trivial task, and without a DREAM.3D microstructure representation it is not easy to obtain the ellipsoidal grain shape. Previous work with eigenstrains and the EVPFFT model had only assumed spherical grains because of this. The spherical Eshelby tensor solution is given by Eq. (6)

$$S_{ijkl} = \frac{5\nu - 1}{15(1-\nu)}\delta_{ij}\delta_{kl} + \frac{4 - 5\nu}{15(1-\nu)}\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \tag{6}$$

where δ is the Knonecker Delta and ν is Poisson's ratio. This spherical assumption is actually the limiting case of $a \to 1$, $b \to 1$, and $c \to 1$ for the full ellipsoidal solution provided in the following section.

2.2.2 Ellipsoidal assumption

In order to try and improve the Eshelby approximation, minimizing the assumptions of the model is important. The most straightforward assumption to improve is by incorporating the grain shape into the calculation by using the best fit ellipsoids calculated from DREAM.3D. This means that we will be fully calculating the isotropic Eshelby tensor with no additional simplifications, making the problem relatively complex. The derivation of these equations is given by Mura [4] and is quite lengthy! The following equations are an adaptation between Mura's [4] and Eshelby's [5] final analytical solution of the Eshelby tensor. In the following equations a, b, and c represent the ellipsoidal semi-axes of the grain/inclusion we let $a \to b \to c$ correspond to $1 \to 2 \to 3$. The first required calculations are θ and k which are the amplitude and modulus of the elliptic integrals given later. These two parameters are calculated via Eq. (7) as follows:

$$\theta = \sin^{-1}\left(\sqrt{1-\frac{c^2}{a^2}}\right) \qquad k = \sqrt{\frac{a^2-b^2}{a^2-c^2}} \tag{7}$$

Once θ and k are calculated, two elliptic integrals—the incomplete elliptic integral of the first and second kind—can be calculated via Eqns. (8) and (9).

$$F(\theta,k) = \int_0^\theta \frac{dw}{\sqrt{1-k^2\sin^2(w)}} \tag{8}$$

$$E(\theta,k) = \int_0^\theta \sqrt{1-k^2\sin^2(w)}dw \tag{9}$$

These integrals do not have any simple closed form solution, so these are both numerically integrated using Gauss-Legendre integration. This is part of the reason this is called an Eshelby approximation given we are not calculating a numerically exact solution. It turns out that these integrations are not that computational expensive compared to the rest of the code so 50-point integration was used. Using more than 10-point integration changes the results very negligibly, though. The next part of the code calculates the single-index I-terms in Eqns. (10)-(12)

$$I_1 = \frac{4\pi abc}{(a^2-b^2)\sqrt{a^2-c^2}}\left[F(\theta,k)-E(\theta,k)\right] \tag{10}$$

$$I_3 = \frac{4\pi abc}{(b^2-c^2)\sqrt{a^2-c^2}}\left[\frac{b\sqrt{a^2-c^2}}{ac}-E(\theta,k)\right] \tag{11}$$

$$I_2 = 4\pi - I_1 - I_3 \tag{12}$$

Following the calculation of the single-index I-terms, the double-index I-terms can be calculated. These terms are first calculated on the off-diagonal of the matrix via Eq. (13) where we begin cyclic permutation of the indices (i.e. let $1 \to 2 \to 3$ alongside $a \to b \to c$).

$$I_{12} = \frac{I_2 - I_1}{a^2+b^2} \tag{13}$$

This use of cyclic permutation is unclear for those unfamiliar, but can be better explained by the follow code snippet:

```
for m in range(0, 3):
    for n in range(0, 3):
        if m != n:
            I_array[m, n] = (I_vector[n] - I_vector[m]) /
            (axes_sq[m] - axes_sq[n])
```

where `I_vector` corresponds to the single-index I-terms previously calculated, `I_array` corresponds to the double-index I-terms, and `axes_sq` is a vector of (a^2, b^2, c^2). Next, the diagonal terms of the double-index I-terms are calculated via Eq. (14) where we again cyclically permute the indices and semi-axes together.

$$I_{11} = \frac{1}{3}\left(\frac{4\pi}{a} - I_{12} - I_{13}\right) \tag{14}$$

Once these terms are calculated, the Eshelby tensor can finally be built. Eqns. (15)-(18) are cyclically permuted and any term that does not exist in the permutation is zero (i.e. $S_{1112} = S_{1223} = S_{1232} = 0$).

