

Supporting information for “Full vector low-temperature magnetic measurements of geologic materials”

September 23, 2014

Contents

1	Introduction	1
2	Import needed libraries into notebook environment	2
2.1	Background subtraction function	2
3	Polycrystalline hematite experiments	2
3.1	IRM-LTI experiment	3
3.1.1	Background subtraction	4
3.1.2	Directional data	6
3.2	MPMS experiment	8
3.3	Comparision between IRM-LTI and MPMS data	8
3.3.1	Estimating the Morin transition	10
4	Single crystal magnetite experiments	13
4.1	IRM-LTI experiments	13
4.1.1	Background subtraction	14
4.1.2	Total vector calculation	16
4.1.3	Directional and intensity plots for all three experiments	16
4.1.4	Additional plots of directional change in <100> experiment	23
5	Polycrystalline magnetite and hematite experiments	26
5.1	IRM-LTI experiment	26
5.1.1	Blank run and background subtraction	27
5.2	Equal area plot	29
5.3	Comparision between IRM-LTI and MPMS data	31
5.3.1	Estimating Morin and Verwey transition temperatures	35
6	Pyrrhotite experiments	36
6.1	IRM-LTI experiments	36
6.1.1	Blank run and background subtraction	37
6.1.2	3-axis and total magnetization plots	38
6.2	Equal area plots	47
7	Umkondo Large Igneous Province sill experiments	51
7.1	Blank probe runs	51
7.1.1	Cold Finger Blank run on July 14, 2013	51
7.1.2	Cold Finger Blank run on July 24, 2013	52
7.2	Experiments on PW10-7d	54
7.2.1	NRM demagnetization	54
7.2.2	MPMS experiments	60

7.2.3	IRM-LTI experiments	62
7.3	PW15-4	71
7.3.1	NRM demagnetization	71
7.4	IRM-LTI experiments	75
7.4.1	LT-cycling of NRM	75
7.4.2	ARM/pARM acquisition and low-temperature cycling	81

1 Introduction

This IPython notebook contains data analysis associated with the paper:

Full vector low-temperature magnetic measurements of geologic materials

Joshua M. Feinberg ¹, Peter A. Solheid ¹, Nicholas L. Swanson-Hysell ^{1,2}, Mike Jackson ¹ and Julie A. Bowles ^{1,3}

¹Institute for Rock Magnetism, University of Minnesota, Minneapolis, MN, USA; ²Department of Earth and Planetary Science, University of California, Berkeley CA, USA; ³Department of Geosciences, University of Wisconsin, Milwaukee, WI, USA

This notebook is part of the paper's supporting online materials and is available at https://github.com/Swanson-Hysell/2014_Feinberg-et-al_LowT along with the necessary files to execute the code. Within this github repository are the following folders:

1. **Code** This folder contains this notebook file as well as necessary non-standard function libraries.
2. **Data** This folder contains the data generated for the case studies presented in this work that are used in the data analysis and presented in the paper.
3. **Manuscript** This folder contains the paper text and figures.

If you are viewing this supporting material as a PDF document and want to run the code in the notebook you will need to download the code from the Github repository and you will also need a Python distribution that includes IPython and the other standard scientific python libraries. There are good instructions for installing IPython/jupyter here: <http://ipython.org/install.html>. Alternatively you can view the notebook online with the jupyter nbviewer at this link: https://nbviewer.jupyter.org/github/swanson-hysell/2014_Feinberg-et-al_low-temp/blob/master/Code/LowT_Data_Analysis.ipynb or this shortened version: <http://bit.ly/1slx6p2>

Within this notebook there are data from runs on a variety of types of specimens:

- **Magnetite**: isothermal remanent magnetizations applied to a single magnetite crystal
- **Hematite**: isothermal remanent magnetization applied to a strongly foliated polycrystalline hematite sample
- **Hematite/Magnetite**: isothermal remanent magnetization applied to a Nature Company polycrystalline sample comprised of hematite and trace magnetite
- **Pyrrhotite**: isothermal remanent magnetization applied to a single crystal of pyrrhotite
- **Coarse-grained diabase**: natural and anhysteretic remanent magnetizations for samples collected from the coarse grained interior of sills from the Umkondo Large Igneous Province of Botswana

2 Import needed libraries into notebook environment

The numpy, scipy, matplotlib and pandas libraries are standard libraries for scientific python (see <http://www.scipy.org>). The pmag and pmagplotlib libraries are part of the PmagPy project and can be obtained from that project's github repository (<https://github.com/ltauxe/PmagPy>). The check_updates and get_version functions of the PmagPy libraries cause errors in the interactive environment of IPython so the code below uses a slightly modified version of those libraries that is included in the Github repository

for this work. There are other functions that are necessary for the interactive plotting of paleomagnetic data (some of them edited from PmagPy) that are within the IPmag.py (see https://github.com/Swanson-Hysell/PmagPy_IPython) library that is imported in the code block below.

```
In [1]: import pmag, pmagplotlib, IPmag
        import numpy as np
        from scipy import interpolate
        import pandas as pd
        from IPython.display import display, Image, display_svg, SVG
        import matplotlib.pyplot as plt
        %matplotlib inline
```

When the notebook is being exported into a PDF it looks much better to have the plots be in svg format rather than png. However, when viewing the notebook as rendered html it is much faster to have the plots be png files so this line will generally be commented out.

```
In [2]: %config InlineBackend.figure_format = 'svg'
```

2.1 Background subtraction function

As discussed in the main text and figures, subtracting the magnetization of the blank probe is a necessary aspect of the data analysis. In order to do so, let's define a function called background_subtraction() that we can use to subtract data of a blank probe run (without a loaded sample) from data experimental runs on specimens. This function uses a spline fit to the background holder data in order to interpolate data to the temperatures where measurements were taking during the sample runs. Examples of fits to background data and subsequent background subtraction are given later in this notebook.

```
In [3]: def background_sub(background_moment, background_temp,
                         measurement_moment, measurement_temp):
    """
    subtract out data from a blank run from measurement data from a sample

    This function takes data measured with the holder alone and subtracts
    it from sample measurement data by interpolating that blank run at the
    temperatures for which data were
    """
    x = background_temp
    y = background_moment
    spline = interpolate.splrep(x,y,s=0)
    xnew = measurement_temp
    background_moment_interpolated = interpolate.splev(xnew,spline,der=0)
    bsub_moment = measurement_moment - background_moment_interpolated
    return bsub_moment
```

3 Polycrystalline hematite experiments

A polycrystalline hematite sample “Labrador hematite” was pulsed with a 1.2 T field to impart an isothermal remanent magnetization (IRM) along the z axis which was then cycled from room temperature to ~130 K and back using the Institute for Rock Magnetism low temperature probe (IRM-LTI) in a superconducting rock magnetometer to collect full vector data. A low-temperature cycling experiment was also conducted using a Magnetic Properties Measurement System (MPMS) to collect 1-axis data.

3.1 IRM-LTI experiment

The data from the IRM-LTI experiment (cooling and warming data separately) can be loaded into a dataframe with the first 5 rows of the dataframe displayed using the `.head()` function. Note that this `.head()` function will be used throughout this notebook in order to give a glimpse of the data without taking the space to print all of the data out. All the data files loaded into the notebook are available in the Github repository.

```
In [4]: Labr_hem_cooling = pd.read_csv('../Data/Labr_hematite/Labr_hem_cooling.csv')
Labr_hem_warming = pd.read_csv('../Data/Labr_hematite/Labr_hem_warming.csv')
Labr_hem_warming.head()
```

```
Out[4]:
```

	specimen	Mx [Am2]	My [Am2]	Mz [Am2]	MN [Am2]	\
0	Labrador hematite 02a	-7.398300e-08	-1.075430e-08	-0	-0	
1	Labrador hematite 02a	-7.496650e-08	-2.935900e-09	-0	-0	
2	Labrador hematite 02a	-7.558250e-08	-1.850900e-09	-0	-0	
3	Labrador hematite 02a	-7.607250e-08	-1.196550e-09	-0	-0	
4	Labrador hematite 02a	-8.034000e-08	-2.277900e-09	-0	-0	

	ME [Am2]	MV [Am2]	Mtot [Am2]	Dec [deg]	Inc [deg]	...	\
0	-1.075430e-08	7.398300e-08	0.000000	181.371	9.34415	...	
1	-2.935900e-09	7.496650e-08	0.000000	180.373	9.44262	...	
2	-1.850900e-09	7.558250e-08	0.000000	180.238	9.62523	...	
3	-1.196550e-09	7.607250e-08	0.000000	180.155	9.77792	...	
4	-2.277900e-09	8.034000e-08	0.000001	180.264	9.23522	...	

	time	series	position[cm]	Drift_ratio	Ja/Jr	run#	Description	\
0	1:25:55 PM	1	NaN	NaN	0	NaN	NaN	
1	1:33:42 PM	1	NaN	NaN	0	NaN	NaN	
2	1:42:26 PM	1	NaN	NaN	0	NaN	NaN	
3	1:48:29 PM	1	NaN	NaN	0	NaN	NaN	
4	1:55:15 PM	1	NaN	NaN	0	NaN	NaN	

	Meas_T[K]	Mode	Type
0	128	DISCRETE	NaN
1	133	DISCRETE	NaN
2	139	DISCRETE	NaN
3	143	DISCRETE	NaN
4	148	DISCRETE	NaN

[5 rows x 35 columns]

During the warming portion of the experiment one of the data points was collected during a flux jump on the SRM that was noted by the analyzer at the time and therefore has ‘flux jump’ in the ‘Description’ column. This point should be removed from the dataframe.

```
In [5]: Labr_hem_warming = Labr_hem_warming[Labr_hem_warming['Description'] != 'flux jump']
Labr_hem_warming.reset_index(inplace=True)
```

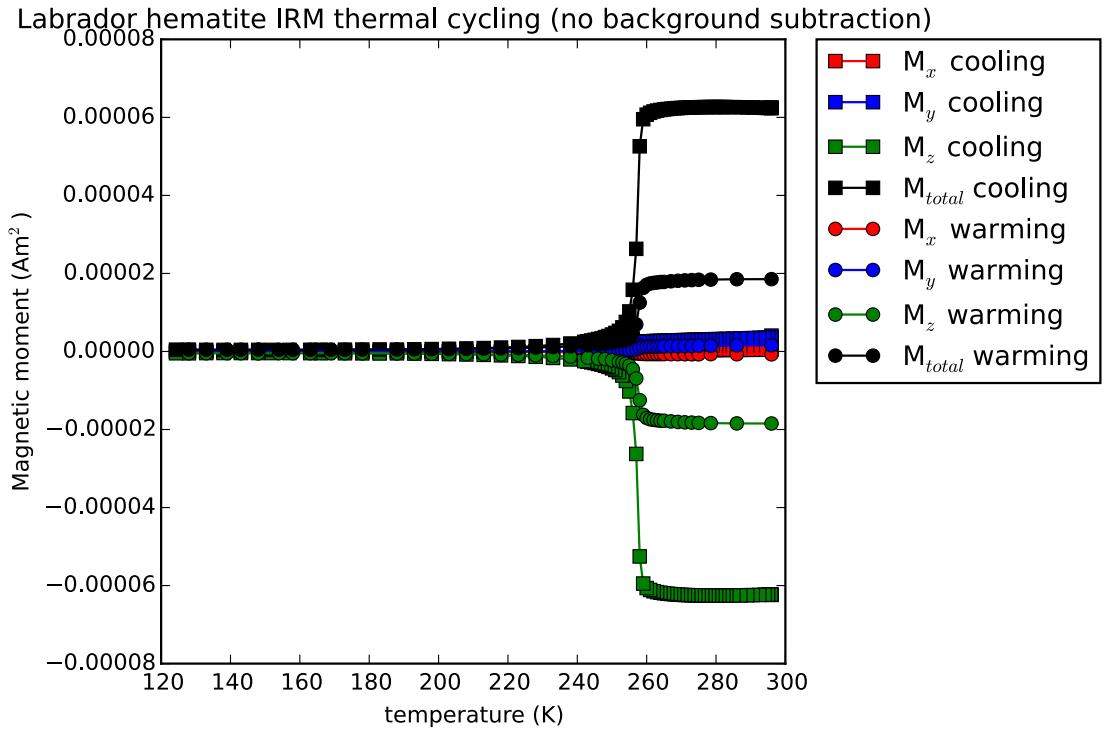
The M_x , M_y , M_z , M_{total} values can be plotted as a function of temperature for this experiment showing the pronounced loss of remanence associated with cooling through hematite’s Morin transition.

```
In [6]: plt.figure(figsize=(5,5))
plt.plot(Labr_hem_cooling['Meas_T[K]'], Labr_hem_cooling['Mx[Am2]'],
          'rs-', label='M$_x$ cooling')
plt.plot(Labr_hem_cooling['Meas_T[K]'], Labr_hem_cooling['My[Am2]'],
```

```

        'bs-' ,label='M$_y$ cooling')
plt.plot(Labr_hem_cooling['Meas_T[K]'], Labr_hem_cooling['Mz[Am2]'],
         'gs-' ,label='M$_z$ cooling')
plt.plot(Labr_hem_cooling['Meas_T[K]'], Labr_hem_cooling['Mtot[Am2]'],
         'ks-' ,label='M$_{total}$ cooling')
plt.plot(Labr_hem_warming['Meas_T[K]'], Labr_hem_warming['Mx[Am2]'],
         'ro-' ,label='M$_x$ warming')
plt.plot(Labr_hem_warming['Meas_T[K]'], Labr_hem_warming['My[Am2]'],
         'bo-' ,label='M$_y$ warming')
plt.plot(Labr_hem_warming['Meas_T[K]'], Labr_hem_warming['Mz[Am2]'],
         'go-' ,label='M$_z$ warming')
plt.plot(Labr_hem_warming['Meas_T[K]'], Labr_hem_warming['Mtot[Am2]'],
         'ko-' ,label='M$_{total}$ warming')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('temperature (K)')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.title('Labrador hematite IRM thermal cycling (no background subtraction)')
plt.show()

```



3.1.1 Background subtraction

These data need to have the background subtracted from them. We will use a run of the blank probe without a loaded sample that was conducted on 04/28/2010. Data from that blank run is imported into dataframes and plotted below.

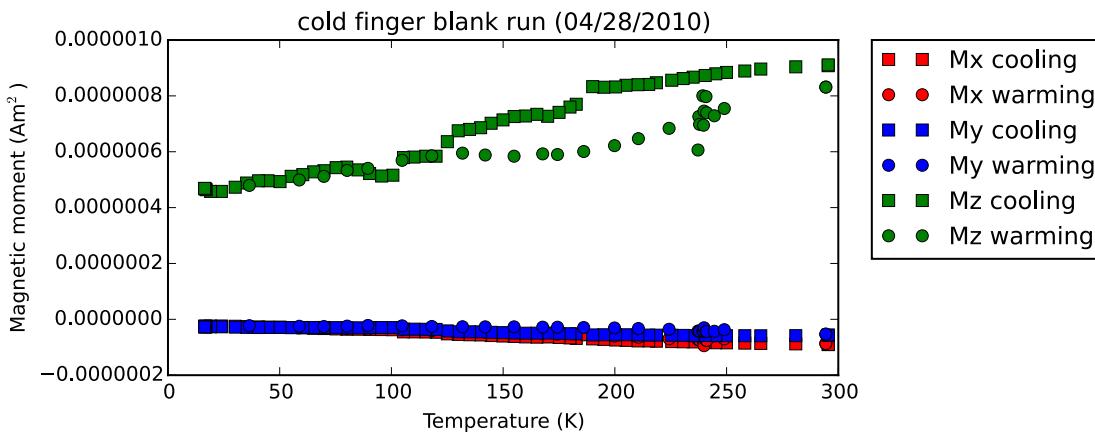
```
In [7]: blank_hem_data_cooling = pd.read_csv('../Data/Labr_hematite/20100428_blank_cooling.csv')
blank_hem_data_warming = pd.read_csv('../Data/Labr_hematite/20100428_blank_warming.csv')
```

```

figure=1
plt.figure(num=figure, figsize=(6,3))
plt.plot(blank_hem_data_cooling['Meas_T[K]'], blank_hem_data_cooling['Mx[Am2]'], 'rs',
         label='Mx cooling')
plt.plot(blank_hem_data_warming['Meas_T[K]'], blank_hem_data_warming['Mx[Am2]'], 'ro',
         label='Mx warming')
plt.plot(blank_hem_data_cooling['Meas_T[K]'], blank_hem_data_cooling['My[Am2]'], 'bs',
         label='My cooling')
plt.plot(blank_hem_data_warming['Meas_T[K]'], blank_hem_data_warming['My[Am2]'], 'bo',
         label='My warming')
plt.plot(blank_hem_data_cooling['Meas_T[K]'], blank_hem_data_cooling['Mz[Am2]'], 'gs',
         label='Mz cooling')
plt.plot(blank_hem_data_warming['Meas_T[K]'], blank_hem_data_warming['Mz[Am2]'], 'go',
         label='Mz warming')

plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('cold finger blank run (04/28/2010)')
plt.show()

```



The background_subtraction() function can be used to subtract the interpolation of data from this cold finger blank run from the measured data during the low-temperature cycling experiment on the hematite. Given that the magnetization of the cold finger looks to be more or less reversible with temperature and that there is better resolution (without a gap between 250 and 300 K) for the cooling curve, the cooling curve will be used for the background subtraction.

```

In [8]: #For the interpolate function within the background_subtraction function to
#work, the x values (temperature) need to be sorted to be ascending.
blank_hem_data_cooling = blank_hem_data_cooling.sort(['Meas_T[K]'])

Mx_cooling_bsub = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                   blank_hem_data_cooling['Meas_T[K]'],
                                   Labr_hem_cooling['Mx[Am2]'],
                                   Labr_hem_cooling['Meas_T[K]'])

My_cooling_bsub = background_sub(blank_hem_data_cooling['My[Am2]'],
                                   blank_hem_data_cooling['Meas_T[K]'],
                                   Labr_hem_cooling['My[Am2]'],
                                   Labr_hem_cooling['Meas_T[K]'])

```

```
My_cooling_bsub = background_sub(blank_hem_data_cooling['My[Am2]'],
                                   blank_hem_data_cooling['Meas_T[K]'],
                                   Labr_hem_cooling['My[Am2]'],
                                   Labr_hem_cooling['Meas_T[K]'])
```

```

blank_hem_data_cooling['Meas_T[K]'],
Labr_hem_cooling['My[Am2]'],
Labr_hem_cooling['Meas_T[K]'])

Mz_cooling_bsub = background_sub(blank_hem_data_cooling['Mz[Am2]'],
blank_hem_data_cooling['Meas_T[K]'],
Labr_hem_cooling['Mz[Am2]'],
Labr_hem_cooling['Meas_T[K]'])

Mx_warming_bsub = background_sub(blank_hem_data_cooling['Mx[Am2]'],
blank_hem_data_cooling['Meas_T[K]'],
Labr_hem_warming['Mx[Am2]'],
Labr_hem_warming['Meas_T[K]'])

My_warming_bsub = background_sub(blank_hem_data_cooling['My[Am2]'],
blank_hem_data_cooling['Meas_T[K]'],
Labr_hem_warming['My[Am2]'],
Labr_hem_warming['Meas_T[K]'])

Mz_warming_bsub = background_sub(blank_hem_data_cooling['Mz[Am2]'],
blank_hem_data_cooling['Meas_T[K]'],
Labr_hem_warming['Mz[Am2]'],
Labr_hem_warming['Meas_T[K]'])

#calculate the full vector after background subtraction
Mtotal_cooling = np.sqrt(Mx_cooling_bsub**2+My_cooling_bsub**2+Mz_cooling_bsub**2)
Mtotal_warming = np.sqrt(Mx_warming_bsub**2+My_warming_bsub**2+Mz_warming_bsub**2)
#the specimen weighs 1.0129 g to get the units into Am2/kg let's divide by 1.0129/1000
Mtotal_cooling_mass_normalized = Mtotal_cooling/(1.0129/1000)
Mtotal_warming_mass_normalized = Mtotal_warming/(1.0129/1000)

