Anyway, on to more exciting news, it looks like my code got final approval for release, and it can be included in OpenXY (this actually happened two months ago, I’ve just been slow).  I also used an MIT license but with the addition of a boilerplate disclaimer.  For the pop-up license info when you first open OpenXY, please add that some of the software is covered under this license:

“Copyright 2021 National Technology & Engineering Solutions of Sandia, LLC (NTESS). Under the terms of Contract DE-NA0003525 with NTESS, the U.S. Government retains certain rights in this software.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.”

I added in an version of GetDeformationGradient that calls all of these different methods.  More options will need to be added to the drop down menu in AdvancedSettings to get access to these methods.  CalcF\_XASGO is the only one that isn’t still under development.  I’ll give a quick rundown of each of these .m files:

CalcF\_XASGO.m – Based on my 2018 paper, this uses advanced image correlation instead of simple cross correlation.  There are a lot of inputs which are just set right now in GetDeformationGradient.m that we may want to add to the GUI someday.  Much of its output metrics are likewise no passed up the chain.  The file is pretty much self-contained.  It makes a lot of messy figures if you run it with the test button at the moment.

CalcF\_KBS.m and CalcF\_KBS.m – Kikuchi bandlet surrogate method.  Bands are isolated in Fourier space (most of this idea borrowed from Farangis Ram), machine learning is used to create a surrogate model of the individual bands, and the pattern center and deformation of the band are iterated until the simulated matches the test pattern.  Bragg’s law is followed, and technically pattern center is easily resolved from strain.  This should be the best way to deal with the reference material from NIST (I’ve isolated much of the error of cross-correlation methods on the reference materials to Bragg’s law).  These are still in the works, as there are plenty of hiccups to work out, so they are not even directly integrated into GetDeformationGradient yet, so you have to use Test\_CalcF\_BandletStrain.m.

CalcF\_Amoeba.m – Nelder-Mead algorithm applied to HREBSD.  The idea of this method is that it is slow, but extremely robust for a bad initial guess.  Uses the same interpolation from XASGO.  I haven’t tested it much, and it is only accessed in commented out code in the FDelta section of GetDeformation gradient.  Amobea\_Opt.m is just my homebrew implementation of Nelder-Mead, which is probably unnecessary because I think it’s built in to matlab.

ResolveFandDelta.m – Much of the action for this method is in GetDeformationGradient.m.  The idea here is that we have 12 unknowns in HREBSD, 9 lattice distortion terms and three PC variables.  We also can scrounge up 12 constraints: 8 from HREBSD, 3 from the traction free condition, and 1 if we assume that the stress is deviatoric.  We use CalcF and then we apply the traction free condition to separate F and DeltaPC.  This is done iteratively with simulated patterns.  Basically this is an extra finicky simulated pattern method that also spits out pattern center at every point.  It’s pretty hiccuppy, but it has a lot of potential, it makes cute fuzzy graphs of the scan area in 3-space if you plot the PCout files, meaning it’s precise enough to give you an approximating of your scan spacing.  I also included my version of HREBSD Main that accepts PCout as an output of GetDeformationGradient, but this is such a minor modification, I’m not slapping a license on it.  The bajillion output method is pretty outdated anyway, we should probably just be outputting a structure so it’s easy to change this sort of thing.

This email is getting pretty long, so I’ll wrap up this one and then continue on another one.  There’s more code, but in the form of scripts that read and process AnalysisParams files (including all my dislocation density stuff) that don’t really need to be integrated into OpenXY, just included in a different folder.  I’ll sort that out and send it tomorrow hopefully.  Please don’t upload anything to github yet, I’d like to get all the files together first.  We should probably set up a phone call sometime, I’m free most of next week every afternoon and all day Friday.

-Tim

Marcus and David,

Nothing in this batch is terribly profound, just plotting and then blmap2 is from my most recent paper, but its only like 35 lines, so how exciting can it be?  I made a general script, APPlotScriptGeneral.m, but I have separate versions of these for lots of scans.  Please let me know if there are any missing functions.  All of this stuff should probably be banished to a subdirectory.

-Tim

Oct 22 21

For reference frames, I think it makes more sense to add them in with the microscope settings.  Advanced settings is mostly to do with Strain/Misorientation and Dislocation Density.  Just four reference frames is a good start (you’ll notice that Setting 3 in EDAX is the same as the default Oxford frame, for example, and Setting 3 in Oxford is EDAX’s default in a hilarious† parallel).  The four EDAX frames are all easy in that they involve rotation about the z axis and can all be fixed by adding or subtracting some multiple of pi/2 to phi1.  In general, however, the conversion can be done as follows:

[p1new,Pnew, p2new] = gmat2euler(euler2gmat(p1,P,p2)\*g\_fix); %I’m not sure if this syntax works for vectors of Euler angles

Where p1, P and p2 are the angles that the software spit out and g\_fix is a matrix that rotates from the OpenXYdefault reference frame to the reference frame setting.  Put even more simply, the above is saying:

G\_OpenXYsamplereferenceframe\_to\_crystal = g\_actualsampleframe\_to\_crystal\*g\_ OpenXYsamplereferenceframe\_to\_actualsampleframe

For example g\_fix for the second setting in Oxford is:

[0    -1    0]

[0     0    1]

[-1    0    0]

G\_fix\*[1;0;0] = [0;0;-1], i.e. what is the x direction in the default frame is actually negative z in the Oxford2 frame.  A simple way to construct g\_fix is to think of the first column as what the x direction in the default frame would be in the new reference frame, the second column is the y direction and the third column is the z direction.

Also, [x,y,z] = [RD,TD,ND] = [A1,A2,A3].  I just realized all the diagrams and my explanation all use different conventions.

Right, clear as mud, I think.  For now, I’d collect data sets using each convention to test.   If your method worked, your test geometry GUI will show everything matching.  Alternatively you could just make g\_fix an input as an option and let the user sort it out.  Might be easier to call me about this if you have more questions.

CalcFDelta is complete, it is CalcF\_KBS that is not.

CalcF\_XASGO should be an option in advanced settings.  It’s basically just a slicker version of remapping, so it should go next to that.  In theory you could run CalcF\_XASGO as an option in dislocation density calcs too (just as you could remapping) but so far my experiments there haven’t been fruitful.

I believe that has answered all your questions, and raised a few more.  Keep asking and I’ll keep answering.

-Tim

†Reference frames have not actually been funny to me since 2014.

**From:** Marcus Behling <[barcusmehling@hotmail.com](mailto:barcusmehling@hotmail.com)>   
**Sent:** Monday, October 25, 2021 4:32 PM  
**To:** Ruggles, Timothy <[truggle@sandia.gov](mailto:truggle@sandia.gov)>; Fullwood, David (External Contact) <[dfullwood@byu.edu](mailto:dfullwood@byu.edu)>  
**Subject:** [EXTERNAL] OpenXY Additions

Hi Tim, I have a few more questions about the OpenXYadditions before I start working on them.

Reference Frames : The plan is to add the ability, in the advanced settings GUI, to select different reference frames.  We'll add at least at 3 for EDAX and the standard oxford reference frame.  Which other Oxford reference frames should be included?  Are there any other reference frames that should be included apart from these?  If so, what is the conversion for these to the standard Oxford reference frame?

CalcF\_Delta : The plan is to add an option to use this in the PC Cal GUI rather than the XASGO method.  Is this function complete yet?  I thought it was incomplete.

CalcF\_XASGO : How would you like this to be implemented?  Dr. Fullwood said that maybe we could have an option to set it as the default in the main settings GUI.

Thanks for all the help!

Marcus