$$S_{1111} = \frac{3}{8\pi(1-\nu)}a^2 I_{11} + \frac{1-2\nu}{8\pi(1-\nu)}I_1 \tag{15}$$

$$S_{1122} = \frac{1}{8\pi(1-\nu)}b^2 I_{12} + \frac{1-2\nu}{8\pi(1-\nu)}I_1 \tag{16}$$

$$S_{1133} = \frac{1}{8\pi(1-\nu)}c^2 I_{13} + \frac{1-2\nu}{8\pi(1-\nu)}I_1 \tag{17}$$

$$S_{1212} = \frac{a^2+b^2}{16\pi(1-\nu)}I_{12} + \frac{1-2\nu}{16\pi(1-\nu)}(I_1 + I_2) \tag{18}$$

These equations are not very easy to cyclically permute, so refer to the code for more guidance on how these calculations take place.

2.2.3 Validation

The output Eshelby tensor was compared to an Eshelby tensor calculated from a Fortran code written by Dr. Ricardo Lebensohn as a check to make sure the values were correct. After lots of trial and error, especially with the cyclic permutations, the tensors finally matched (with small and negligible numerical differences). With the Eshelby tensor calculations validated, I was then able to move forward with calculating the eigenstrains.

2.3 Change of basis

The best fit ellipsoids given by DREAM.3D do not have semi-axes that align with the global reference frame. Instead, the ellipsoids have some given orientation to them. DREAM.3D provides this orientation as Euler angles in $z - x' - z''$ notation. As such, the basis of the elastic strain and eigenstrain tensors needs to be converted to and frome the ellipsoids local reference frame. Therefore, the orientation matrix Q needs to be calculated from the Euler angles. Euler angles simply define a set of three consecutive rotations around on axis at a time. The first rotation with an angle ϕ occurs around the z axis via Eq. (19).

$$D_{ij} = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

D_{ij} is simply the orientation matrix for this single rotation. The next rotation of angle θ occurs around the new x-axis or the x'-axis and the orientation matrix for this rotation is given by Eq. (20).

$$C_{ij} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{20}$$

The final rotation of ψ occurs around the new z-axis or the z''-axis and is given by Eq. (21).

$$B_{ij} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

As the final orientation is simply a combination of this set of rotations, the full orientation matrix can then be built by simple multiplication of Eqns. (19)-(21) to form

$$Q_{il} = B_{ij}C_{jk}D_{kl} = \begin{bmatrix} \mathrm{c}(\psi)\mathrm{c}(\phi) - \mathrm{c}(\theta)\mathrm{s}(\phi)\mathrm{s}(\psi) & \mathrm{c}(\psi)\mathrm{s}(\phi) + \mathrm{c}(\theta)\mathrm{c}(\phi)\mathrm{s}(\psi) & \mathrm{s}(\psi)\mathrm{s}(\theta) \\ -\mathrm{s}(\psi)\mathrm{c}(\phi) - \mathrm{c}(\theta)\mathrm{s}(\phi)\mathrm{c}(\psi) & -\mathrm{s}(\psi)\mathrm{s}(\phi) + \mathrm{c}(\theta)\mathrm{c}(\phi)\mathrm{c}(\psi) & \mathrm{c}(\psi)\mathrm{s}(\theta) \\ \mathrm{s}(\theta)\mathrm{s}(\phi) & -\mathrm{s}(\theta)\mathrm{c}(\phi) & \mathrm{c}(\theta) \end{bmatrix} \tag{22}$$

where c and s represent shorthand for cosine and sine, respectively. This orientation matrix can then be used to switch between the global and local ellipsoid reference frames.

2.4 Eigenstrain calculations

The following sections will describe the final equations that incorporate the Eshelby tensor S_{ijkl} and the orientation matrix Q_{ij} as well as some of the code used to complete these calculations.

2.4.1 Calculations

The calculations given here assume that the elastic strain tensor $\varepsilon_{ij}^{\mathrm{e}}$, the orientation matrix Q_{ij}, and the Eshelby tensor S_{ijkl} are known or have been previously calculated using the equations given in the previous sections. The first calculation is to change the basis of the elastic strain tensor into the ellipsoid reference frame using Eq. (23)

$$\varepsilon_{il}^{e'} = Q_{ij}\varepsilon_{jk}^{e}Q_{lk} \tag{23}$$

where $\varepsilon_{il}^{e'}$ is the elastic strain tensor in the best fit ellipsoid reference frame and Q_{ij} is the orientation matrix calculated from Eq. (22). The next calculation utilizes S_{ijkl} to calculate the eigenstrain tensor from the elastic strain tensor using Eq. (24)