```

3.1.2 Directional data

In addition to plotting the magnetization along each axis, the directions of the data can be plotted on an equal area plot. The code below generates a plot that shows the directions from throughout the low-temperature cycling both without background subtraction and with background subtraction.

```

In [9]: directions_cooling_cart=[]
directions_warming_cart=[]

for n in range(len(Labr_hem_cooling)):
    directions_cooling_cart.append([Labr_hem_cooling['Mx[Am2]'][n],
                                    Labr_hem_cooling['My[Am2]'][n],
                                    Labr_hem_cooling['Mz[Am2]'][n]])
for n in range(len(Labr_hem_warming)):
    directions_warming_cart.append([Labr_hem_warming['Mx[Am2]'][n],
                                    Labr_hem_warming['My[Am2]'][n],
                                    Labr_hem_warming['Mz[Am2]'][n]])

directions_cooling_DI=pmag.cart2dir(directions_cooling_cart)
directions_warming_DI=pmag.cart2dir(directions_warming_cart)

directions_bsub_cooling_cart=[]
directions_bsub_warming_cart=[]

```

```

for n in range(len(Labr_hem_cooling)):
    directions_bsub_cooling_cart.append([Mx_cooling_bsub[n],
                                         My_cooling_bsub[n],
                                         Mz_cooling_bsub[n]])
for n in range(len(Labr_hem_warming)):
    directions_bsub_warming_cart.append([Mx_warming_bsub[n],
                                         My_warming_bsub[n],
                                         Mz_warming_bsub[n]])

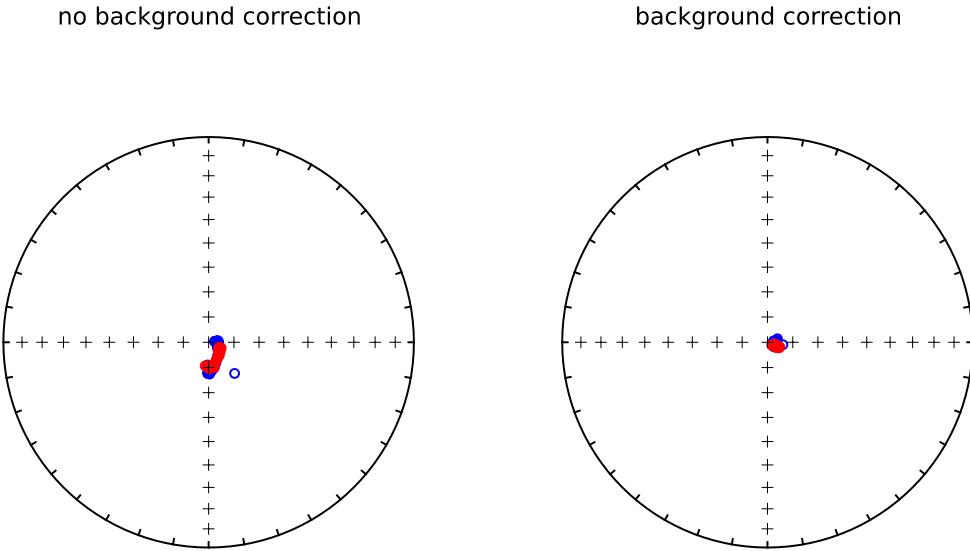
directions_bsub_cooling_DI=pmag.cart2dir(directions_bsub_cooling_cart)
directions_bsub_warming_DI=pmag.cart2dir(directions_bsub_warming_cart)

fig=plt.figure(figsize=(8,5),dpi=160)

plt.subplot(121)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(directions_cooling_DI,'b')
IPmag.iplotDI(directions_warming_DI,'r')
plt.title('no background correction')

plt.subplot(122)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(directions_bsub_cooling_DI,'b')
IPmag.iplotDI(directions_bsub_warming_DI,'r')
plt.title('background correction')
plt.savefig('code_output/Labr_hem_eqarea.pdf')
plt.show()

```



It is clear from the above plot that the background subtraction implementation was effective in removing signal associated with the blank probe. Note that at the time of this experiment the probe was more magnetic

than it is at present as there were subsequent improvements. See the main text for details.

3.2 MPMS experiment

In addition to the 3-axis data obtained with the IRM-LTI, we obtained single axis data on the hematite sample using an MPMS. These data can be imported into a data frame.

```
In [10]: Labr_hem_MPMS = pd.read_csv('../Data/Labr_hematite/Labr_hem_MPMS.csv')
Labr_hem_MPMS.head()

Out[10]:      T [K]  Bapp [T]  M [Am2/kg]  reg fit          timestamp  T [K].1 \
0  299.9905        0   0.041991  0.970553  5/7/2010 12:11:45 PM  19.99694
1  294.3364        0   0.041716  0.970341  5/7/2010 12:15:20 PM  25.49144
2  289.1934        0   0.041456  0.970064  5/7/2010 12:16:10 PM  30.56160
3  284.0423        0   0.041179  0.969775  5/7/2010 12:16:56 PM  35.57994
4  279.7256        0   0.040682  0.969927  5/7/2010 12:19:34 PM  40.53156

      Bapp [T].1  M [Am2/kg].1  reg fit.1  timestamp.1
0            0     0.001129    0.989705  5/7/10 15:18
1            0     0.001129    0.989654  5/7/10 15:20
2            0     0.001129    0.989615  5/7/10 15:21
3            0     0.001128    0.989613  5/7/10 15:22
4            0     0.001128    0.989653  5/7/10 15:23
```

3.3 Comparision between IRM-LTI and MPMS data

As can be seen in the output of the code cell below, the magnetization measured at ~ 289 K (the temperature to which the plots have been normalized) for the specimen was about 33% lower on the specimen on the MPMS than on the IRM-LTI. This could be due, in part, to a portion of the magnetization being off of the one axis being measured on the MPMS although it is unlikely that a misorientation would be as great as the 48° angle that would be required for this to be the sole explanation for the difference.

```
In [11]: Labr_hem_MPMS_289 = Labr_hem_MPMS['M [Am2/kg]'][2]
Labr_hem_LTI_289 = Mtotal_cooling_mass_normalized[7]
print 'At this temperature (K):'
print Labr_hem_MPMS['T [K]'][2]
print 'The magnetization of the hematite sample on the MPMS is (Am2/kg):'
print Labr_hem_MPMS_289
print ''
print 'At this temperature (K):'
print Labr_hem_cooling['Meas_T[K]'][7]
print 'The magnetization of the hematite sample on the IRM-LTI is (Am2/kg):'
print Labr_hem_LTI_289
print ""
print 'The percentage difference between the magnetizations is:'
print ((Labr_hem_LTI_289-Labr_hem_MPMS_289)/Labr_hem_LTI_289)*100
```

At this temperature (K):

289.1934

The magnetization of the hematite sample on the MPMS is (Am2/kg):

0.041456254

At this temperature (K):

289.0

The magnetization of the hematite sample on the IRM-LTI is (Am2/kg):

0.0622871852112

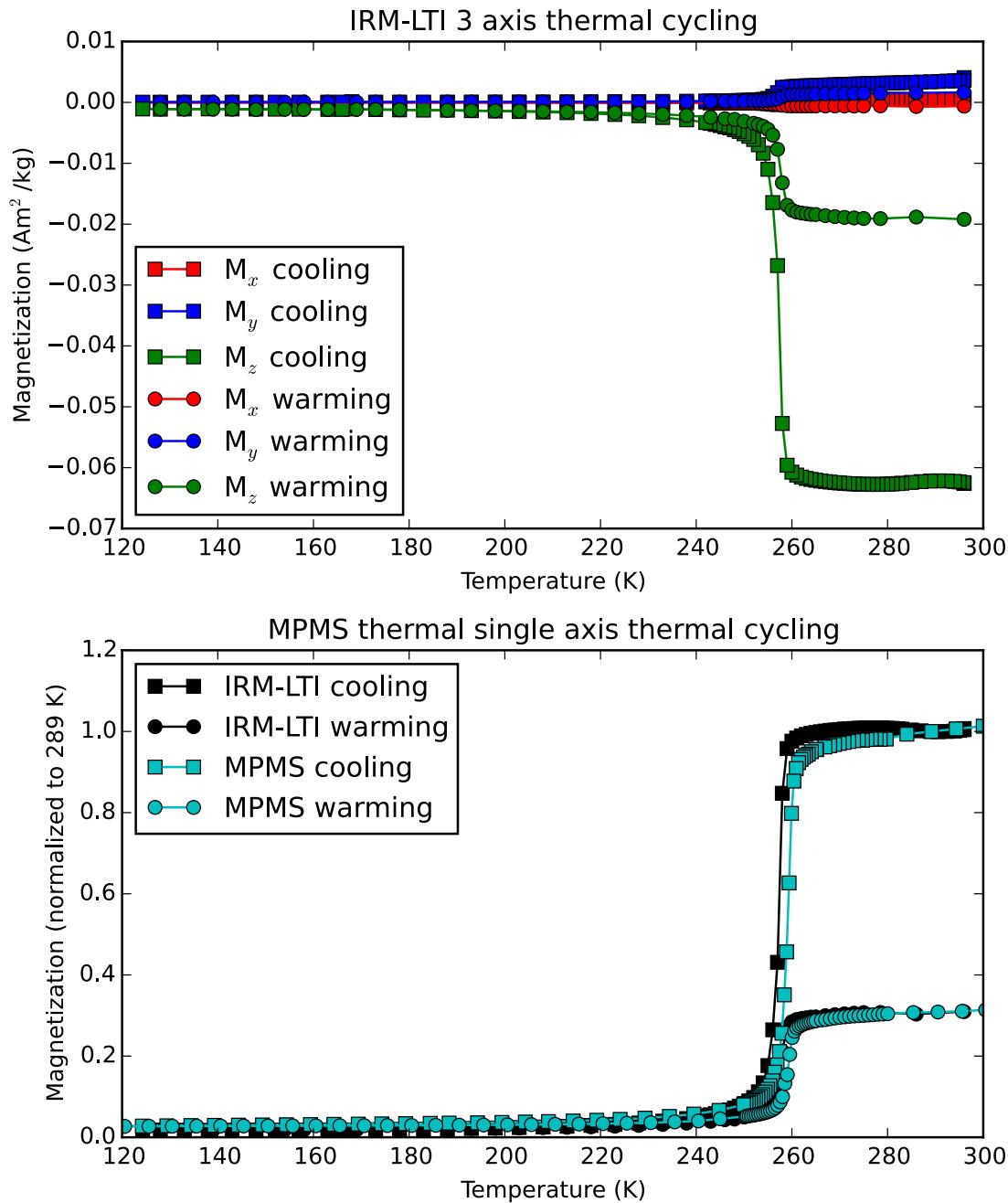
The percentage difference between the magnetizations is:
33.4433658232

Let's make a plot where the MPMS and LTI data are normalized so that they can be directly compared.

```
In [12]: adjustprops = dict(left=0.1, bottom=0.1, right=0.97, top=0.93, wspace=0.25, hspace=0.25)
fig=plt.figure(figsize=(6,8),dpi=160)
fig.subplots_adjust(**adjustprops)
ax1 = plt.subplot(211)

ax1.plot(Labr_hem_cooling['Meas_T[K]'], Mx_cooling_bsub/(1.0129/1000),
          'rs-',label='M$_x$ cooling')
ax1.plot(Labr_hem_cooling['Meas_T[K]'], My_cooling_bsub/(1.0129/1000),
          'bs-',label='M$_y$ cooling')
ax1.plot(Labr_hem_cooling['Meas_T[K]'], Mz_cooling_bsub/(1.0129/1000),
          'gs-',label='M$_z$ cooling')
ax1.plot(Labr_hem_warming['Meas_T[K]'], Mx_warming_bsub/(1.0129/1000),
          'ro-',label='M$_x$ warming')
ax1.plot(Labr_hem_warming['Meas_T[K]'], My_warming_bsub/(1.0129/1000),
          'bo-',label='M$_y$ warming')
ax1.plot(Labr_hem_warming['Meas_T[K]'], Mz_warming_bsub/(1.0129/1000),
          'go-',label='M$_z$ warming')
plt.legend(loc=3)
plt.title('IRM-LTI 3 axis thermal cycling')
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetization (Am$^2$/kg)')

ax2 = plt.subplot(212)
ax2.plot(Labr_hem_cooling['Meas_T[K]'], Mtotal_cooling_mass_normalized/Labr_hem_LTI_289,
          'ks-', label='IRM-LTI cooling')
ax2.plot(Labr_hem_warming['Meas_T[K]'], Mtotal_warming_mass_normalized/Labr_hem_LTI_289,
          'ko-', label='IRM-LTI warming')
ax2.plot(Labr_hem_MPMS['T [K]'], Labr_hem_MPMS['M [Am2/kg]']/Labr_hem_MPMS_289,
          'cs-', label='MPMS cooling')
ax2.plot(Labr_hem_MPMS['T [K].1'], Labr_hem_MPMS['M [Am2/kg].1']/Labr_hem_MPMS_289,
          'co-', label='MPMS warming')
plt.xlim(120, 300)
plt.ylim(ymin=0)
plt.title('MPMS thermal single axis thermal cycling')
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetization (normalized to 289 K)')
plt.legend(loc=2)
plt.savefig('code_output/Labr_hem_data.pdf')
plt.show()
```



3.3.1 Estimating the Morin transition

There looks to be a slight temperature difference between loss of remanence observed using the IRM-LTI and the MPMS. This difference can be seen more clearly in the gradient of the low-temperature cycling data. These gradient data can then be used to make estimates of the Morin transition temperature with each method to quantify the magnitude of this temperature difference.

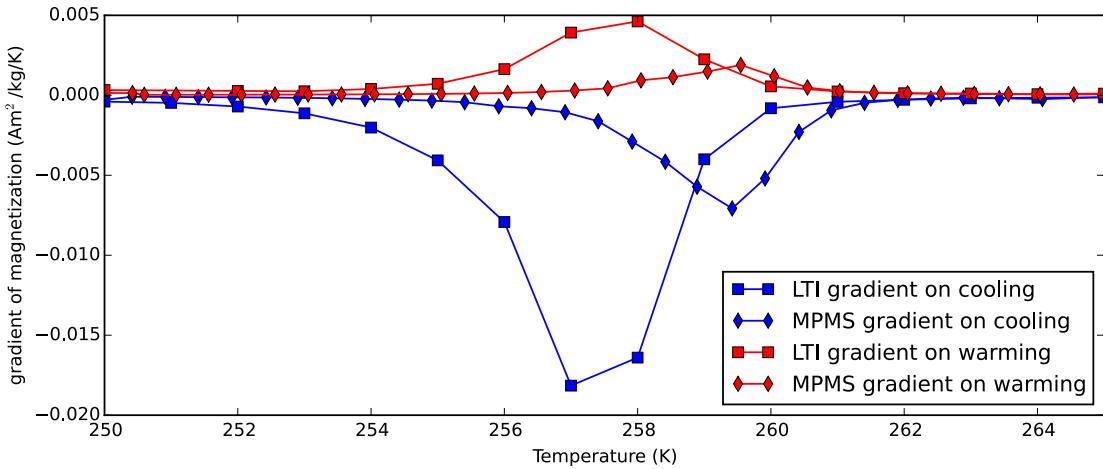
```
In [13]: gradient_LTI_cooling = np.gradient(Mtotal_cooling_mass_normalized)
gradient_MPMS_cooling = np.gradient(Labr_hem_MPMS['M [Am²/kg]'][0:94])
```

```

gradient_LTI_warming = np.gradient(Mtotal_warming_mass_normalized)
gradient_MPMS_warming = np.gradient(Labr_hem_MPMS['M [Am2/kg].1'])

fig=plt.figure(figsize=(10,4))
plt.plot(Labr_hem_cooling['Meas_T[K]'], gradient_LTI_cooling,
          'bs-', label='LTI gradient on cooling')
plt.plot(Labr_hem_MPMS['T [K]'][0:94],gradient_MPMS_cooling,
          'bd-', label='MPMS gradient on cooling')
plt.plot(Labr_hem_warming['Meas_T[K]'], gradient_LTI_warming,
          'rs-', label='LTI gradient on warming')
plt.plot(Labr_hem_MPMS['T [K].1'],gradient_MPMS_warming,
          'rd-', label='MPMS gradient on warming')
plt.xlim(250, 265)
plt.xlabel('Temperature (K)')
plt.ylabel('gradient of magnetization (Am$^2$/kg/K)')
plt.legend(loc=4)
plt.show()

```



To estimate the Morin temperature using these data, let's fit cubic splines to the data. For comparison of the warming and cooling data the negative of the gradient upon warming will be calculated and plotted. The plot compares the data points themselves with the interpolated spline fit.

In [14]: `from scipy.interpolate import interp1d`

```

#interpolate the data using a cubic spline
f_LTI_cooling_inter = interp1d(Labr_hem_cooling['Meas_T[K]'], gradient_LTI_cooling,
                                 kind='cubic')
f_MPMS_cooling_inter = interp1d(Labr_hem_MPMS['T [K]'][0:94], gradient_MPMS_cooling,
                                 kind='cubic')
f_LTI_warming_inter = interp1d(Labr_hem_warming['Meas_T[K]'], -gradient_LTI_warming,
                                 kind='cubic')
f_MPMS_warming_inter = interp1d(Labr_hem_MPMS['T [K].1'], -gradient_MPMS_warming,
                                 kind='cubic')

xnew = np.linspace(250,265,100)

```

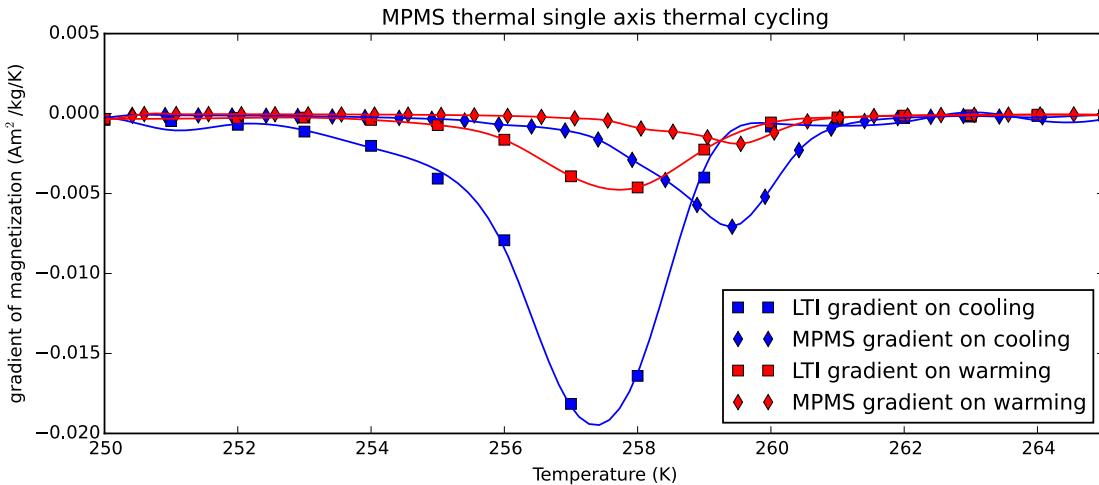
```

gradient_MPMS_cooling_interpolated = f_MPMS_cooling_inter(xnew)
gradient_LTI_cooling_interpolated = f_LTI_cooling_inter(xnew)
gradient_MPMS_warming_interpolated = f_MPMS_warming_inter(xnew)
gradient_LTI_warming_interpolated = f_LTI_warming_inter(xnew)

fig=plt.figure(figsize=(10,4))
plt.plot(Labr_hem_cooling['Meas_T[K]'],gradient_LTI_cooling,
          'bs',label='LTI gradient on cooling')
plt.plot(Labr_hem_MPMS['T [K]'][0:94],gradient_MPMS_cooling,
          'bd', label='MPMS gradient on cooling')
plt.plot(Labr_hem_warming['Meas_T[K]'],-gradient_LTI_warming,
          'rs',label='LTI gradient on warming')
plt.plot(Labr_hem_MPMS['T [K].1'],-gradient_MPMS_warming,
          'rd', label='MPMS gradient on warming')
plt.xlim(250, 265)
plt.title('MPMS thermal single axis thermal cycling')
plt.xlabel('Temperature (K)')
plt.ylabel('gradient of magnetization (Am$^2$/kg/K)')
plt.legend(loc=4)

plt.plot(xnew, gradient_MPMS_cooling_interpolated, 'b-')
plt.plot(xnew, gradient_LTI_cooling_interpolated, 'b-')
plt.plot(xnew, gradient_MPMS_warming_interpolated, 'r-')
plt.plot(xnew, gradient_LTI_warming_interpolated, 'r-')
plt.show()

```



The downhill simplex algorithm as implemented in `scipy.optimize` can be used to find the peak of the interpolated data. This peak is the estimate of the Morin transition temperature.

```

In [15]: from scipy.optimize import fmin
gradient_LTI_cooling_min = fmin(f_LTI_cooling_inter, x0=257)
print 'The peak gradient for LTI cooling is at: ' + str(gradient_LTI_cooling_min)
print ""
gradient_MPMS_cooling_min = fmin(f_MPMS_cooling_inter, x0=259)
print 'The peak gradient for MPMS cooling is at: ' + str(gradient_MPMS_cooling_min)
print ""

```

```

gradient_LTI_warming_min = fmin(f_LTI_warming_inter, x0=257)
print 'The peak gradient for LTI warming is at: ' + str(gradient_LTI_warming_min)
print ""
gradient_MPMS_warming_min = fmin(f_MPMS_warming_inter, x0=259)
print 'The peak gradient for MPMS warming is at: ' + str(gradient_MPMS_warming_min)

Optimization terminated successfully.
    Current function value: -0.019496
    Iterations: 18
    Function evaluations: 36
The peak gradient for LTI cooling is at: [ 257.37332764]

Optimization terminated successfully.
    Current function value: -0.007081
    Iterations: 18
    Function evaluations: 37
The peak gradient for MPMS cooling is at: [ 259.39302902]

Optimization terminated successfully.
    Current function value: -0.004769
    Iterations: 18
    Function evaluations: 36
The peak gradient for LTI warming is at: [ 257.7300869]

Optimization terminated successfully.
    Current function value: -0.001900
    Iterations: 18
    Function evaluations: 37
The peak gradient for MPMS warming is at: [ 259.51168861]

```

Given that the magnitude of the transition is greater upon cooling, we use the Morin transition temperature estimates from the cooling curves in the main text of the paper where we note that there is a 2 degree difference between the Morin transition temperature was estimated using the IRM-LTI (257.4 K) in comparision to the MPMS (259.4 K).