$$\varepsilon_{ij}^{*'} = (S_{ijkl} - I_{ijkl})^{-1}\varepsilon_{kl}^{e'} \tag{24}$$

where $\varepsilon^{*'}_{ij}$ is the eigenstrain tensor in the ellipsoid reference frame. This calculation is not very straightforward as inverting a rank 4 tensor requires some abstraction. The rank 4 Eshelby tensor and rank 4 identity matrix are mapped into 9x9 matrices, inverted using standard matrix inversion, and then finally mapped back into rank 4 tensors to complete the calculation. The final calculation is rotate the eigenstrain tensor back into the global reference frame to complete the eigenstrain calculations via Eq. (25)

$$\varepsilon^{*}_{il} = Q_{ji}\varepsilon^{*'}_{jk}Q_{kl} \tag{25}$$

where ε^{*}_{il} is the final eigenstrain tensor. These calculations are then repeated for every grain in the microstructure, which is 29662 grains in the case of the AFRL challenge microstructure.

2.4.2 Python functions

In the Python code there are functions that calculate different parts of the eigenstrain approximation. Here is a list of the following functions with a brief description of what the function does:

- `rank_2_to_voigt` - Converts a symmetric rank 2 tensor into Voigt notation (i.e. tensor T_{ij} goes to T_k where $k = 1 \rightarrow 6$).
- `voigt_to_rank_2` - Converts a tensor in Voigt notation to a rank 2 tensor.
- `rank_4_to_matrix` - Maps a rank 4 tensor into a $9x9$ matrix.
- `matrix_to_rank_4` - Maps a $9x9$ matrix into a rank 4 tensor.
- `EshelbyTensor` - Calculates the rank 4 Eshelby tensor using the calculations in Sec. 2.2.
- `CalculateEigenstrains` - Calculates the eigenstrains for a given ellipsoid with a given elastic strain tensor.
- `ReadCsvDREAM3D` - Reads in the ellipsoid semi-axes, Euler angles, and initial elastic strain tensor data from a DREAM.3D feature data .csv file.
- `OrientationMatrix` - Calculates the orientation matrix from three Euler angles using Eq. (22).
- `WriteOutputs` - Writes two different eigenstrain output files for use with the EVPFFT code.
- `EigenstrainsMain` - Main function that calculates eigenstrains for all grains using the previous functions.

There is also a Pytest file that checks `EshelbyTensor` calculations and `OrientationMatrix` for matrix orthogonality.

3 Results

Before diving into the results, I want to quickly note that the tensile sample is deformed under uniaxial tension on the y-axis. Therefore, tensor or matrix components with the subscript "22" are axial components, "11" and "33" are transverse components, and the rest are shear components.

3.1 Equilibration

Now that all of the introduction and methodology is out of the way, lets compare how the eigenstrains calculated with the Python code (ellipsoidal shape assumption) perform versus the eigenstrains calculated with the Fortran code (spherical shape assumption). The method used for this comparison is to initialize an EVPFFT simulation with the calculated eigenstrains and then equilibrate the micromechanical fields. We then compare the EVPFFT equilibrated elastic fields to the initial elastic fields used in the eigenstrain calculation. A better Eshelby approximation technique would yield a higher R^2 value between these two and a slope m closer to 1 (β is simply the inverse of the slope). Table 1 shows the β_{ij} comparisons between the spherical and ellipsoidal eigenstrain calculations both before and after the correctional matrix is applied. The β_{ij} matrix is symmetric due to the symmetry of the strain tensors, so we only write the six unique components of the matrix. Note that after correction, both shape assumptions have β_{ij} values very close to 1, whereas before correction the spherical shape assumption has much larger β_{ij} values compared to the ellipsoidal shape assumption. Table 2 shows the R^2_{ij} values associated with the line of best fit used to calculate β_{ij}. Appendix A has 4 figures that pictorially show the correlation plots of the data contained in Tables 1 and 2. From Table 2, both the uncorrected and corrected ellipsoidal eigenstrains had higher R^2_{ij} values compared to both cases of the spherical eigenstrains.

Table 1: Calculated correctional matrix β_{ij} values for both grain shape assumptions before and after correction. The ideal case is $\beta_{ij} = 1$ meaning no correction is required.