4 Single crystal magnetite experiments

4.1 IRM-LTI experiments

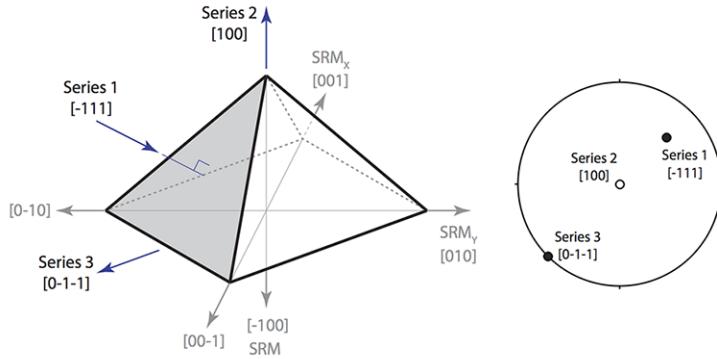
A specimen cut from single crystal magnetite octahedron (IKON magnetite) underwent a series of experiments where IRM magnetizations were applied along different axes and then cycled to low temperature using the IRM-LTI.

- Series 1 experiment was designed to apply an IRM along the < 111 >
- Series 2 experiment was designed to apply an IRM along the < 100 > “SIRM 1.2T (350 Volt pulse) along 100 (along Z)”
- Series 3 experiment was designed to apply the magnetization along the < 110 >

A schematic drawing of the specimen and its orientation relative to the measurement axes of the superconducting rock magnetometer (SRM) and to the orientations of three separate 1.2 T IRMs is shown below.

```
In [16]: Image(filename='..../Data/Ikon_magnetite/experiment_schematic.png'))
```

```
Out[16]:
```



The data from these experiments can be imported into data frames.

```
In [17]: Ikon_series1_cooling = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series1_cooling.csv')
Ikon_series1_warming = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series1_warming.csv')

Ikon_series2_cooling = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series2_cooling.csv')
Ikon_series2_warming = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series2_warming.csv')

Ikon_series3_cooling = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series3_cooling.csv')
Ikon_series3_warming = pd.read_csv('../Data/Ikon_magnetite/Ikon_mag_series3_warming.csv')
```

4.1.1 Background subtraction

The blank holder data used for the background correction of the Labrador hematite sample will also be used to the background correction for these experiments as these data were collected in the same interval of time that the instrument was in use.

```
In [18]: #For the interpolate function to work the x values (temperature) need to be ascending.
#The blank_hem_data_cooling can be sorted so that temperatures are ascending
blank_hem_data_cooling = blank_hem_data_cooling.sort(['Meas_T[K]'])

Ikon_series1_cooling['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_cooling['Mx[Am2]'],
                                                       Ikon_series1_cooling['Meas_T[K]'])

Ikon_series1_cooling['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_cooling['My[Am2]'],
                                                       Ikon_series1_cooling['Meas_T[K]'])

Ikon_series1_cooling['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_cooling['Mz[Am2]'],
                                                       Ikon_series1_cooling['Meas_T[K]'])

Ikon_series1_warming['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_warming['Mx[Am2]'],
                                                       Ikon_series1_warming['Meas_T[K]'])
```

```

Ikon_series1_warming['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_warming['My[Am2]'],
                                                       Ikon_series1_warming['Meas_T[K]'])

Ikon_series1_warming['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series1_warming['Mz[Am2]'],
                                                       Ikon_series1_warming['Meas_T[K]'])

Ikon_series2_cooling['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_cooling['Mx[Am2]'],
                                                       Ikon_series2_cooling['Meas_T[K]'])

Ikon_series2_cooling['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_cooling['My[Am2]'],
                                                       Ikon_series2_cooling['Meas_T[K]'])

Ikon_series2_cooling['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_cooling['Mz[Am2]'],
                                                       Ikon_series2_cooling['Meas_T[K]'])

Ikon_series2_warming['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_warming['Mx[Am2]'],
                                                       Ikon_series2_warming['Meas_T[K]'])

Ikon_series2_warming['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_warming['My[Am2]'],
                                                       Ikon_series2_warming['Meas_T[K]'])

Ikon_series2_warming['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series2_warming['Mz[Am2]'],
                                                       Ikon_series2_warming['Meas_T[K]'])

Ikon_series3_cooling['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_cooling['Mx[Am2]'],
                                                       Ikon_series3_cooling['Meas_T[K]'])

Ikon_series3_cooling['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_cooling['My[Am2]'],
                                                       Ikon_series3_cooling['Meas_T[K]'])

Ikon_series3_cooling['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_cooling['Mz[Am2]'],
                                                       Ikon_series3_cooling['Meas_T[K]'])

```

```

Ikon_series3_cooling['Meas_T[K]'])

Ikon_series3_warming['Mx_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mx[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_warming['Mx[Am2]'],
                                                       Ikon_series3_warming['Meas_T[K]'])

Ikon_series3_warming['My_bsub[Am2]'] = background_sub(blank_hem_data_cooling['My[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_warming['My[Am2]'],
                                                       Ikon_series3_warming['Meas_T[K]'])

Ikon_series3_warming['Mz_bsub[Am2]'] = background_sub(blank_hem_data_cooling['Mz[Am2]'],
                                                       blank_hem_data_cooling['Meas_T[K]'],
                                                       Ikon_series3_warming['Mz[Am2]'],
                                                       Ikon_series3_warming['Meas_T[K]'])

```

4.1.2 Total vector calculation

From these background corrected three-axis data the total vector can be calculated.

```

In [19]: Ikon_series1_cooling['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series1_cooling['Mx_bsub[Am2]**2+
                                                               Ikon_series1_cooling['My_bsub[Am2]**2+
                                                               Ikon_series1_cooling['Mz_bsub[Am2]**2])

Ikon_series1_warming['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series1_warming['Mx_bsub[Am2]**2+
                                                       Ikon_series1_warming['My_bsub[Am2]**2+
                                                       Ikon_series1_warming['Mz_bsub[Am2]**2)

Ikon_series2_cooling['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series2_cooling['Mx_bsub[Am2]**2+
                                                       Ikon_series2_cooling['My_bsub[Am2]**2+
                                                       Ikon_series2_cooling['Mz_bsub[Am2]**2)

Ikon_series2_warming['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series2_warming['Mx_bsub[Am2]**2+
                                                       Ikon_series2_warming['My_bsub[Am2]**2+
                                                       Ikon_series2_warming['Mz_bsub[Am2]**2)

Ikon_series3_cooling['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series3_cooling['Mx_bsub[Am2]**2+
                                                       Ikon_series3_cooling['My_bsub[Am2]**2+
                                                       Ikon_series3_cooling['Mz_bsub[Am2]**2)

Ikon_series3_warming['Mtotal_bsub[Am2]'] = np.sqrt(Ikon_series3_warming['Mx_bsub[Am2]**2+
                                                       Ikon_series3_warming['My_bsub[Am2]**2+
                                                       Ikon_series3_warming['Mz_bsub[Am2]**2)

```

4.1.3 Directional and intensity plots for all three experiments

Plot the three-axis data for each experiment.

```

In [20]: plt.figure(figsize=(3,3),dpi=160)
plt.plot(Ikon_series1_cooling['Meas_T[K]'], Ikon_series1_cooling['Mx_bsub[Am2]', 'rs-',
                           label='Mx cooling')
plt.plot(Ikon_series1_cooling['Meas_T[K]'], Ikon_series1_cooling['My_bsub[Am2]', 'bs-',
                           label='My cooling')
plt.plot(Ikon_series1_cooling['Meas_T[K]'], Ikon_series1_cooling['Mz_bsub[Am2]', 'gs-',
                           label='Mz cooling')
plt.plot(Ikon_series1_warming['Meas_T[K]'], Ikon_series1_warming['Mx_bsub[Am2]', 'ro-',
                           label='Mx warming')

```

```

plt.plot(Ikon_series1_warming['Meas_T[K]'], Ikon_series1_warming['My_bsub[Am2]'], 'bo-',  

         label='My warming')  

plt.plot(Ikon_series1_warming['Meas_T[K]'], Ikon_series1_warming['Mz_bsub[Am2]'], 'go-',  

         label='Mz warming')  

plt.ylabel('Moment (x10$^{-6}$ Am$^2$)')  

plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')  

plt.xlabel('Temperature (K)')  

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)  

plt.title('IKON magnetite (series 1 experiment)')  

plt.show()

plt.figure(figsize=(3,3),dpi=160)  

plt.plot(Ikon_series2_cooling['Meas_T[K]'], Ikon_series2_cooling['Mx_bsub[Am2]'], 'rs-',  

         label='Mx cooling')  

plt.plot(Ikon_series2_cooling['Meas_T[K]'], Ikon_series2_cooling['My_bsub[Am2]'], 'bs-',  

         label='My cooling')  

plt.plot(Ikon_series2_cooling['Meas_T[K]'], Ikon_series2_cooling['Mz_bsub[Am2]'], 'gs-',  

         label='Mz cooling')  

plt.plot(Ikon_series2_warming['Meas_T[K]'], Ikon_series2_warming['Mx_bsub[Am2]'], 'ro-',  

         label='Mx warming')  

plt.plot(Ikon_series2_warming['Meas_T[K]'], Ikon_series2_warming['My_bsub[Am2]'], 'bo-',  

         label='My warming')  

plt.plot(Ikon_series2_warming['Meas_T[K]'], Ikon_series2_warming['Mz_bsub[Am2]'], 'go-',  

         label='Mz warming')  

plt.ylabel('Moment (x10$^{-6}$ Am$^2$)')  

plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')  

plt.xlabel('Temperature (K)')  

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)  

plt.title('IKON magnetite (series 2 experiment)')  

plt.savefig('code_output/Ikon_magnetite_series2.pdf')  

plt.show()

plt.figure(figsize=(3,3),dpi=160)  

plt.plot(Ikon_series3_cooling['Meas_T[K]'], Ikon_series3_cooling['Mx_bsub[Am2]'], 'rs-',  

         label='Mx cooling')  

plt.plot(Ikon_series3_cooling['Meas_T[K]'], Ikon_series3_cooling['My_bsub[Am2]'], 'bs-',  

         label='My cooling')  

plt.plot(Ikon_series3_cooling['Meas_T[K]'], Ikon_series3_cooling['Mz_bsub[Am2]'], 'gs-',  

         label='Mz cooling')  

plt.plot(Ikon_series3_warming['Meas_T[K]'], Ikon_series3_warming['Mx_bsub[Am2]'], 'ro-',  

         label='Mx warming')  

plt.plot(Ikon_series3_warming['Meas_T[K]'], Ikon_series3_warming['My_bsub[Am2]'], 'bo-',  

         label='My warming')  

plt.plot(Ikon_series3_warming['Meas_T[K]'], Ikon_series3_warming['Mz_bsub[Am2]'], 'go-',  

         label='Mz warming')  

plt.ylabel('Moment (x10$^{-6}$ Am$^2$)')  

plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')  

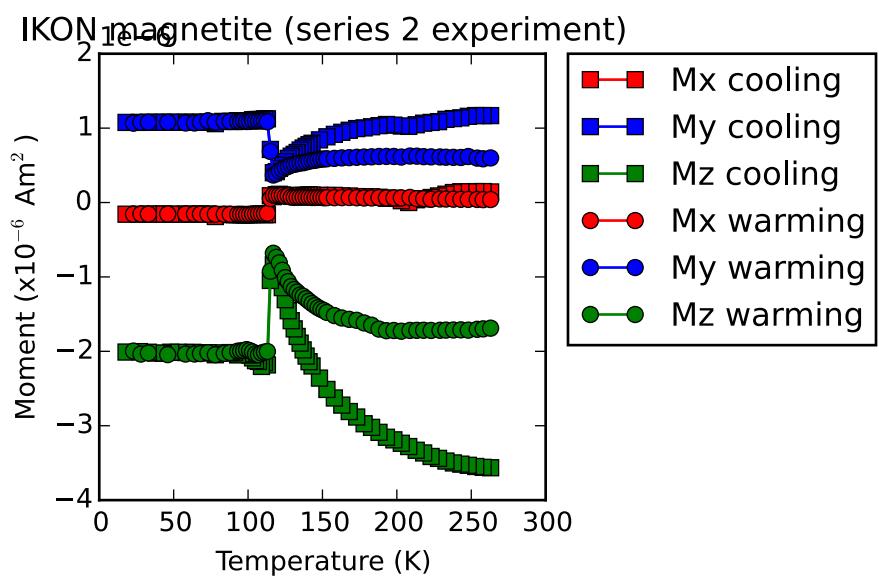
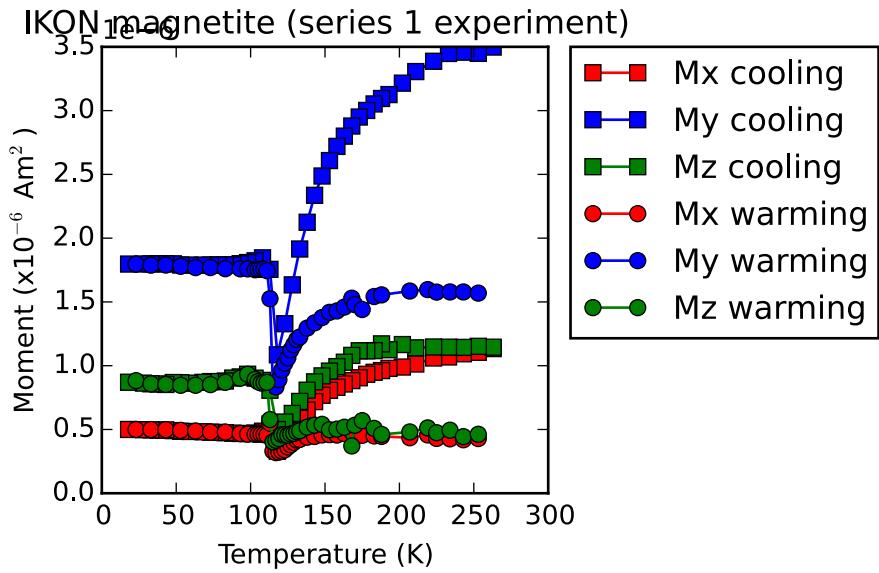
plt.xlabel('Temperature (K)')  

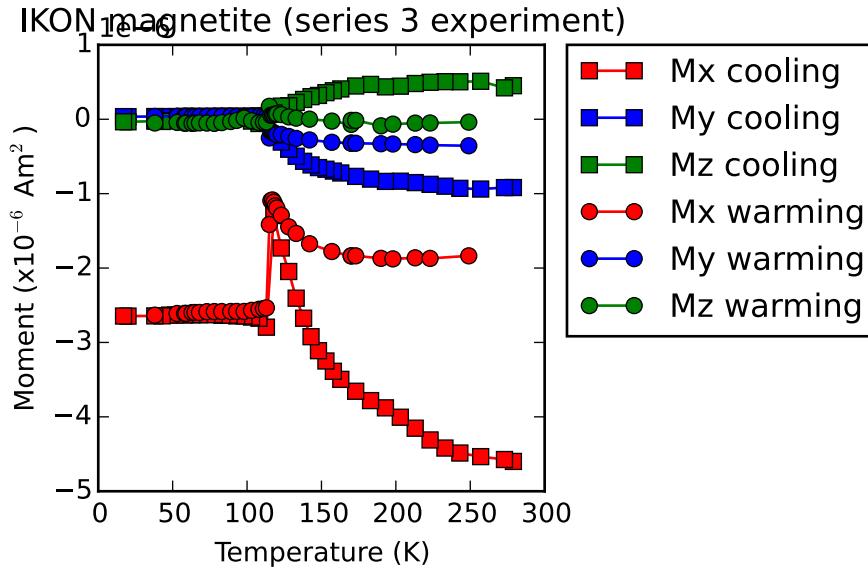
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)  

plt.title('IKON magnetite (series 3 experiment)')  

plt.show()

```

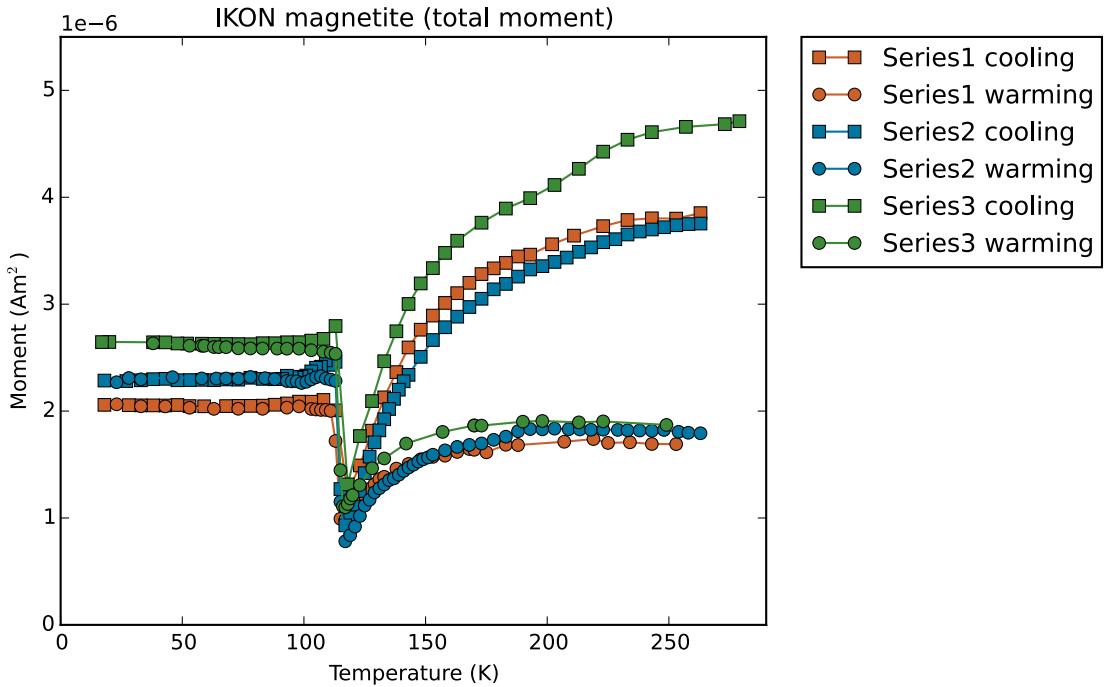




The total moment as calculated from the 3-axis data is plotted below in a single plot that shows all three experiments.

```
In [21]: plt.figure(figsize=(6,5),dpi=160)
plt.plot(Ikon_series1_cooling['Meas_T[K]'], Ikon_series1_cooling['Mtotal_bsub[Am2]'], 's-',
         label='Series1 cooling',color='#CD5F28')
plt.plot(Ikon_series1_warming['Meas_T[K]'], Ikon_series1_warming['Mtotal_bsub[Am2]'], 'o-',
         label='Series1 warming',color='#CD5F28')
plt.plot(Ikon_series2_cooling['Meas_T[K]'], Ikon_series2_cooling['Mtotal_bsub[Am2]'], 's-',
         label='Series2 cooling',color='#0077A2')
plt.plot(Ikon_series2_warming['Meas_T[K]'], Ikon_series2_warming['Mtotal_bsub[Am2]'], 'o-',
         label='Series2 warming',color='#0077A2')
plt.plot(Ikon_series3_cooling['Meas_T[K]'], Ikon_series3_cooling['Mtotal_bsub[Am2]'], 's-',
         label='Series3 cooling',color='#3B8B36')
plt.plot(Ikon_series3_warming['Meas_T[K]'], Ikon_series3_warming['Mtotal_bsub[Am2]'], 'o-',
         label='Series3 warming',color='#3B8B36')

plt.ylabel('Moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.xlim(0, 290)
plt.ylim(0, .0000055)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('IKON magnetite (total moment)')
plt.savefig('code_output/Ikon_magnetite_total.pdf')
plt.show()
```



The remanence lost in each experiment between 260 K and just prior to the Verwey transition increase.

```
In [22]: series1_260 = Ikon_series1_cooling['Mtotal_bsub[Am2]'][0]
          series2_260 = Ikon_series2_cooling['Mtotal_bsub[Am2]'][0]
          series3_260 = Ikon_series3_cooling['Mtotal_bsub[Am2]'][2]
          series1_min = Ikon_series1_cooling['Mtotal_bsub[Am2]'].min()
          series2_min = Ikon_series2_cooling['Mtotal_bsub[Am2]'].min()
          series3_min = Ikon_series3_cooling['Mtotal_bsub[Am2]'].min()
          print 'Series 1 percent lost between 260K and minima'
          print ((series1_260-series1_min)/series1_260)*100
          print 'Series 2 percent lost between 260K and minima'
          print ((series2_260-series2_min)/series2_260)*100
          print 'Series 3 percent lost between 260K and minima'
          print ((series3_260-series3_min)/series3_260)*100

Series 1 percent lost between 260K and minima
67.8202160088
Series 2 percent lost between 260K and minima
75.1987424572
Series 3 percent lost between 260K and minima
71.7873125177
```

The remanence lost in each experiment between 260 K and the plateau following the Verwey transition increase.

```
In [23]: series1_260 = Ikon_series1_cooling['Mtotal_bsub[Am2]'][0]
          series2_260 = Ikon_series2_cooling['Mtotal_bsub[Am2]'][0]
          series3_260 = Ikon_series3_cooling['Mtotal_bsub[Am2]'][2]
          series1_50 = Ikon_series1_cooling['Mtotal_bsub[Am2]'][35]
          series2_50 = Ikon_series2_cooling['Mtotal_bsub[Am2]'][58]
```

```

series3_50 = Ikon_series3_cooling['Mtotal_bsub[Am2]'][33]
print 'Series 1 post-Verwey plateau remanence as percent of 260K remanence'
print (series1_50/series1_260)*100
print 'Series 2 post-Verwey plateau remanence as percent of 260K remanence'
print (series2_50/series2_260)*100
print 'Series 3 post-Verwey plateau remanence as percent of 260K remanence'
print (series3_50/series3_260)*100

Series 1 post-Verwey plateau remanence as percent of 260K remanence
53.4020260118
Series 2 post-Verwey plateau remanence as percent of 260K remanence
60.9993296955
Series 3 post-Verwey plateau remanence as percent of 260K remanence
56.4923557659

```

The remanence lost in each experiment following thermal cycling.

```

In [24]: series1_260_warming = Ikon_series1_warming['Mtotal_bsub[Am2]'][41]
          series2_260_warming = Ikon_series2_warming['Mtotal_bsub[Am2]'][63]
          series3_260_warming = Ikon_series3_warming['Mtotal_bsub[Am2]'][35]
          print 'Series 1 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence'
          print (series1_260_warming/series1_260)*100
          print 'Series 2 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence'
          print (series2_260_warming/series2_260)*100
          print 'Series 3 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence'
          print (series3_260_warming/series3_260)*100

Series 1 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence
43.8987720207
Series 2 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence
47.7466227538
Series 3 post-thermal cycling remanence (at 250-260K) as percent of 260K remanence
40.1523508649

```

The directional data for the three experiments is plotted on a single equal area plot.

```

In [25]: directions_cart_series1_cooling=[]
          directions_cart_series1_warming=[]

          for n in range(len(Ikon_series1_cooling)):
              directions_cart_series1_cooling.append([Ikon_series1_cooling['Mx_bsub[Am2]'][n],
                                                       Ikon_series1_cooling['My_bsub[Am2]'][n],
                                                       Ikon_series1_cooling['Mz_bsub[Am2]'][n]])
          for n in range(len(Ikon_series1_warming)):
              directions_cart_series1_warming.append([Ikon_series1_warming['Mx_bsub[Am2]'][n],
                                                       Ikon_series1_warming['My_bsub[Am2]'][n],
                                                       Ikon_series1_warming['Mz_bsub[Am2]'][n]])

          directions_DI_series1_cooling=pmag.cart2dir(directions_cart_series1_cooling)
          directions_DI_series1_warming=pmag.cart2dir(directions_cart_series1_warming)

          directions_cart_series2_cooling=[]
          directions_cart_series2_warming=[]

          for n in range(len(Ikon_series2_cooling)):

```

```

directions_cart_series2_cooling.append([Ikon_series2_cooling['Mx_bsub[Am2]'][n],
                                       Ikon_series2_cooling['My_bsub[Am2]'][n],
                                       Ikon_series2_cooling['Mz_bsub[Am2]'][n]])
for n in range(len(Ikon_series2_warming)):
    directions_cart_series2_warming.append([Ikon_series2_warming['Mx_bsub[Am2]'][n],
                                             Ikon_series2_warming['My_bsub[Am2]'][n],
                                             Ikon_series2_warming['Mz_bsub[Am2]'][n]])

directions_DI_series2_cooling=pmag.cart2dir(directions_cart_series2_cooling)
directions_DI_series2_warming=pmag.cart2dir(directions_cart_series2_warming)

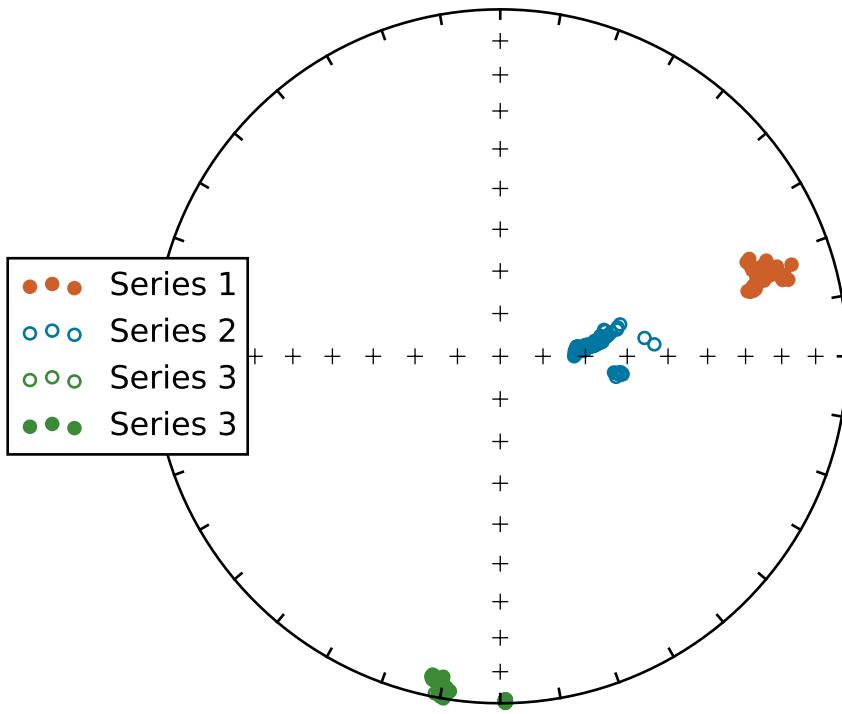
directions_cart_series3_cooling=[]
directions_cart_series3_warming=[]

for n in range(len(Ikon_series3_cooling)):
    directions_cart_series3_cooling.append([Ikon_series3_cooling['Mx_bsub[Am2]'][n],
                                             Ikon_series3_cooling['My_bsub[Am2]'][n],
                                             Ikon_series3_cooling['Mz_bsub[Am2]'][n]])
for n in range(len(Ikon_series3_warming)):
    directions_cart_series3_warming.append([Ikon_series3_warming['Mx_bsub[Am2]'][n],
                                             Ikon_series3_warming['My_bsub[Am2]'][n],
                                             Ikon_series3_warming['Mz_bsub[Am2]'][n]])

directions_DI_series3_cooling=pmag.cart2dir(directions_cart_series3_cooling)
directions_DI_series3_warming=pmag.cart2dir(directions_cart_series3_warming)

fignum = 1
plt.figure(num=fignum, figsize=(6, 6), dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(directions_DI_series1_cooling, color='#CD5F28', label='Series 1')
IPmag.iplotDI(directions_DI_series1_warming, color='#CD5F28')
IPmag.iplotDI(directions_DI_series2_cooling, color='#0077A2', label='Series 2')
IPmag.iplotDI(directions_DI_series2_warming, color='#0077A2')
IPmag.iplotDI(directions_DI_series3_cooling, color='#3B8B36', label='Series 3')
IPmag.iplotDI(directions_DI_series3_warming, color='#3B8B36')
plt.legend(loc=6)
plt.savefig('code_output/Ikon_magnetite_EA.pdf')
plt.show()

```



4.1.4 Additional plots of directional change in <100> experiment

There are some quite interesting directional changes in the Series 2 data set. The data from that experiment is plotted below on Zijderveld plots.

```
In [26]: #make a datablock with this format: [treatment step, dec, inc, intensity]
datablock_cooling=[]
for n in range(0,len(Ikon_series2_cooling)):
    datablock_cooling.append([Ikon_series2_cooling['Meas_T[K]'][n],
                           directions_DI_series2_cooling[n][0],
                           directions_DI_series2_cooling[n][1],
                           directions_DI_series2_cooling[n][2]])
datablock_warming=[]
for n in range(0,len(Ikon_series2_warming)):
    datablock_warming.append([Ikon_series2_warming['Meas_T[K]'][n],
```

```

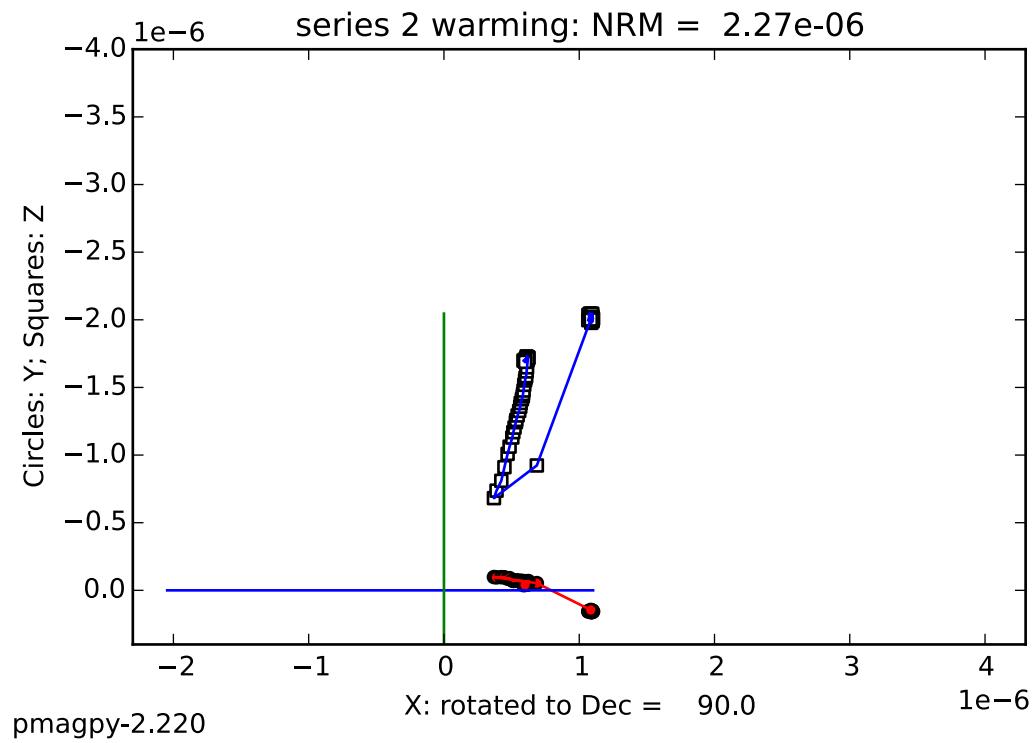
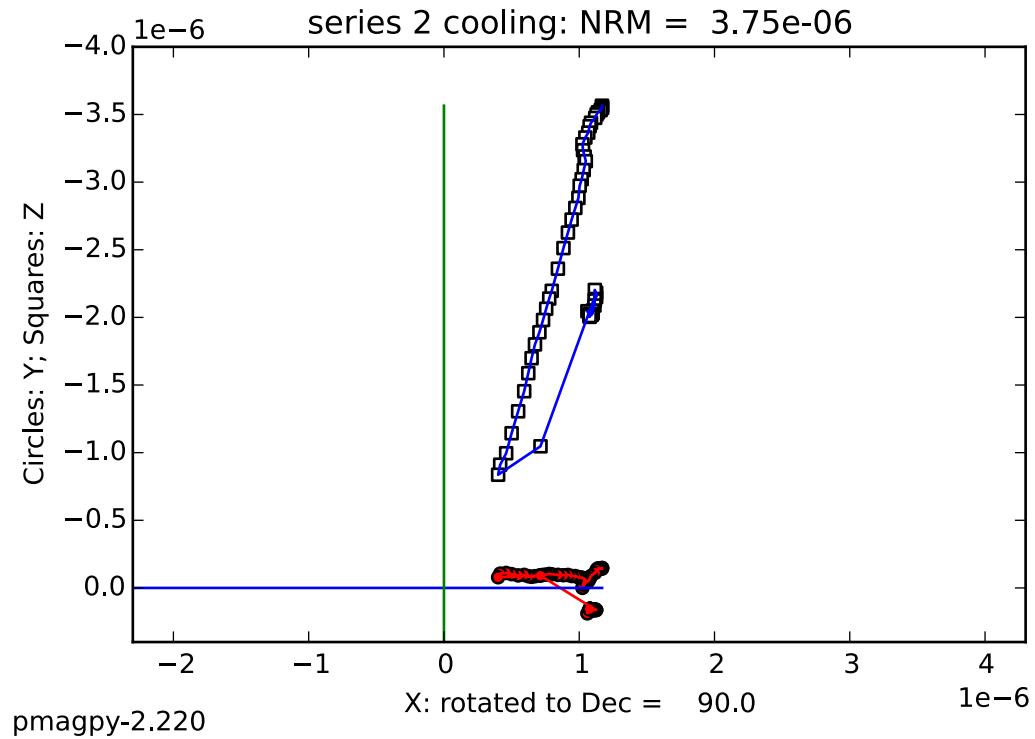
        directions_DI_series2_warming[n][0],
        directions_DI_series2_warming[n][1],
        directions_DI_series2_warming[n][2]))
datablock_all=datablock_cooling+datablock_warming

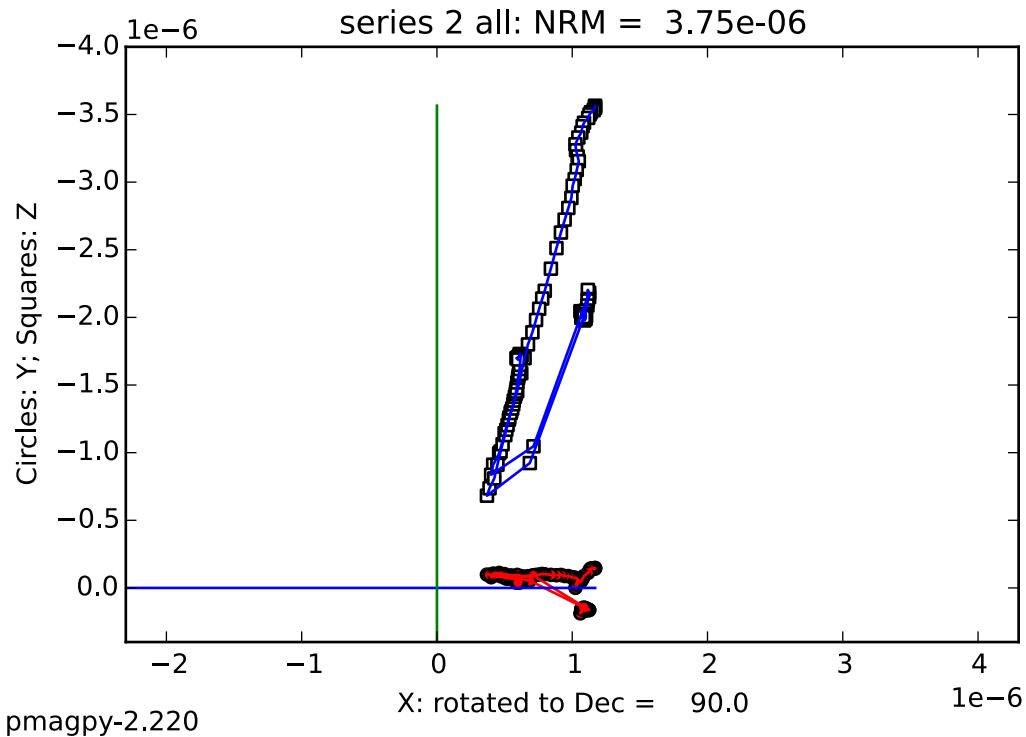
#plot with pmagplotlib.plotZ(fignum,datablock,angle,s, norm)
fignum = 1
pmagplotlib.plotZ(1,datablock_cooling,90,'series 2 cooling',0)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
plt.ylim(.0000004,-.000004,)
plt.xlim(0,.00002,)
plt.savefig('code_output/Ikon_magnetite_series2_zplot_cooling.pdf')
plt.show()

fignum = 2
pmagplotlib.plotZ(1,datablock_warming,90,'series 2 warming',0)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
plt.ylim(.0000004,-.000004,)
plt.xlim(0,.00002,)
plt.savefig('code_output/Ikon_magnetite_series2_zplot_warming.pdf')
plt.show()

fignum = 3
pmagplotlib.plotZ(1,datablock_all,90,'series 2 all',0)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
plt.ylim(.0000004,-.000004,)
plt.xlim(0,.00002,)
plt.savefig('code_output/Ikon_magnetite_series2_zplot.pdf')
plt.show()

```





5 Polycrystalline magnetite and hematite experiments

This sample is from a product marketed as “Nature Company hematite” (referred to as naco in the code and the data folder) that is a polycrystalline mix of hematite with magnetite impurity. Low-temperature cycling experiments on this specimen (cf-naco-1) were conducted on the IRM-LTI and the MPMS.