Shape assumption	Correction	β_{11}	β_{22}	β_{33}	β_{23}	β_{13}	β_{12}
Spherical	Uncorrected	1.07	0.78	1.51	0.86	1.23	1.06
	Corrected	1.01	0.99	0.98	0.97	1.00	1.01
Ellipsoidal	Uncorrected	1.03	1.02	1.09	0.93	1.10	1.03
	Corrected	1.00	1.00	1.01	0.98	1.00	1.00

Table 2: Calculated R^2_{ij} values for both grain shape assumptions before and after correction. The ideal case is $R^2 = 1$ meaning the fit is perfectly linear

Shape assumption	Correction	R^2_{11}	R^2_{22}	R^2_{33}	R^2_{23}	R^2_{13}	R^2_{12}
Spherical	Uncorrected	0.87	0.93	0.74	0.86	0.92	0.91
	Corrected	0.87	0.91	0.83	0.92	0.91	0.91
Ellipsoidal	Uncorrected	0.93	0.97	0.92	0.90	0.93	0.91
	Corrected	0.92	0.97	0.92	0.88	0.93	0.90

3.2 Full micromechanical simulations

Although the β_{ij} and R^2_{ij} values can be used to judge the efficacy of the different eigenstrain approximations, perhaps a better performance evaluation is comparing the predictions of full EVPFFT simulations using varying Eshelby approximations. The full EVPFFT simulations carried out in this section model the elastic and plastic response of the specimen and the microstructure was loaded and unloaded to get to each of the six macroscopic stress states (S1-S6) seen in Fig. 1. Since the AFRL challenge has ended, we have access the actual strain measurements of the challenge grains at these states. As such, we are able to calculate the difference between our EVPFFT model predictions and the actual values. These predictions were graded by an L2 error metric, but it is only important to note that lower error is better. Fig. 2 shows the error values for six different models at the six macroscopic loading states.

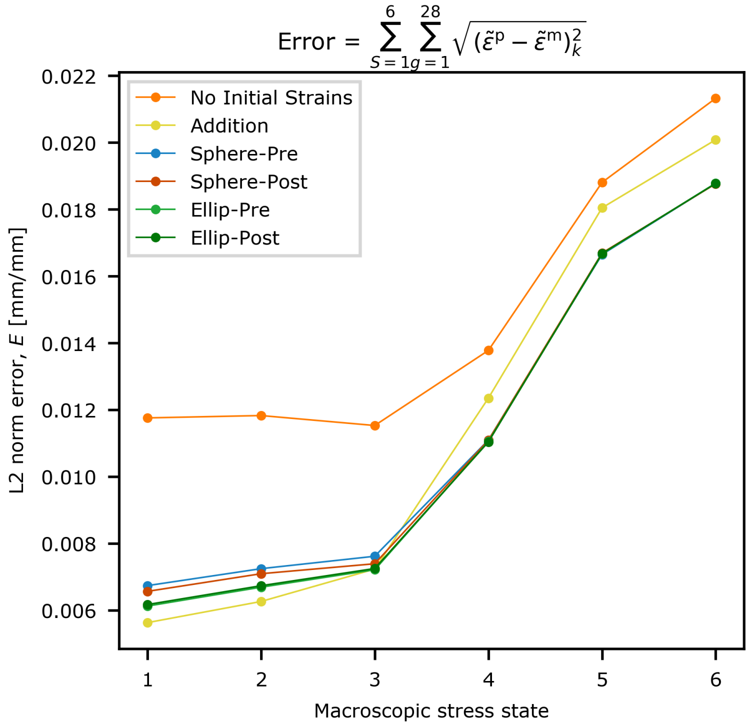

Figure 2: L2 Error between predicted and actual strain values for six different EVPFFT models. Lower error values imply a better model. The "No Initial Strains" model shows a model with no incorporation of the residual strains at state 0 (initial state). The "Addition" model shows a model that simply adds the elastic strains at state 0 to the predictions at all states S1-S6. The "Ellip" and "Sphere" models show the shape assumption (ellipsoidal and spherical) used in the Eshelby tensor calculations and whether or not the model is before (pre) or after (post) the addition of the β_{ij} matrix.

From Fig. 2, the two ellipsoidal shape assumption models performed better than both the spherical shape assumption models. Also, all four of the eigenstrain models converge at S4. This convergence is due to the grains yielding and reaching a maximum elastic strain value that the eigenstrains do not change. It is also interesting to look at the total error (summation of the errors at each stress state) for each model. Fig. 3 shows the relative performance of each model compared to the worst performing model.