5.1 IRM-LTI experiment

```
In [27]: Naco_cooling = pd.read_csv('..../Data/NaCo/Naco_cooling.csv')
Naco_warming = pd.read_csv('..../Data/NaCo/Naco_warming.csv')
Naco_cooling.head()

Out[27]:    specimen      Mx [Am2]      My [Am2]      Mz [Am2]      MN [Am2]      ME [Am2]  \
0  cf-naco-1  0.000083 -4.335590e-08 -0.000003  0.000083 -4.335590e-08
1  cf-naco-1  0.000083 -4.275720e-08 -0.000003  0.000083 -4.275720e-08
2  cf-naco-1  0.000083 -4.030780e-08 -0.000003  0.000083 -4.030780e-08
3  cf-naco-1  0.000083 -5.599900e-08 -0.000003  0.000083 -5.599900e-08
4  cf-naco-1  0.000083 -5.100280e-08 -0.000003  0.000083 -5.100280e-08

      MV [Am2]      Mtot [Am2]      Dec [deg]      Inc [deg]      ...      time      series  \
0 -0.000003  0.000083 -0.029839 -2.17446      ...  10:41:36 AM      1
1 -0.000003  0.000083 -0.029466 -2.18214      ...  10:45:05 AM      1
2 -0.000003  0.000083 -0.027778 -2.17731      ...  10:46:28 AM      1
3 -0.000003  0.000083 -0.038592 -2.18009      ...  10:48:48 AM      1
4 -0.000003  0.000083 -0.035174 -2.17820      ...  11:32:19 AM      1
```

```

      position[cm]  Drift_ratio  Ja/Jr  run#  Description  Meas_T[K]      Mode \
0           NaN        NaN     0    NaN    292.893    291.28  DISCRETE
1           NaN        NaN     0    NaN    292.920    291.26  DISCRETE
2           NaN        NaN     0    NaN    292.929    291.26  DISCRETE
3           NaN        NaN     0    NaN    284.969    291.23  DISCRETE
4           NaN        NaN     0    NaN    181.799    290.19  DISCRETE

      Type
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN

[5 rows x 35 columns]

```

5.1.1 Blank run and background subtraction

A blank run with the probe empty was done close in time to the experiment on the cf-naco-a specimen. That data is imported below into a dataframe and then plotted to determine the best way to background subtract the probe measurement from the sample data.

```
In [28]: blank_naco_data_cooling = pd.read_csv('..../Data/NaCo/20110520cf_blank_cooling.csv')
blank_naco_data_warming = pd.read_csv('..../Data/NaCo/20110520cf_blank_warming.csv')
blank_naco_data_warming.head()
```

```

Out[28]:    specimen      Mx [Am2]      My [Am2]      Mz [Am2]      MN [Am2]      ME [Am2]  \
0  CF-blank -4.684120e-08 -6.720820e-08       0       0 -6.720820e-08
1  CF-blank -4.699740e-08 -6.693490e-08       0       0 -6.693490e-08
2  CF-blank -4.690780e-08 -6.685280e-08       0       0 -6.685280e-08
3  CF-blank -4.700080e-08 -6.680970e-08       0       0 -6.680970e-08
4  CF-blank -4.673360e-08 -6.666500e-08       0       0 -6.666500e-08

      MV [Am2]      Mtot [Am2]      Dec [deg]      Inc [deg]      ...      time      series \
0  4.684120e-08          0  -28.5181    18.4050      ...  1:17:33 PM      7
1  4.699740e-08          0  -28.4446    18.4917      ...  1:18:19 PM      7
2  4.690780e-08          0  -28.4037    18.4572      ...  1:19:10 PM      7
3  4.700080e-08          0  -28.4008    18.5009      ...  1:20:15 PM      7
4  4.673360e-08          0  -28.3368    18.4045      ...  1:21:35 PM      7

      position[cm]  Drift_ratio  Ja/Jr  run#  Description  Meas_T[K]      Mode \
0           NaN        NaN     0    NaN    15.227    17.63  DISCRETE
1           NaN        NaN     0    NaN    18.948    22.16  DISCRETE
2           NaN        NaN     0    NaN    20.988    24.07  DISCRETE
3           NaN        NaN     0    NaN    22.501    25.58  DISCRETE
4           NaN        NaN     0    NaN    23.577    27.08  DISCRETE

      Type
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN

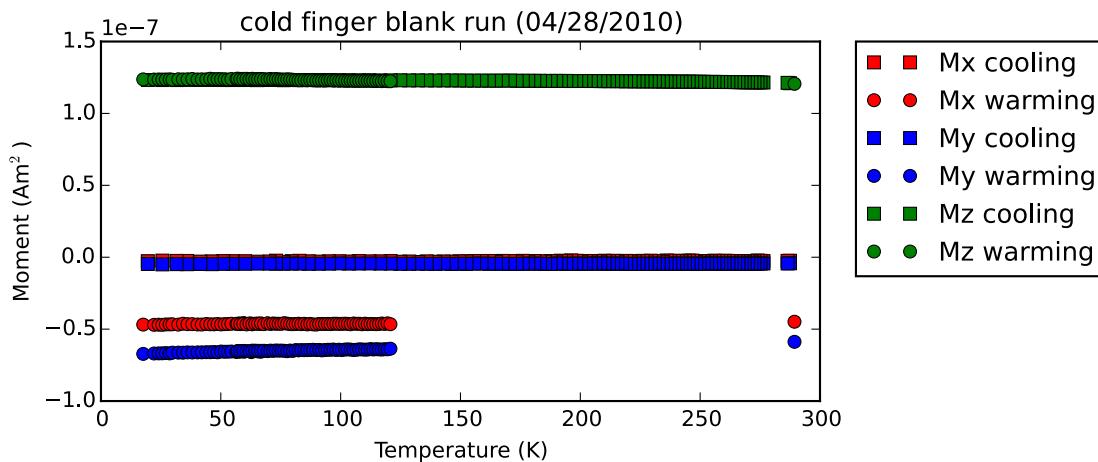
[5 rows x 35 columns]

```

```
In [29]: blank_naco_data_cooling = pd.read_csv('../Data/NaCo/20110520cf_blank_cooling.csv')
blank_naco_data_warming = pd.read_csv('../Data/NaCo/20110520cf_blank_warming.csv')
blank_naco_data_warming.head()
fignum=1
plt.figure(num=fignum, figsize=(6,3), dpi=160)
plt.plot(blank_naco_data_cooling['Meas_T[K]'], blank_naco_data_cooling['Mx[Am2]'], 'rs',
         label='Mx cooling')
plt.plot(blank_naco_data_warming['Meas_T[K]'], blank_naco_data_warming['Mx[Am2]'], 'ro',
         label='Mx warming')
plt.plot(blank_naco_data_cooling['Meas_T[K]'], blank_naco_data_cooling['My[Am2]'], 'bs',
         label='My cooling')
plt.plot(blank_naco_data_warming['Meas_T[K]'], blank_naco_data_warming['My[Am2]'], 'bo',
         label='My warming')
plt.plot(blank_naco_data_cooling['Meas_T[K]'], blank_naco_data_cooling['Mz[Am2]'], 'gs',
         label='Mz cooling')
plt.plot(blank_naco_data_warming['Meas_T[K]'], blank_naco_data_warming['Mz[Am2]'], 'go',
         label='Mz warming')

plt.ylabel('Moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('cold finger blank run (04/28/2010)')

plt.show()
```



The cooling run was completed in full for the blank while there were gaps in the warming data. Let's use the cooling data which looks to be reversible for the z (while there is an unexplained offset in the y and x) for the background correction using the `background_subtraction()` function.

```
In [30]: #For the interpolate function to work the x values (temperature) need to be ascending.
#The blank_hem_data_cooling can be sorted so that temperatures are ascending
blank_naco_data_cooling = blank_naco_data_cooling.sort(['Meas_T[K]'])

Naco_cooling['Mx_bsub[Am2]'] = background_sub(blank_naco_data_cooling['Mx[Am2]'],
                                                blank_naco_data_cooling['Meas_T[K]'],
                                                Naco_cooling['Mx[Am2]'],
                                                Naco_cooling['Meas_T[K]'])
```

```

Naco_cooling['My_bsub[Am2]'] = background_sub(blank_naco_data_cooling['My[Am2]'],
                                              blank_naco_data_cooling['Meas_T[K]'],
                                              Naco_cooling['My[Am2]'],
                                              Naco_cooling['Meas_T[K]'])

Naco_cooling['Mz_bsub[Am2]'] = background_sub(blank_naco_data_cooling['Mz[Am2]'],
                                              blank_naco_data_cooling['Meas_T[K]'],
                                              Naco_cooling['Mz[Am2]'],
                                              Naco_cooling['Meas_T[K]'])

Naco_cooling['Mtotal_bsub[Am2]'] = np.sqrt(Naco_cooling['Mx_bsub[Am2]']**2 +
                                             Naco_cooling['My_bsub[Am2]']**2 +
                                             Naco_cooling['Mz_bsub[Am2]']**2)

Naco_warming['Mx_bsub[Am2]'] = background_sub(blank_naco_data_cooling['Mx[Am2]'],
                                               blank_naco_data_cooling['Meas_T[K]'],
                                               Naco_warming['Mx[Am2]'],
                                               Naco_warming['Meas_T[K]'])

Naco_warming['My_bsub[Am2]'] = background_sub(blank_naco_data_cooling['My[Am2]'],
                                               blank_naco_data_cooling['Meas_T[K]'],
                                               Naco_warming['My[Am2]'],
                                               Naco_warming['Meas_T[K]'])

Naco_warming['Mz_bsub[Am2]'] = background_sub(blank_naco_data_cooling['Mz[Am2]'],
                                               blank_naco_data_cooling['Meas_T[K]'],
                                               Naco_warming['Mz[Am2]'],
                                               Naco_warming['Meas_T[K]'])

Naco_warming['Mtotal_bsub[Am2]'] = np.sqrt(Naco_warming['Mx_bsub[Am2]']**2 +
                                             Naco_warming['My_bsub[Am2]']**2 +
                                             Naco_warming['Mz_bsub[Am2]']**2)

```

5.2 Equal area plot

The direction of the Naco specimen throughout thermal cycling on the LTI can be plotted on an equal area plot. A subsection of this equal area plot was included in the figure concerning the Naco hematite/magnetite specimen in the main text.

```

In [31]: directions_cart_cooling=[]
          directions_cart_warming=[]

          #data point 0 is oriented (remember python starts at 0)
          for n in range(len(Naco_cooling)):
              directions_cart_cooling.append([Naco_cooling['Mx[Am2]'][n],
                                              Naco_cooling['My[Am2]'][n],
                                              Naco_cooling['Mz[Am2]'][n]])
          for n in range(len(Naco_warming)):
              directions_cart_warming.append([Naco_warming['Mx[Am2]'][n],
                                              Naco_warming['My[Am2]'][n],
                                              Naco_warming['Mz[Am2]'][n]])

          directions_DI_cooling=pmag.cart2dir(directions_cart_cooling)
          directions_DI_warming=pmag.cart2dir(directions_cart_warming)

```

```

directions_cart_cooling=[]
directions_cart_warming=[]

#data point 0 is oriented (remember python starts at 0)
for n in range(len(Naco_cooling)):
    directions_cart_cooling.append([Naco_cooling['Mx_bsub[Am2]'][n],
                                    Naco_cooling['My_bsub[Am2]'][n],
                                    Naco_cooling['Mz_bsub[Am2]'][n]])
for n in range(len(Naco_warming)):
    directions_cart_warming.append([Naco_warming['Mx_bsub[Am2]'][n],
                                    Naco_warming['My_bsub[Am2]'][n],
                                    Naco_warming['Mz_bsub[Am2]'][n]])

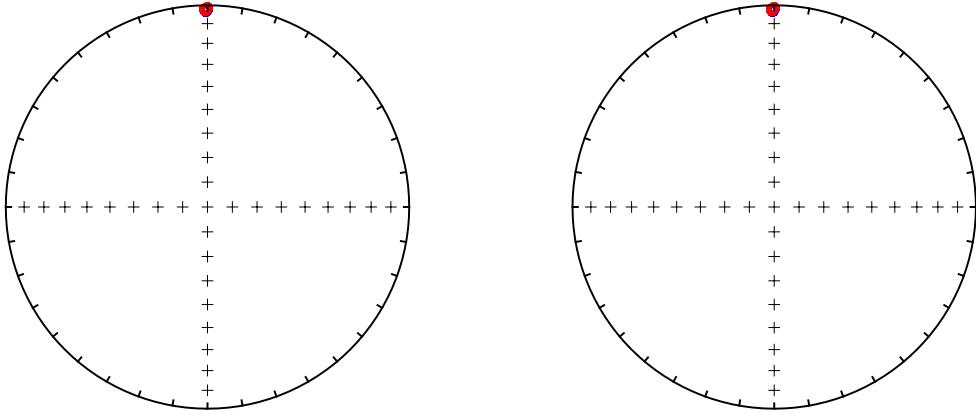
directions_DI_cooling=pmag.cart2dir(directions_cart_cooling)
directions_DI_warming=pmag.cart2dir(directions_cart_warming)

fignum=1
fig=plt.figure(figsize=(8,5),dpi=160)
plt.subplot(121)
plt.figure(num=fignum,figsize=(8,8),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(directions_DI_cooling,'b')
IPmag.iplotDI(directions_DI_warming,'r')
plt.title('NaCo low-t cycling (no background correction)')

plt.subplot(122)
plt.figure(num=fignum,figsize=(8,8),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(directions_DI_cooling,'b')
IPmag.iplotDI(directions_DI_warming,'r')
plt.title('NaCo low-t cycling (background corrected)')
plt.savefig('code_output/Naco_directions.pdf')
plt.show()

```

NaCo low-t cycling (no background correction) NaCo low-t cycling (background corrected)



5.3 Comparision between IRM-LTI and MPMS data

Having subtracted the holder off of the LTI experiment data, the background subtracted data is plotted below along with the data acquired during the low-temperature cycling experiment on the MPMS.

```
In [32]: Naco_MPMS = pd.read_csv('../Data/NaCo/Naco_MPMS.csv', header=3)
Naco_MPMS.head()
```

	T [K]	Bapp [T]	M [Am ² /kg]	reg fit	timestamp	Unnamed: 5
0	NaN	NaN	NaN	NaN	NaN	NaN
1	300.0231	0	0.744367	0.974488	5/27/2011 2:06:42 PM	NaN
2	294.4838	0	0.744338	0.974525	5/27/2011 2:10:08 PM	NaN
3	289.3426	0	0.742235	0.974490	5/27/2011 2:11:13 PM	NaN
4	284.1801	0	0.739180	0.974057	5/27/2011 2:12:05 PM	NaN

	T [K].1	Bapp [T].1	M [Am ² /kg].1	reg fit.1	timestamp.1
0	NaN	NaN	NaN	NaN	NaN
1	14.99718	0	0.079222	0.974658	5/27/2011 3:28:22 PM
2	20.55822	0	0.079240	0.974714	5/27/2011 3:29:41 PM
3	25.55299	0	0.079243	0.974558	5/27/2011 3:30:42 PM
4	30.67751	0	0.079225	0.974482	5/27/2011 3:31:44 PM

```
In [33]: adjustprops = dict(left=0.1, bottom=0.1, right=0.97, top=0.93, wspace=0.25, hspace=0.25)
fig=plt.figure(figsize=(10,9))
fig.subplots_adjust(**adjustprops)
ax1 = plt.subplot(221)

ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['Mx_bsub[Am2]'], 'rs-', label='M_x$ cooling')
ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['My_bsub[Am2]'], 'bs-', label='M_y$ cooling')
```

```

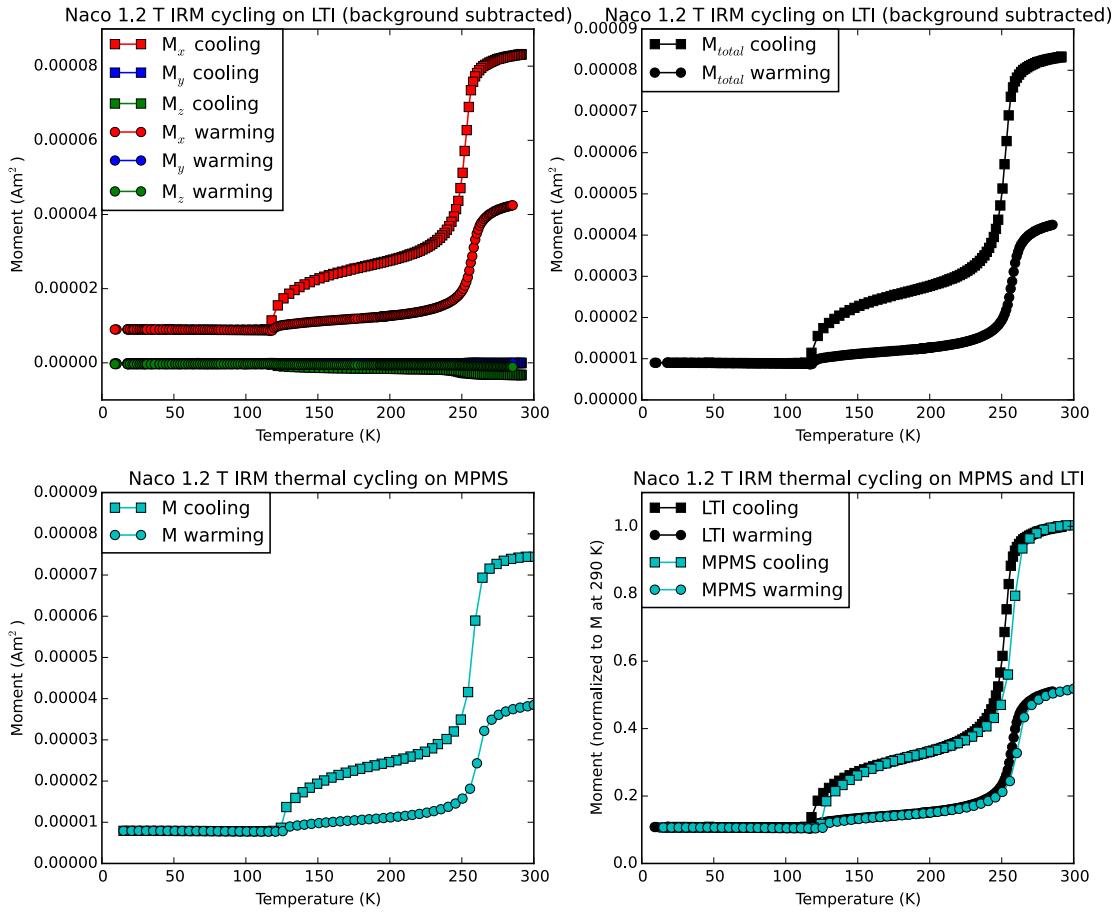
ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['Mz_bsub[Am2]'], 'gs-', label='M_z$ cooling')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['Mx_bsub[Am2]'], 'ro-', label='M_x$ warming')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['My_bsub[Am2]'], 'bo-', label='M_y$ warming')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['Mz_bsub[Am2]'], 'go-', label='M_z$ warming')
plt.legend(loc=2, borderaxespad=0.)
plt.ylabel('Moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.title('Naco 1.2 T IRM cycling on LTI (background subtracted)')

ax2 = plt.subplot(222)
ax2.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['Mtotal_bsub[Am2]'],
          'ks-', label='M_{total}$ cooling')
ax2.plot(Naco_warming['Meas_T[K]'], Naco_warming['Mtotal_bsub[Am2]'],
          'ko-', label='M_{total}$ warming')
plt.legend(loc=2, borderaxespad=0.)
plt.ylabel('Moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.title('Naco 1.2 T IRM cycling on LTI (background subtracted)')

ax3 = plt.subplot(223)
ax3.plot(Naco_MPMS['T [K]'], Naco_MPMS['M [Am2/kg]']/10000, 'cs-', label='M cooling')
ax3.plot(Naco_MPMS['T [K].1'], Naco_MPMS['M [Am2/kg].1']/10000, 'co-', label='M warming')
plt.legend(loc=2, borderaxespad=0.)
plt.ylabel('Moment (Am$^2$)')
plt.ylim(0,0.00009)
plt.xlabel('Temperature (K)')
plt.xlim(0,300)
plt.title('Naco 1.2 T IRM thermal cycling on MPMS')

ax4 = plt.subplot(224)
ax4.plot(Naco_cooling['Meas_T[K]'],
          Naco_cooling['Mtotal_bsub[Am2]']/Naco_cooling['Mtotal_bsub[Am2]'][0],
          'ks-', label='LTI cooling')
ax4.plot(Naco_warming['Meas_T[K]'],
          Naco_warming['Mtotal_bsub[Am2]']/Naco_cooling['Mtotal_bsub[Am2]'][0],
          'ko-', label='LTI warming')
ax4.plot(Naco_MPMS['T [K]'],
          Naco_MPMS['M [Am2/kg]']/Naco_MPMS['M [Am2/kg]'][3],
          'cs-', label='MPMS cooling')
ax4.plot(Naco_MPMS['T [K].1'],
          Naco_MPMS['M [Am2/kg].1']/Naco_MPMS['M [Am2/kg]'][3],
          'co-', label='MPMS warming')
plt.ylabel('Moment (normalized to M at 290 K)')
plt.xlabel('Temperature (K)')
plt.legend(loc=2, borderaxespad=0.)
plt.xlim(0,300)
plt.ylim(ymin=0)
plt.title('Naco 1.2 T IRM thermal cycling on MPMS and LTI')
plt.show()

```



The total magnetic moment between the LTI and MPMS are quite similar. The values obtained on the LTI are slightly higher which could be associated with there being small off-axis components that were not resolved using the MPMS. The code block below displays the values of the magnetic moments at ~290 degrees on both instruments. These values are written on the figure in the main manuscript.

```
In [34]: #values quoted on figure
a = Naco_cooling['Mtotal_bsub[Am2]'][0]
b = Naco_cooling['Meas_T[K]'][5]
print "For the temperature of", b, "on the LTI the magnetic moment is:", a, "Am2"
c = Naco_MPMS['T [K]'][3]
d = Naco_MPMS['M [Am2/kg]'][3]/10000
print "For the temperature of", c, "on the MPMS the magnetic moment is:", d, "Am2"
print "Number of data points collected during LTI run:", len(Naco_cooling['Meas_T[K]'])
```

For the temperature of 289.13 on the LTI the magnetic moment is: 8.33181600855e-05 Am2
 For the temperature of 289.3426 on the MPMS the magnetic moment is: 7.422345e-05 Am2
 Number of data points collected during LTI run: 114

A simplified version of the figure made above needs to be generated to use within the main manuscript.