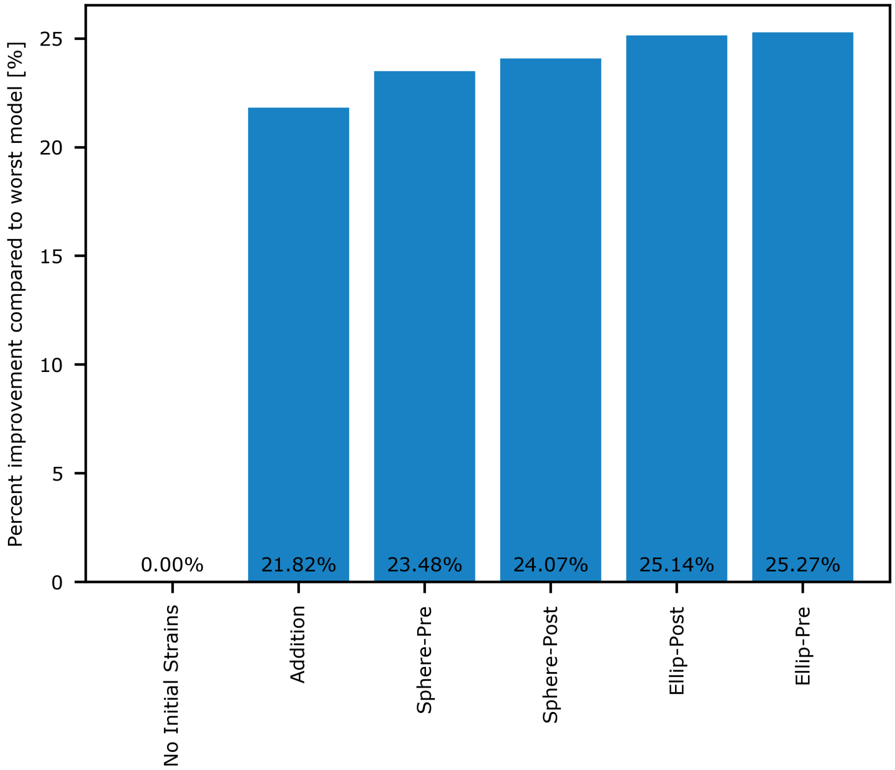

Figure 3: Percent relative performance of each model compared to the model with no incorporation of the initial elastic strains (the worst model). A value of 100% would imply the model perfectly predicts the experimental results. Larger values are better.

Incorporating the initial elastic strains through any method tested improves the results by at least 20%. Incorporation of the ellipsoidal grain shape improves the results by about 1.5% compared to the spherical grain shape assumption.

4 Discussion

Based on the results presented previously, it is clear that the new eigenstrain calculation code improves the predictions of the EVPFFT model. Using the ellipsoidal shape assumption improves on both the spherical shape assumption models, even after the β_{ij} matrix is included. Interesting, the incorporation of the β_{ij} matrix with the ellipsoidal model actually worsens the overall model performance by about 0.1%, although this difference is negligible. An additional benefit of using the ellipsoidal calculations is that the β_{ij} matrix is not necessary as the best-fit slopes are already very close to 1, especially compared to the spherical assumption before correction. The computational time to equilibrate the micromechanical fields to calculate the β_{ij} matrix is about 2 core hours in this case, whereas the ellipsoid eigenstrain calculations takes less than 30 seconds to iterate through all 29662 grains!

Due to the better comparison to experimental results, the better fit and slope of the equilibrated elastic strains, and the reduced computational time, the ellipsoidal eigenstrain calculation only improves on the spherical eigenstrain assumption. The only reason to assume spherical grains is if the grain best fit ellipsoids cannot be easily calculated or if the user simply wants to use a less complex calculation for the eigenstrains that is easier to understand. In all, the ellipsoidal eigenstrain calculation has never been done before with the EVPFFT modeling method and this project shows that this method can work and performs better than any previously used method. This novel approach is shown to have improved the performance of our EVPFFT model and we will be incorporating this calculation into the work we publish in the future!

References

[1] *Air Force Research Laboratory (AFRL) Additive Manufacturing (AM) Modeling Challenge Series.* https://materials-data-facility.github.io/MID3AS-AM-Challenge/. Accessed: 2020-12-07.

[2] Reeju Pokharel and Ricardo A. Lebensohn. "Instantiation of crystal plasticity simulations for micromechanical modelling with direct input from microstructural data collected at light sources". In: *Scripta Materialia* 132 (2017), pp. 73–77. DOI: https://doi.org/10.1016/j.scriptamat.2017.01.025.

[3] Ricardo A. Lebensohn, Anand K. Kanjarla, and Philip Eisenlohr. "An elasto-viscoplastic formulation based on fast Fourier transforms for the prediction of micromechanical fields in polycrystalline materials". In: *International Journal of Plasticity* 32-33 (2012), pp. 59–69. DOI: https://doi.org/10.1016/j.ijplas.2011.12.005.

[4] Toshio Mura. *Micromechanics of Defects in Solids.* 2nd ed. Martinus-Nijhoff, Dordrecht, 1987. ISBN: 9024732565.

[5] J. D. Eshelby. "The Determination of the Elastic Field of an Ellipsoidal Inclusion, and Related Problems". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 241.1226 (1957), pp. 376–396.

[6] Vahid Tari et al. "Validation of micro-mechanical FFT-based simulations using High Energy Diffraction Microscopy on Ti-7Al". In: *Acta Materialia* 154 (2018), pp. 273–283. DOI: https://doi.org/10.1016/j.actamat.2018.05.036.

[7] Michael Groeber and Michael Jackson. "DREAM.3D: A Digital Representation Environment for the Analysis of Microstructure in 3D". In: *Integrating Materials and Manufacturing Innovation* 3 (Feb. 2014), p. 5. DOI: 10.1186/2193-9772-3-5.

Appendix

A Model comparison plots

The following figures show the fit between the equilibrated elastic strains and the initial elastic strains with the ideal fit being the line $y = x$.

A.1 Spherical grain assumption

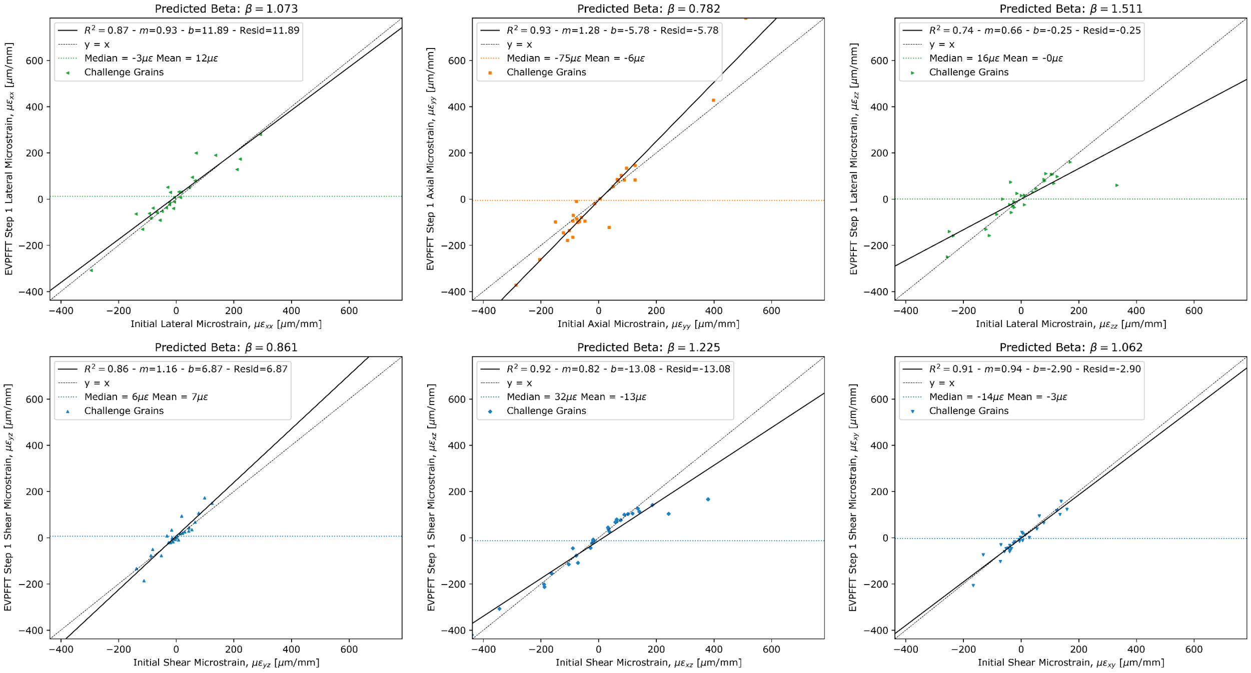

Figure A1: Elastic strain components from the spherical eigenstrain calculations before correction.