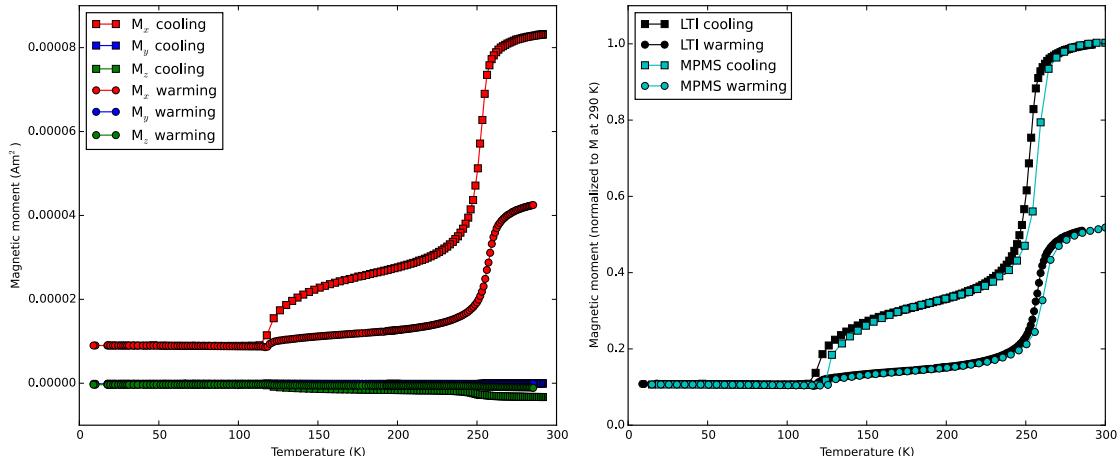
```
In [35]: adjustprops = dict(left=0.1, bottom=0.1, right=0.97, top=0.93, wspace=0.15, hspace=0.1)
fig=plt.figure(figsize=(14,6),dpi=160)
fig.subplots_adjust(**adjustprops)
```

```

ax1 = plt.subplot(121)
ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['Mx_bsub[Am2]'],
          'rs-', label='M$_x$ cooling')
ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['My_bsub[Am2]'],
          'bs-', label='M$_y$ cooling')
ax1.plot(Naco_cooling['Meas_T[K]'], Naco_cooling['Mz_bsub[Am2]'],
          'gs-', label='M$_z$ cooling')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['Mx_bsub[Am2]'],
          'ro-', label='M$_x$ warming')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['My_bsub[Am2]'],
          'bo-', label='M$_y$ warming')
ax1.plot(Naco_warming['Meas_T[K]'], Naco_warming['Mz_bsub[Am2]'],
          'go-', label='M$_z$ warming')
plt.legend(loc=2)
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
# plt.title('thermal cycling on IRM-LTI (background subtracted)')

ax2 = plt.subplot(122)
ax2.plot(Naco_cooling['Meas_T[K]'],
          Naco_cooling['Mtotal_bsub[Am2]']/Naco_cooling['Mtotal_bsub[Am2]'][0],
          'ks-', label='LTI cooling')
ax2.plot(Naco_warming['Meas_T[K]'],
          Naco_warming['Mtotal_bsub[Am2]']/Naco_cooling['Mtotal_bsub[Am2]'][0],
          'ko-', label='LTI warming')
ax2.plot(Naco_MPMS['T [K]'], Naco_MPMS['M [Am2/kg]']/Naco_MPMS['M [Am2/kg]'][3],
          'cs-', label='MPMS cooling')
ax2.plot(Naco_MPMS['T [K].1'], Naco_MPMS['M [Am2/kg].1']/Naco_MPMS['M [Am2/kg]'][3],
          'co-', label='MPMS warming')
plt.ylabel('Magnetic moment (normalized to M at 290 K)')
plt.xlabel('Temperature (K)')
# plt.title('thermal cycling on IRM-LTI and MPMS')
plt.legend(loc=2)
plt.xlim(0,300)
plt.ylim(ymin=0)
plt.savefig('code_output/Naco_LTcycle.pdf')
plt.show()

```



5.3.1 Estimating Morin and Verwey transition temperatures

In order to estimate the Morin and Verwey transition temperatures, the gradient of the cooling data can be plotted.

```
In [36]: gradient_LTI_cooling = np.gradient(Naco_cooling['Mtotal_bsub[Am2]'])

from scipy.interpolate import interp1d

#interpolate the data using a cubic spline
f_LTI_cooling_inter = interp1d(Naco_cooling['Meas_T[K]'][3:],
                                gradient_LTI_cooling[3:],
                                kind='cubic')

xnew = np.linspace(80,280,1000)

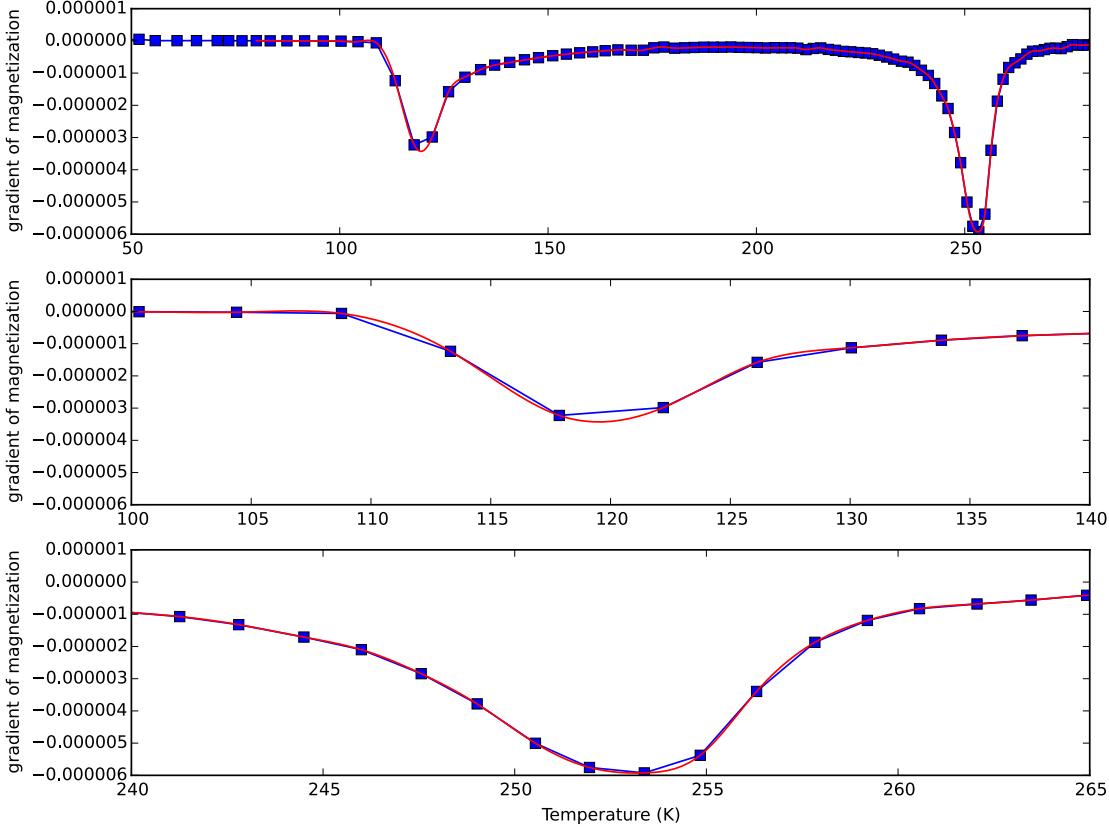
gradient_LTI_cooling_interpolated = f_LTI_cooling_inter(xnew)

plt.figure(figsize=(10,8),dpi=160)
plt.subplot(311)
plt.plot(Naco_cooling['Meas_T[K]'], gradient_LTI_cooling,
         'bs-',label='LTI gradient on cooling')
plt.plot(xnew, gradient_LTI_cooling_interpolated, 'r-')
plt.xlim(50, 280)
plt.ylabel('gradient of magnetization')

plt.subplot(312)
plt.plot(Naco_cooling['Meas_T[K]'], gradient_LTI_cooling,
         'bs-',label='LTI gradient on cooling')
plt.plot(xnew, gradient_LTI_cooling_interpolated, 'r-')
plt.xlim(100, 140)
plt.ylabel('gradient of magnetization')

plt.subplot(313)
plt.plot(Naco_cooling['Meas_T[K]'], gradient_LTI_cooling,
         'bs-',label='LTI gradient on cooling')
plt.plot(xnew, gradient_LTI_cooling_interpolated, 'r-')
plt.xlim(240, 265)
plt.ylabel('gradient of magnetization')

plt.xlabel('Temperature (K)')
plt.show()
```



```
In [37]: LTI_morin = fmin(f_LTI_cooling_inter, x0=252)
      print 'The Morin estimate using LTI cooling data is: ' + str(LTI_morin)
      print ""
      LTI_verwey = fmin(f_LTI_cooling_inter, x0=118)
      print 'The Verwey estimate using LTI cooling data is: ' + str(LTI_verwey)
      print ""
```

```
Optimization terminated successfully.
      Current function value: -0.000006
      Iterations: 18
      Function evaluations: 36
The Morin estimate using LTI cooling data is: [ 253.17009888]

Optimization terminated successfully.
      Current function value: -0.000003
      Iterations: 17
      Function evaluations: 34
The Verwey estimate using LTI cooling data is: [ 119.51677246]
```

6 Pyrrhotite experiments

6.1 IRM-LTI experiments

There were three experiments performed on this single crystal of pyrrhotite:

- an isothermal remanent magnetization bisecting the c and a_1 axes (series 5)
- an isothermal remanent magnetization along the c axis (series 6)
- an isothermal remanent magnetization bisecting the a_1 and $-a_3$ axes (series 7)

Import the data into a dataframe

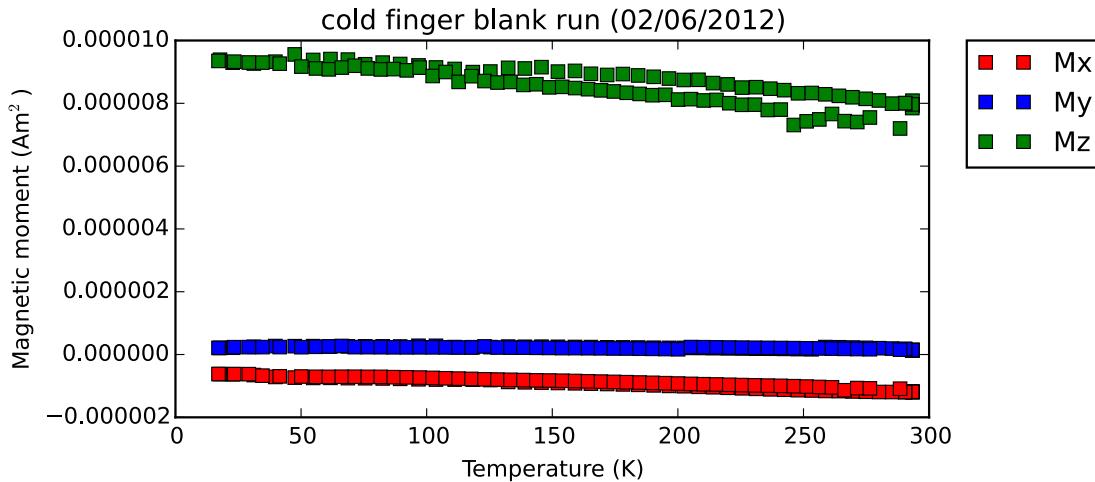
```
In [38]: pyrrhotite_data = pd.read_csv('../Data/Pyrrhotite/Pyrrhotite_Experiments.csv')
```

6.1.1 Blank run and background subtraction

A blank run of the empty coldfinger probe was conducted on 2/6/2012 (the day before the series 5 experiment). The data can be loaded into a dataframe.

```
In [39]: blank_for_pyrrhotite = pd.read_csv('../Data/Pyrrhotite/cf_blank.csv')
```

```
plt.figure(figsize=(6,3),dpi=160)
plt.plot(blank_for_pyrrhotite['measurement_T[K]'], blank_for_pyrrhotite['Mx[Am2]'],
         'rs', label='Mx')
plt.plot(blank_for_pyrrhotite['measurement_T[K]'], blank_for_pyrrhotite['My[Am2]'],
         'bs', label='My')
plt.plot(blank_for_pyrrhotite['measurement_T[K]'], blank_for_pyrrhotite['Mz[Am2]'],
         'gs', label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('cold finger blank run (02/06/2012)')
plt.show()
```



The probe's magnetic moment appears to be largely reversible. Let's use the values upon cooling of the blank probe for the background correction of the pyrrhotite experimental data.

```
In [40]: blank_for_pyrrhotite_cooling = blank_for_pyrrhotite[0:51]
blank_for_pyrrhotite_cooling = blank_for_pyrrhotite_cooling.sort(['measurement_T[K]'])

pyrrhotite_data['Mx_bsub[Am2]'] = background_sub(blank_for_pyrrhotite_cooling['Mx[Am2]'],
                                                blank_for_pyrrhotite_cooling['measurement_T[K]'])
```

```

pyrrhotite_data['Mx [Am2]'],
pyrrhotite_data['Meas_T [K]'])
pyrrhotite_data['My_bsub [Am2]'] = background_sub(blank_for_pyrrhotite_cooling['My [Am2]'],
blank_for_pyrrhotite_cooling['measurement_T [K]'],
pyrrhotite_data['My [Am2]'],
pyrrhotite_data['Meas_T [K]']))
pyrrhotite_data['Mz_bsub [Am2]'] = background_sub(blank_for_pyrrhotite_cooling['Mz [Am2]'],
blank_for_pyrrhotite_cooling['measurement_T [K]'],
pyrrhotite_data['Mz [Am2]'],
pyrrhotite_data['Meas_T [K]']))

pyrrhotite_data['Mtotal_bsub [Am2]'] = np.sqrt(pyrrhotite_data['Mx_bsub [Am2]']**2 +
pyrrhotite_data['My_bsub [Am2]']**2 +
pyrrhotite_data['Mz_bsub [Am2]']**2)

```

6.1.2 3-axis and total magnetization plots

Split the data into series 5, 6 and 7, reset the indices for each series and view the first 5 rows of pyrrhotite_data_series7.

```
In [41]: pyrrhotite_data_series5 = pyrrhotite_data.ix[pyrrhotite_data.series==5]
pyrrhotite_data_series6 = pyrrhotite_data.ix[pyrrhotite_data.series==6]
pyrrhotite_data_series6.reset_index(inplace=True)
pyrrhotite_data_series7 = pyrrhotite_data.ix[pyrrhotite_data.series==7]
pyrrhotite_data_series7.reset_index(inplace=True)
pyrrhotite_data_series7.head()
```

```
Out[41]:   index          specimen    Mx [Am2]    My [Am2]    Mz [Am2]    MN [Am2]  \
0    230    CF AA pyrrhotite-5  0.000130  0.000063 -0.000029  0.000130
1    231    CF AA pyrrhotite-5  0.000148  0.000007 -0.000051  0.000148
2    232    CF AA pyrrhotite-5  0.000148  0.000007 -0.000051  0.000148
3    233    CF AA pyrrhotite-5  0.000148  0.000007 -0.000051  0.000148
4    234    CF AA pyrrhotite-5  0.000148  0.000007 -0.000051  0.000148

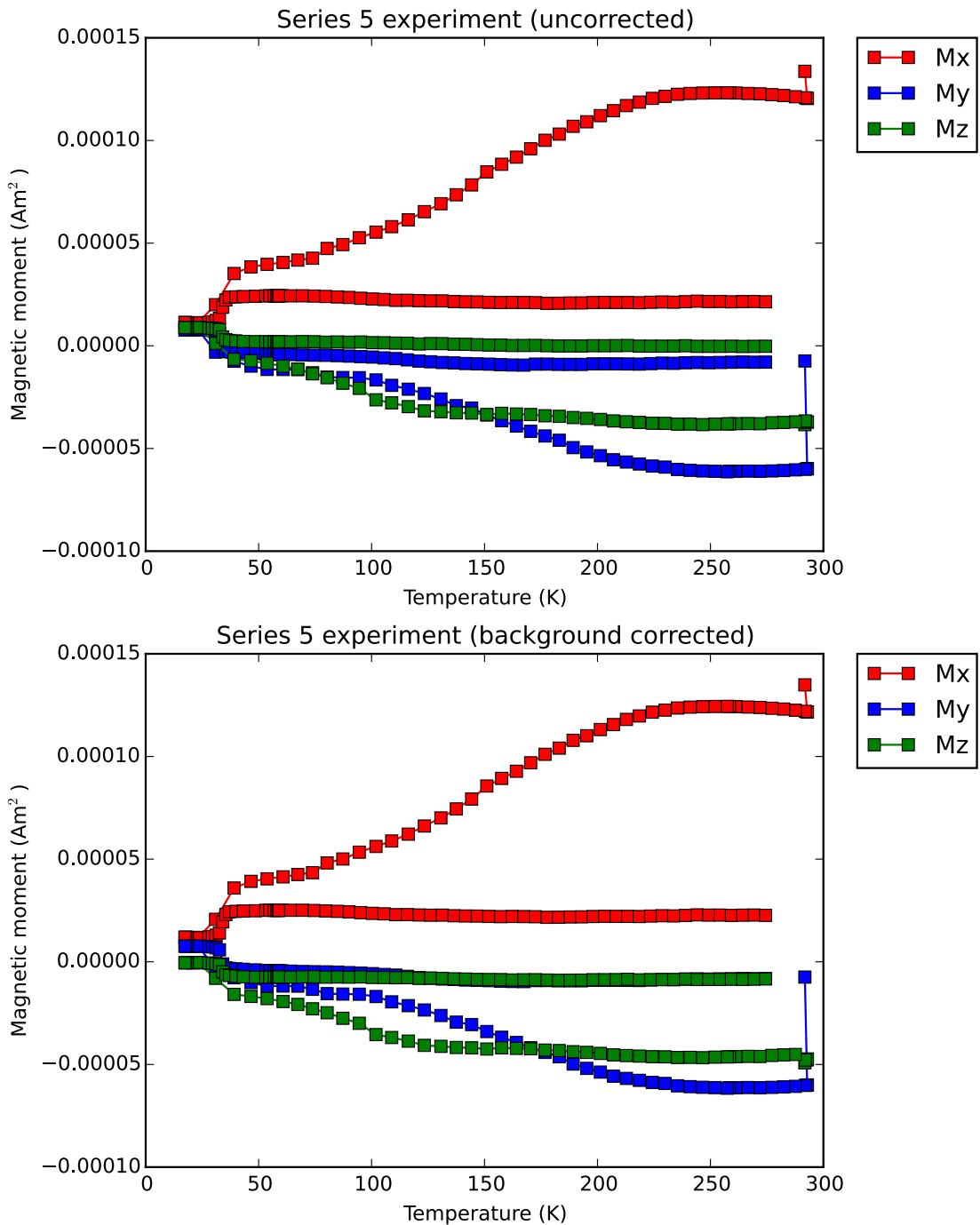
      ME [Am2]    MV [Am2]    Mtot [Am2]    Dec [deg]    ...    Ja/Jr    run#  \
0  0.000063 -0.000029  0.000147  25.83010    ...        0    NaN
1  0.000007 -0.000051  0.000156  2.89964    ...        0    NaN
2  0.000007 -0.000051  0.000156  2.88449    ...        0    NaN
3  0.000007 -0.000051  0.000156  2.87935    ...        0    NaN
4  0.000007 -0.000051  0.000156  2.87825    ...        0    NaN

      Description  Meas_T [K]      Mode  Type  Mx_bsub [Am2]  My_bsub [Am2]  \
0      289.42    290.745  DISCRETE  NaN  0.000131  0.000063
1      290.98    291.961  DISCRETE  NaN  0.000149  0.000007
2      291.24    292.114  DISCRETE  NaN  0.000149  0.000007
3      291.33    292.170  DISCRETE  NaN  0.000149  0.000007
4      291.40    292.230  DISCRETE  NaN  0.000149  0.000007

      Mz_bsub [Am2]  Mtotal_bsub [Am2]
0      -0.000037    0.000150
1      -0.000062    0.000162
2      -0.000062    0.000161
3      -0.000062    0.000161
4      -0.000062    0.000161
```

```
[5 rows x 40 columns]
```

```
In [42]: plt.figure(figsize=(6,10),dpi=160)
plt.subplot(211)
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['Mx[Am2]'],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['My[Am2]'],
         'bs-', label='My')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['Mz[Am2]'],
         'gs-', label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 5 experiment (uncorrected)')
plt.subplot(212)
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['Mx_bsub[Am2]'],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['My_bsub[Am2]'],
         'bs-', label='My')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'], pyrrhotite_data_series5['Mz_bsub[Am2]'],
         'gs-', label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 5 experiment (background corrected)')
plt.show()
```