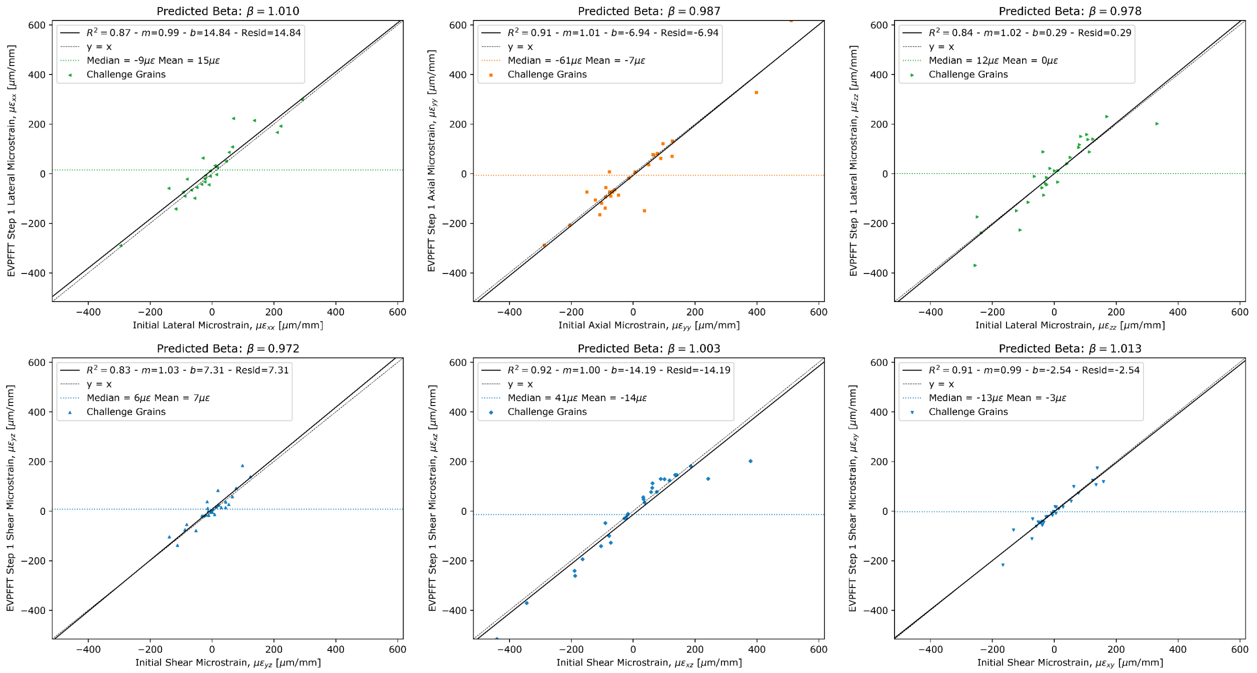

Figure A2: Elastic strain components from the spherical eigenstrain calculations after correction.

A.2 Ellipsoidal grain assumption

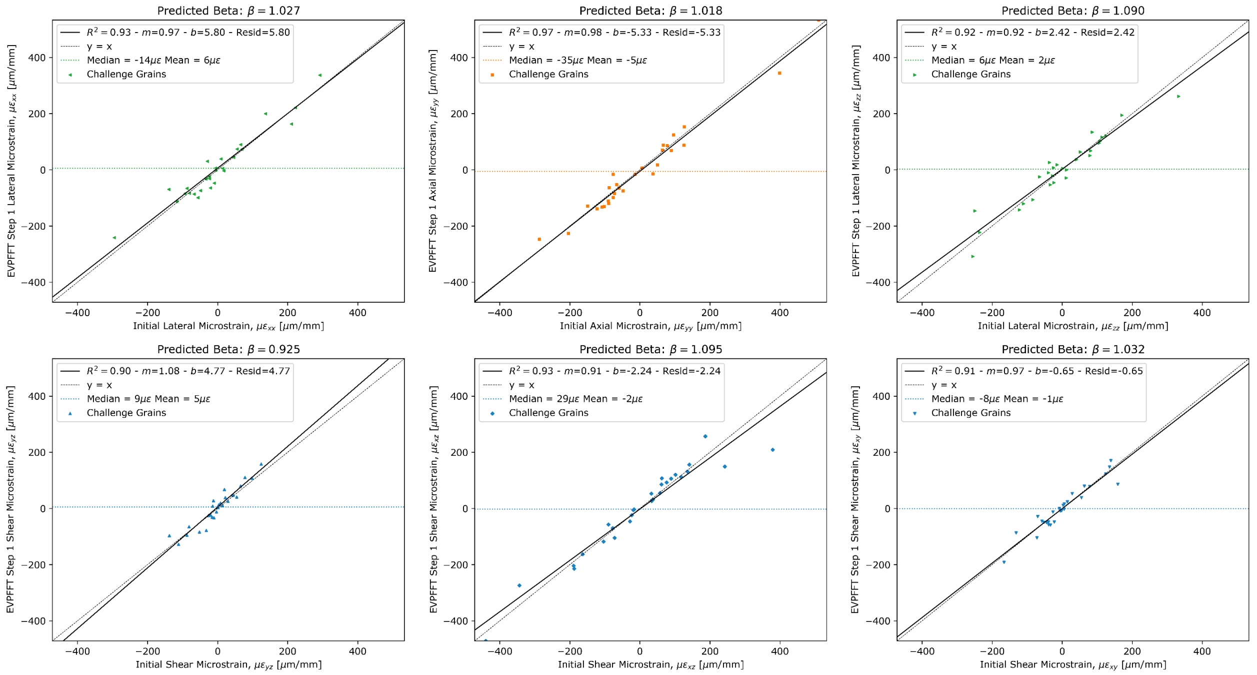

Figure A3: Elastic strain components from the ellipsoidal eigenstrain calculations before correction.

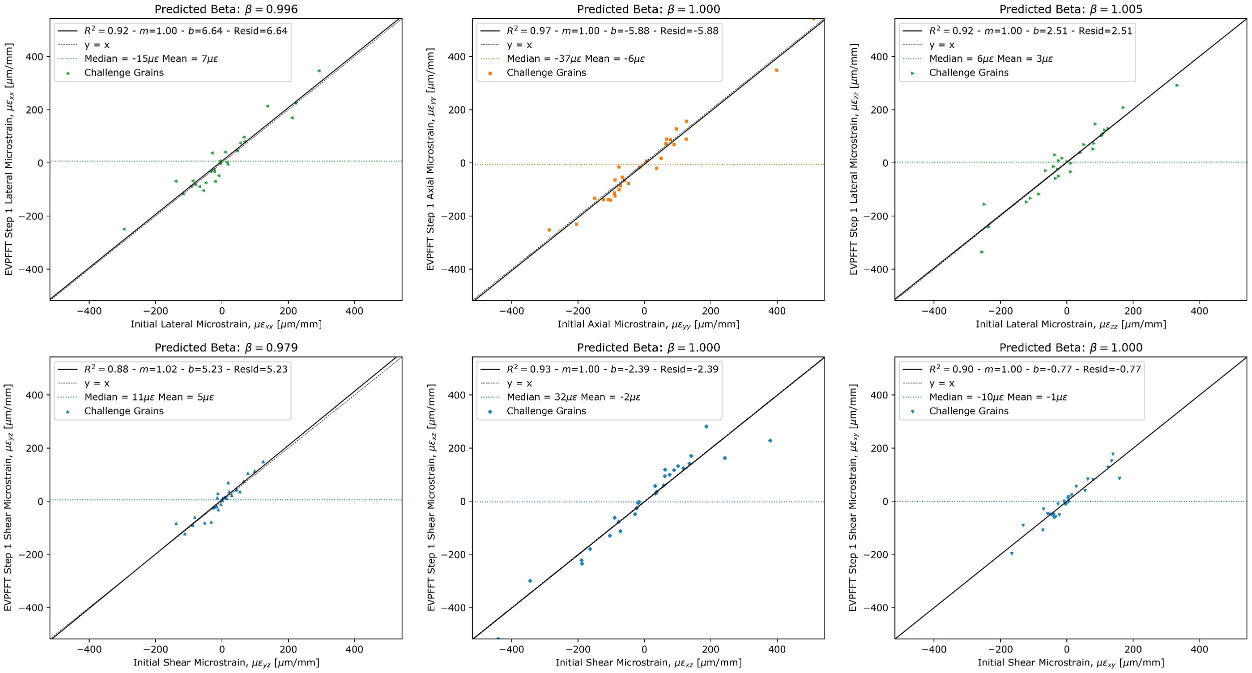

Figure A4: Elastic strain components from the ellipsoidal eigenstrain calculations after correction.