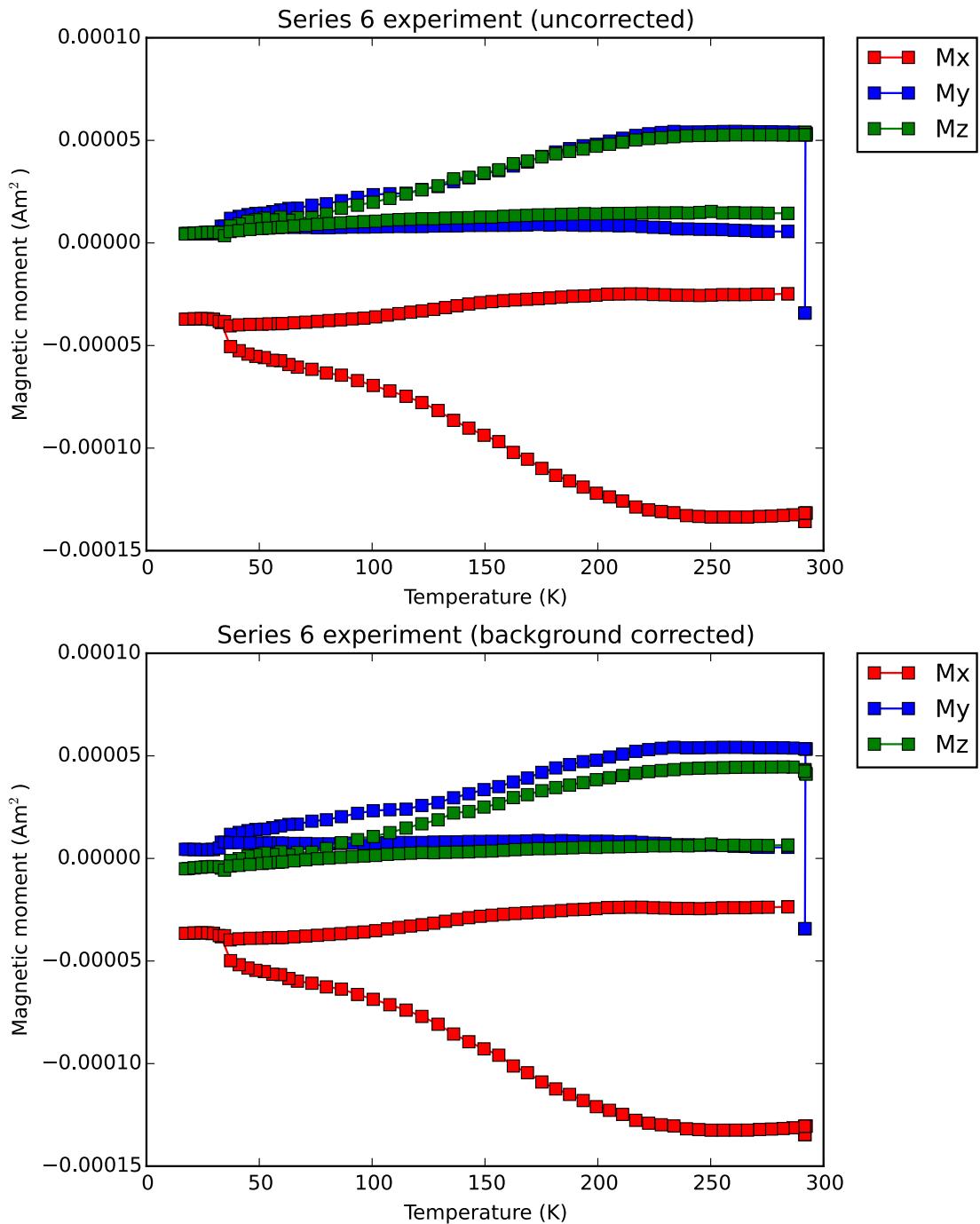


```
In [43]: plt.figure(figsize=(6,10),dpi=160)
plt.subplot(211)
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['Mx[Am2]'],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['My[Am2]'],
         'bs-', label='My')
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['Mz[Am2]'],
         'gs-', label='Mz')
```

```

    'gs-', label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 6 experiment (uncorrected)')
plt.subplot(212)
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['Mx_bsub[Am2]'],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['My_bsub[Am2]'],
         'bs-', label='My')
plt.plot(pyrrhotite_data_series6['Meas_T[K]'], pyrrhotite_data_series6['Mz_bsub[Am2]'],
         'gs-', label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 6 experiment (background corrected)')
plt.show()

```

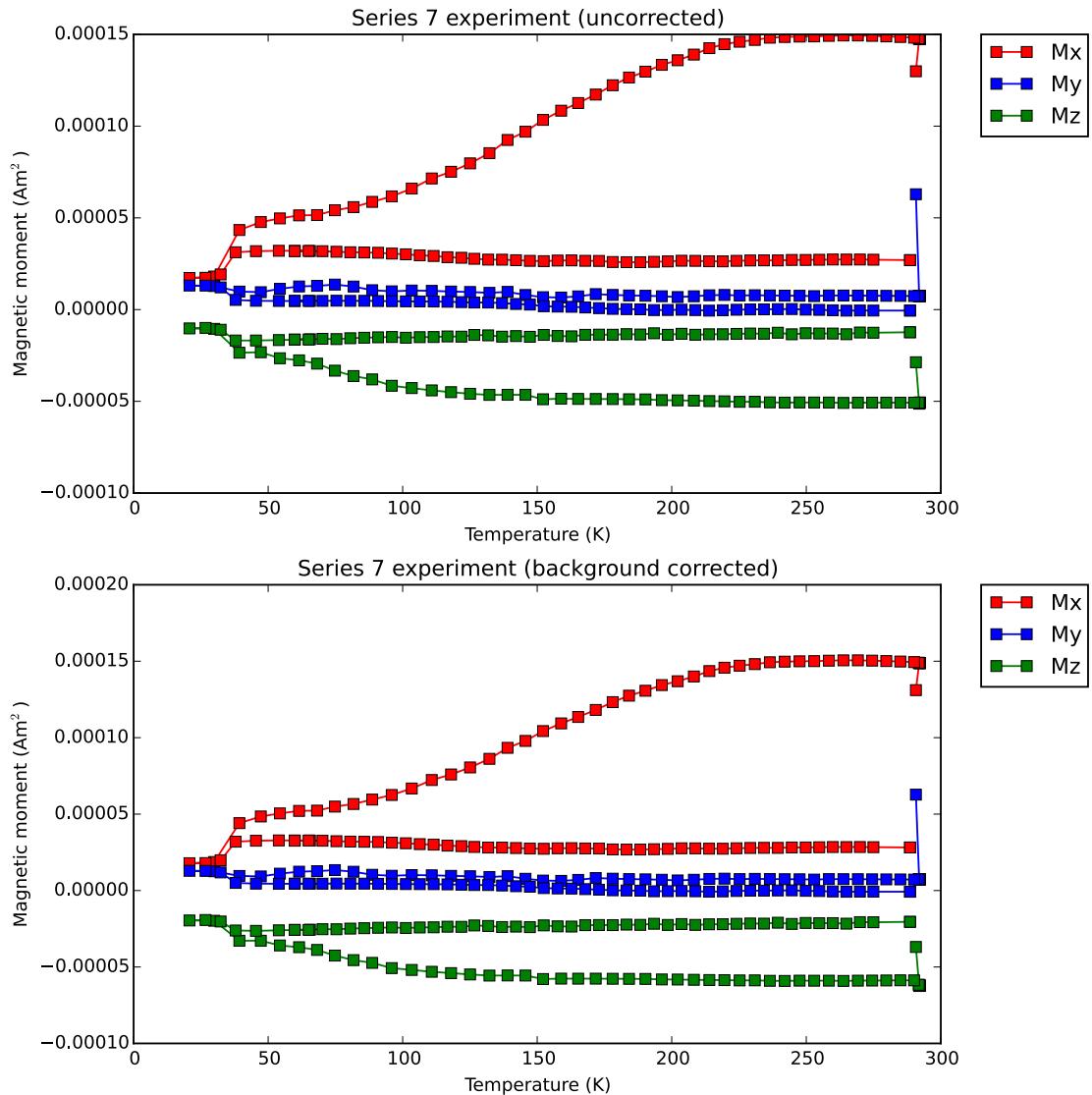


```
In [44]: plt.figure(figsize=(8,10),dpi=160)
plt.subplot(211)
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['Mx[Am2]'],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['My[Am2]'],
         'bs-', label='My')
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['Mz[Am2]'],
         'gs-', label='Mz')
```

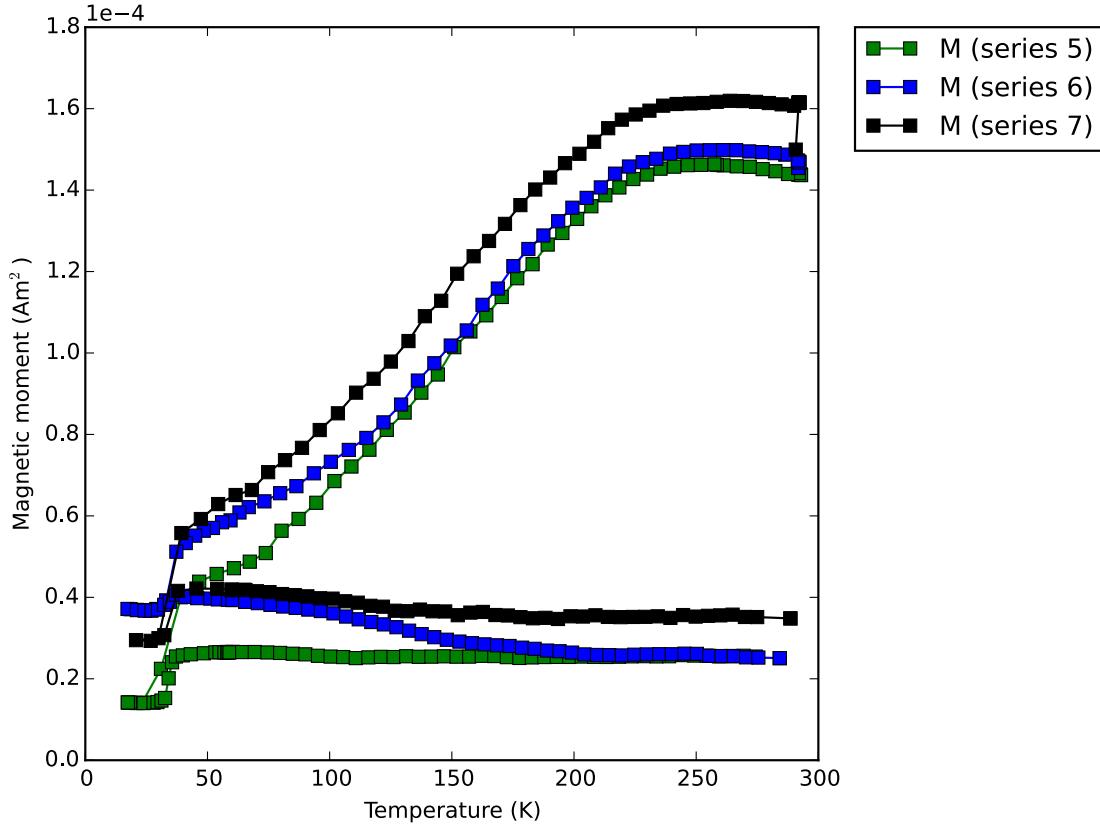
```

'gs-' , label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 7 experiment (uncorrected)')
plt.subplot(212)
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['Mx_bsub[Am2]'],
         'rs-' , label='Mx')
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['My_bsub[Am2]'],
         'bs-' , label='My')
plt.plot(pyrrhotite_data_series7['Meas_T[K]'], pyrrhotite_data_series7['Mz_bsub[Am2]'],
         'gs-' , label='Mz')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Series 7 experiment (background corrected)')
plt.show()

```



```
In [45]: plt.figure(figsize=(6,6),dpi=160)
plt.plot(pyrrhotite_data_series5['Meas_T[K]'],
         pyrrhotite_data_series5['Mtotal_bsub[Am2]'],
         'gs-', label='M (series 5)')
plt.plot(pyrrhotite_data_series6['Meas_T[K]'],
         pyrrhotite_data_series6['Mtotal_bsub[Am2]'],
         'bs-', label='M (series 6)')
plt.plot(pyrrhotite_data_series7['Meas_T[K]'],
         pyrrhotite_data_series7['Mtotal_bsub[Am2]'],
         'ks-', label='M (series 7)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.savefig('code_output/pyrrhotite_totalM_allseries.pdf')
plt.show()
```



```
In [46]: pyrrhotite_data_series7['Meas_T[K]']
```

```
Out[46]: 0      290.745
          1      291.961
```

```

2    292.114
3    292.170
4    292.230
5    292.252
6    292.283
7    291.859
8    291.835
9    290.075
10   284.960
11   279.741
12   274.496
13   269.182
14   263.808
...
89   213.871
90   218.972
91   224.069
92   229.171
93   234.283
94   239.450
95   244.630
96   249.785
97   254.831
98   259.870
99   264.904
100  269.936
101  274.958
102  288.559
103  288.568
Name: Meas_T[K], Length: 104, dtype: float64

```

```

In [47]: s5_temp_290 = pyrrhotite_data_series5['Meas_T[K]'][3]
s5_moment_290 = pyrrhotite_data_series5['Mtotal_bsub[Am2]'][3]

s5_temp_20 = pyrrhotite_data_series5['Meas_T[K]'][49]
s5_moment_20 = pyrrhotite_data_series5['Mtotal_bsub[Am2]'][49]

s5_temp_last = pyrrhotite_data_series5['Meas_T[K]'][114]
s5_moment_last = pyrrhotite_data_series5['Mtotal_bsub[Am2]'][114]

print 'For the series 5 experiment'
print "The percentage remanence remaining after cooling from " + str(s5_temp_290) + " to " + s
print s5_moment_20/s5_moment_290*100

print "The percentage remanence remaining after thermal cycling back to " + str(s5_temp_last)
print s5_moment_last/s5_moment_290*100

s6_temp_290 = pyrrhotite_data_series6['Meas_T[K]'][6]
s6_moment_290 = pyrrhotite_data_series6['Mtotal_bsub[Am2]'][6]

s6_temp_20 = pyrrhotite_data_series6['Meas_T[K]'][56]
s6_moment_20 = pyrrhotite_data_series6['Mtotal_bsub[Am2]'][56]

s6_temp_last = pyrrhotite_data_series6['Meas_T[K]'][114]

```

```

s6_moment_last = pyrrhotite_data_series6['Mtotal_bsub[Am2]'][114]

print ""
print 'For the series 6 experiment'
print "The percentage remanence remaining after cooling from " + str(s6_temp_290) + " to " + s
print s6_moment_20/s6_moment_290*100

print "The percentage remanence remaining after thermal cycling back to " + str(s5_temp_last)
print s6_moment_last/s6_moment_290*100

s7_temp_290 = pyrrhotite_data_series7['Meas_T[K]'][9]
s7_moment_290 = pyrrhotite_data_series7['Mtotal_bsub[Am2]'][9]

s7_temp_20 = pyrrhotite_data_series7['Meas_T[K]'][51]
s7_moment_20 = pyrrhotite_data_series7['Mtotal_bsub[Am2]'][51]

s7_temp_last = pyrrhotite_data_series7['Meas_T[K]'][103]
s7_moment_last = pyrrhotite_data_series7['Mtotal_bsub[Am2]'][103]

print ""
print 'For the series 6 experiment'
print "The percentage remanence remaining after cooling from " + str(s7_temp_290) + " to " + s
print s7_moment_20/s7_moment_290*100

print "The percentage remanence remaining after thermal cycling back to " + str(s5_temp_last)
print s7_moment_last/s7_moment_290*100

```

For the series 5 experiment

The percentage remanence remaining after cooling from 292.01 to 20.604 is
9.75562057483

The percentage remanence remaining after thermal cycling back to 274.388 is
17.6555731402

For the series 6 experiment

The percentage remanence remaining after cooling from 291.577 to 18.551 is
25.1795062718

The percentage remanence remaining after thermal cycling back to 274.388 is
17.0246798302

For the series 6 experiment

The percentage remanence remaining after cooling from 290.075 to 20.758 is
18.349355273

The percentage remanence remaining after thermal cycling back to 274.388 is
21.6751911695

In each of the experiments the Besnus transition is prominent. Let's make a plot that zooms in the transition to highlight this aspect of the datasets. The series 5 warming data has a lot of points across the transition.

```

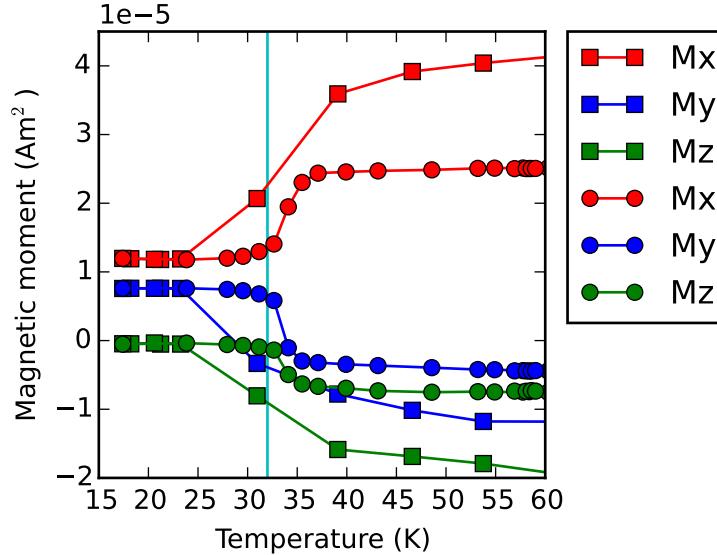
In [48]: plt.figure(figsize=(3,3),dpi=160)
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][:52],
         pyrrhotite_data_series5['Mx_bsub[Am2]'][:52],
         'rs-', label='Mx')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][:52],
         pyrrhotite_data_series5['My_bsub[Am2]'][:52],

```

```

'bs-' , label='My')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][:52],
         pyrrhotite_data_series5['Mz_bsub[Am2]'][:52],
         'gs-' , label='Mz')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][52:],
         pyrrhotite_data_series5['Mx_bsub[Am2]'][52:],
         'ro-' , label='Mx')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][52:],
         pyrrhotite_data_series5['My_bsub[Am2]'][52:],
         'bo-' , label='My')
plt.plot(pyrrhotite_data_series5['Meas_T[K]'][52:],
         pyrrhotite_data_series5['Mz_bsub[Am2]'][52:],
         'go-' , label='Mz')
plt.vlines(32, -2e-5,4.5e-5, colors='c')
plt.xlim(15,60)
plt.ylim(-2e-5,4.5e-5)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
#plt.title('Series 5 experiment (background corrected)')
plt.savefig('code_output/pyrrhotite_detail.pdf')
plt.show()

```



In the above plot, the cooling rate is significantly faster than the warming rate. As a result: (a) cooling measurements are widely spaced and (b) the sample is probably not in thermal equilibrium while cooling through the Besnus transition resulting in apparent hysteresis.

6.2 Equal area plots

The directional data for these experiments is quite interesting. The pulse magnetizations were applied in different directions, but in each case the magnetization itself was acquired in the basal plane of the crystal.

```

In [49]: series5_cart=[]
    for n in range(len(pyrrhotite_data_series5)):
        series5_cart.append([pyrrhotite_data_series5['Mx[Am2]'][n],
                            pyrrhotite_data_series5['My[Am2]'][n],
                            pyrrhotite_data_series5['Mz[Am2]'][n]])
    series5_cart_bsub=[]
    for n in range(len(pyrrhotite_data_series5)):
        series5_cart_bsub.append([pyrrhotite_data_series5['Mx_bsub[Am2]'][n],
                                pyrrhotite_data_series5['My_bsub[Am2]'][n],
                                pyrrhotite_data_series5['Mz_bsub[Am2]'][n]])

    series5_DI=pmag.cart2dir(series5_cart)
    series5_DI_bsub=pmag.cart2dir(series5_cart_bsub)

    fignum=1
    fig=plt.figure(figsize=(8,5),dpi=160)
    plt.subplot(121)
    pmagplotlib.plotNET(fignum)
    IPmag.iplotDI(series5_DI,'b')
    plt.title('series 5 (raw)')

    plt.subplot(122)
    pmagplotlib.plotNET(fignum)
    IPmag.iplotDI(series5_DI_bsub,'b')
    plt.title('series 5 (bsub)')
    plt.show()

    series6_cart=[]
    for n in range(len(pyrrhotite_data_series6)):
        series6_cart.append([pyrrhotite_data_series6['Mx[Am2]'][n],
                            pyrrhotite_data_series6['My[Am2]'][n],
                            pyrrhotite_data_series6['Mz[Am2]'][n]])
    series6_cart_bsub=[]
    for n in range(len(pyrrhotite_data_series6)):
        series6_cart_bsub.append([pyrrhotite_data_series6['Mx_bsub[Am2]'][n],
                                pyrrhotite_data_series6['My_bsub[Am2]'][n],
                                pyrrhotite_data_series6['Mz_bsub[Am2]'][n]])

    series6_DI=pmag.cart2dir(series6_cart)
    series6_DI_bsub=pmag.cart2dir(series6_cart_bsub)

    fignum=2
    fig=plt.figure(figsize=(8,5),dpi=160)
    plt.subplot(121)
    pmagplotlib.plotNET(fignum)
    IPmag.iplotDI(series6_DI,'b')
    plt.title('series 6 (raw)')

    plt.subplot(122)
    pmagplotlib.plotNET(fignum)
    IPmag.iplotDI(series6_DI_bsub,'b')
    plt.title('series 6 (bsub)')
    plt.show()

```

```

series7_cart=[]
for n in range(len(pyrrhotite_data_series7)):
    series7_cart.append([pyrrhotite_data_series7['Mx[Am2]'][n],
                         pyrrhotite_data_series7['My[Am2]'][n],
                         pyrrhotite_data_series7['Mz[Am2]'][n]])

series7_cart_bsub=[]
for n in range(len(pyrrhotite_data_series7)):
    series7_cart_bsub.append([pyrrhotite_data_series7['Mx_bsub[Am2]'][n],
                             pyrrhotite_data_series7['My_bsub[Am2]'][n],
                             pyrrhotite_data_series7['Mz_bsub[Am2]'][n]])

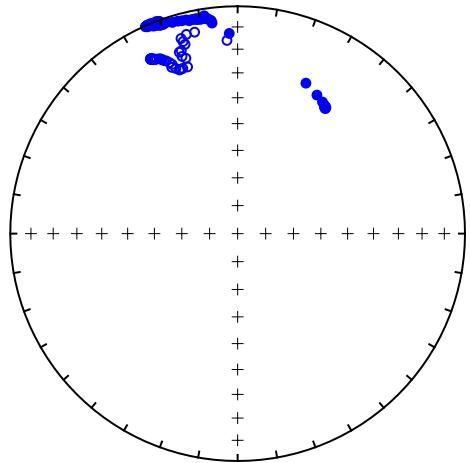
series7_DI=pmag.cart2dir(series7_cart)
series7_DI_bsub=pmag.cart2dir(series7_cart_bsub)

fignum=3
fig=plt.figure(figsize=(8,5),dpi=160)
plt.subplot(121)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(series7_DI,'b')
plt.title('series 7 (raw)')

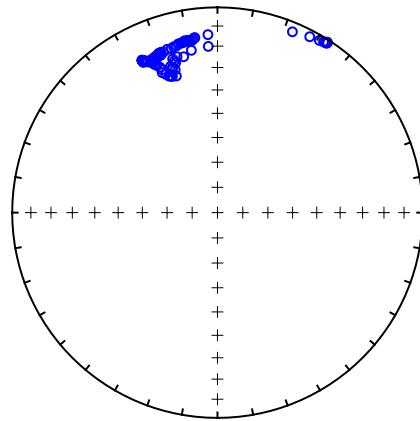
plt.subplot(122)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(series7_DI_bsub,'b')
plt.title('series 7 (bsub)')
plt.show()

```

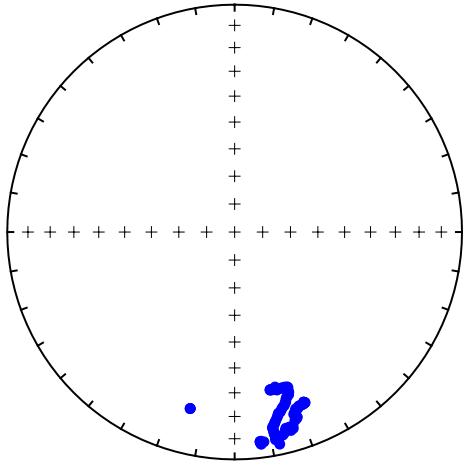
series 5 (raw)



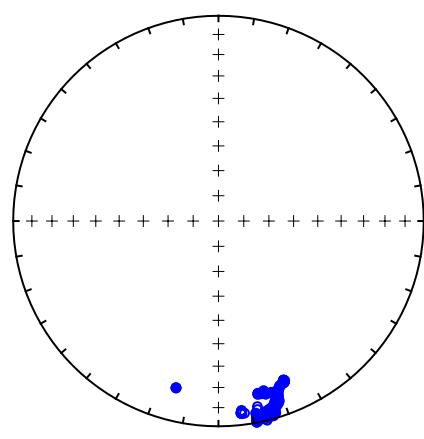
series 5 (bsub)



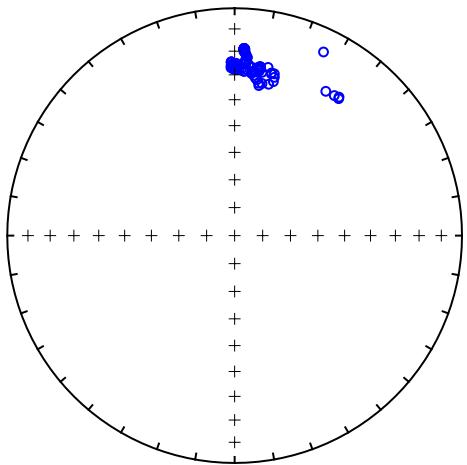
series 6 (raw)



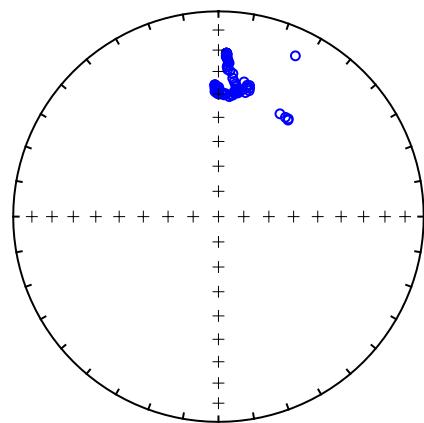
series 6 (bsub)



series 7 (raw)



series 7 (bsub)



For the figure in the manuscript, we presented the background-corrected data upon cooling for each experiment. The code below generates the equal area plots for that figure.

In [50]: `plt.figure(figsize=(12,5),dpi=160)`

```

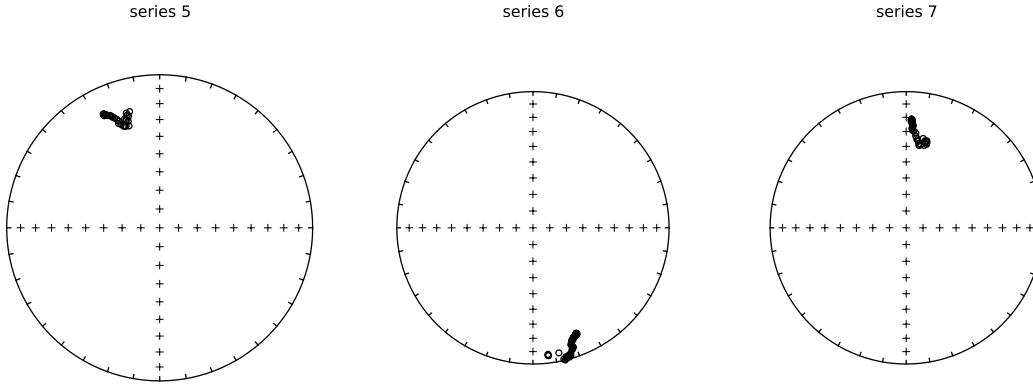
plt.subplot(131)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(series5_DI_bsub[1:45], 'k')
#just plotting the points during cooling
plt.title('series 5')

plt.subplot(132)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(series6_DI_bsub[4:56], 'k')
#just plotting the points during cooling
plt.title('series 6')

plt.subplot(133)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(series7_DI_bsub[1:50], 'k')
plt.title('series 7')

plt.savefig('code_output/pyrrhotite_equal_area.pdf')
plt.show()

```



7 Umkondo Large Igneous Province sill experiments

7.1 Blank probe runs

7.1.1 Cold Finger Blank run on July 14, 2013

In [51]: `blank1_data_cooling = pd.read_table('../Data/Umkondo/Blank/20130714_blank_cooling.txt')`
`blank1_data_warming = pd.read_table('../Data/Umkondo/Blank/20130714_blank_warming.txt')`
`blank1_data_cooling.head()`

Out[51]:

measurement_T[K]	Mx [Am2]	My [Am2]	Mz [Am2]	M_N [Am2]
0	18.34 -1.339330e-08	6.227380e-09	-6.973380e-09	-1.339330e-08
1	26.43 -2.113220e-08	2.720220e-08	-9.012280e-09	-2.113220e-08
2	32.02 -2.133780e-08	3.355980e-08	-8.730760e-09	-2.133780e-08
3	37.88 -2.063060e-08	3.554850e-08	-8.082560e-09	-2.063060e-08
4	42.43 -1.917700e-08	3.565210e-08	-7.594180e-09	-1.917700e-08

```

moment [Am2]
0 1.633370e-08
1 3.560550e-08
2 4.071600e-08
3 4.188850e-08
4 4.118860e-08

```

7.1.2 Cold Finger Blank run on July 24, 2013

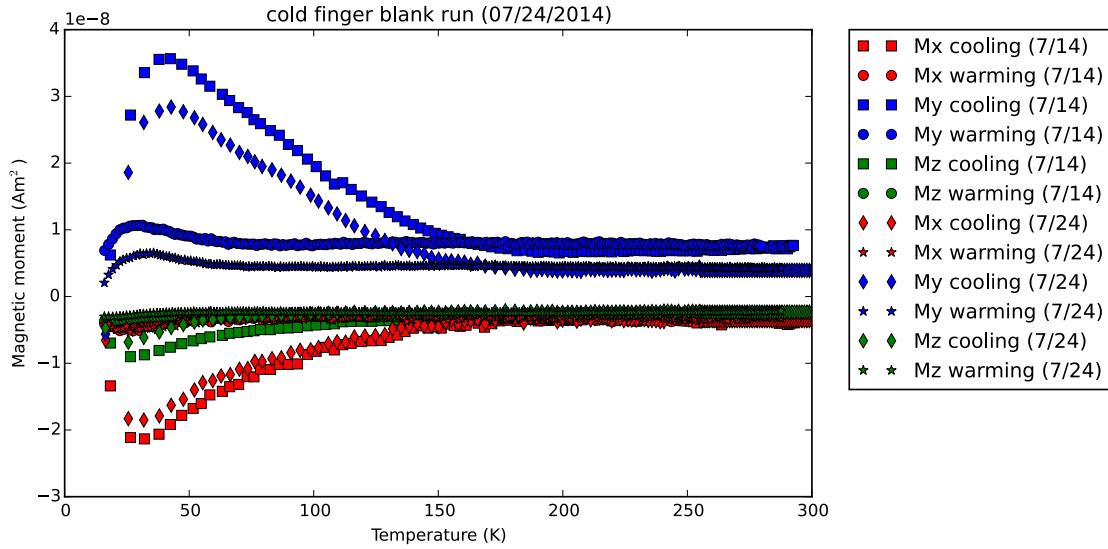
Import the blank run data into dataframes.

```
In [52]: blank2_data_cooling = pd.read_table('../Data/Umkondo/Blank/20130724_blank_cooling.txt')
blank2_data_warming = pd.read_table('../Data/Umkondo/Blank/20130724_blank_warming.txt')
blank2_data_warming.tail()
```

```
Out[52]:   measurement_T[K]      Mx [Am2]      My [Am2]      Mz [Am2]      moment [Am2]
240          284.97 -2.893900e-09  4.405760e-09 -2.202820e-09  5.712950e-09
241          285.97 -2.974400e-09  4.076580e-09 -2.199280e-09  5.504760e-09
242          286.94 -3.071310e-09  4.289740e-09 -2.199440e-09  5.715980e-09
243          287.60 -2.690500e-09  4.263650e-09 -2.193600e-09  5.498130e-09
244          287.61 -3.253950e-09  4.218480e-09 -2.131440e-09  5.738190e-09
```

```
In [53]: plt.figure(figsize=(8,5),dpi=300)
plt.plot(blank1_data_cooling['measurement_T[K]'], blank1_data_cooling['Mx [Am2]'], 'rs',
         label='Mx cooling (7/14)')
plt.plot(blank1_data_warming['measurement_T[K]'], blank1_data_warming['Mx [Am2]'], 'ro',
         label='Mx warming (7/14)')
plt.plot(blank1_data_cooling['measurement_T[K]'], blank1_data_cooling['My [Am2]'], 'bs',
         label='My cooling (7/14)')
plt.plot(blank1_data_warming['measurement_T[K]'], blank1_data_warming['My [Am2]'], 'bo',
         label='My warming (7/14)')
plt.plot(blank1_data_cooling['measurement_T[K]'], blank1_data_cooling['Mz [Am2]'], 'gs',
         label='Mz cooling (7/14)')
plt.plot(blank1_data_warming['measurement_T[K]'], blank1_data_warming['Mz [Am2]'], 'go',
         label='Mz warming (7/14)')
plt.plot(blank2_data_cooling['measurement_T[K]'], blank2_data_cooling['Mx [Am2]'], 'rd',
         label='Mx cooling (7/24)')
plt.plot(blank2_data_warming['measurement_T[K]'], blank2_data_warming['Mx [Am2]'], 'r*',
         label='Mx warming (7/24)')
plt.plot(blank2_data_cooling['measurement_T[K]'], blank2_data_cooling['My [Am2]'], 'bd',
         label='My cooling (7/24)')
plt.plot(blank2_data_warming['measurement_T[K]'], blank2_data_warming['My [Am2]'], 'b*',
         label='My warming (7/24)')
plt.plot(blank2_data_cooling['measurement_T[K]'], blank2_data_cooling['Mz [Am2]'], 'gd',
         label='Mz cooling (7/24)')
plt.plot(blank2_data_warming['measurement_T[K]'], blank2_data_warming['Mz [Am2]'], 'g*',
         label='Mz warming (7/24)')

plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('cold finger blank run (07/24/2014)')
plt.show()
```



The origin of this signal from the sample holder apparatus is currently unknown and will be investigated further in the future. The signal seems to remain broadly similar through repeated thermal cycling (compare the July 14, 2013 data to the July 24, 2013 data) so we will take the approach of subtracting the holder signal from the sample data. To do this holder subtraction, it is necessary to interpolate the holder signal at the temperatures for which sample data are collected. Here is an example of the spline interpolation of these data that we will use for this purpose. This spline interpolation is the one that is used in the `background_sub()` function defined above.

```
In [54]: x = blank1_data_warming['measurement_T[K]']
y = blank1_data_warming['My[Am2]']
spline = interpolate.splrep(x,y,s=0)
xnew = np.linspace(blank1_data_warming['measurement_T[K]'].min(),
                   blank1_data_warming['measurement_T[K]'].max()-1, 300)
ynew = interpolate.splev(xnew,spline,der=0)

plt.figure(num=fignum,figsize=(8,5),dpi=300)
plt.plot(x, y, 'bo',
          label='My warming')
plt.plot(xnew,ynew, 'r-')

#For the interpolate function to work the x values (temperature) need to be ascending.
#The blank1_data_cooling can be sorted so that temperatures are ascending
blank1_data_cooling = blank1_data_cooling.sort(['measurement_T[K]'])

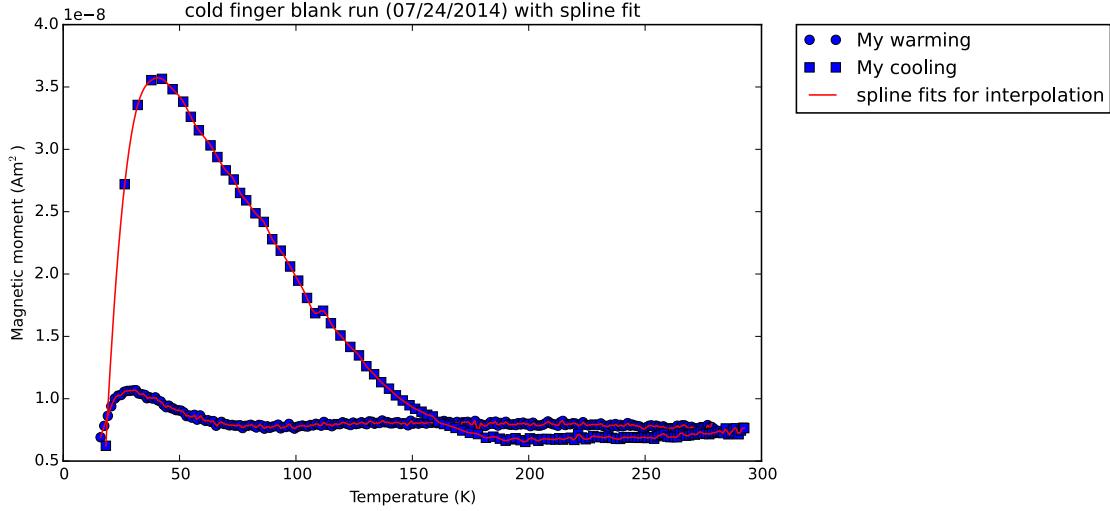
x = blank1_data_cooling['measurement_T[K]']
y = blank1_data_cooling['My[Am2]']
tck = interpolate.splrep(x,y,s=0)
xnew = np.linspace(blank1_data_cooling['measurement_T[K]'].min(),
                   blank1_data_cooling['measurement_T[K]'].max(), 300)
ynew = interpolate.splev(xnew,tck,der=0)

plt.plot(x, y, 'bs',
          label='My cooling')
plt.plot(xnew,ynew, 'r-',label='spline fits for interpolation')
```

```

plt.ylabel('Magnetic moment (Am$^2$)')
plt.xlabel('Temperature (K)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('cold finger blank run (07/24/2014) with spline fit')
plt.show()

```



7.2 Experiments on PW10-7d

PW10 is a sill we refer to as the Rapitsane Sill from within the Umkondo Large Igneous Province in Botswana. Location: S 24.41968°, E 025.58463°. MPMS experiments and liquid nitrogen demagnetization revealed that this specimen lost a high percentage of its remanence upon cycling to and from low-temperature in low/near-zero field. The remanence removed during this low-temperature cycling is clearly an overprint on top of a primary

7.2.1 NRM demagnetization

```
In [55]: PW10_7a_demag = pd.read_csv('../Data/Umkondo/PW10/PW10-7a.csv')
PW10_7a_demag
```

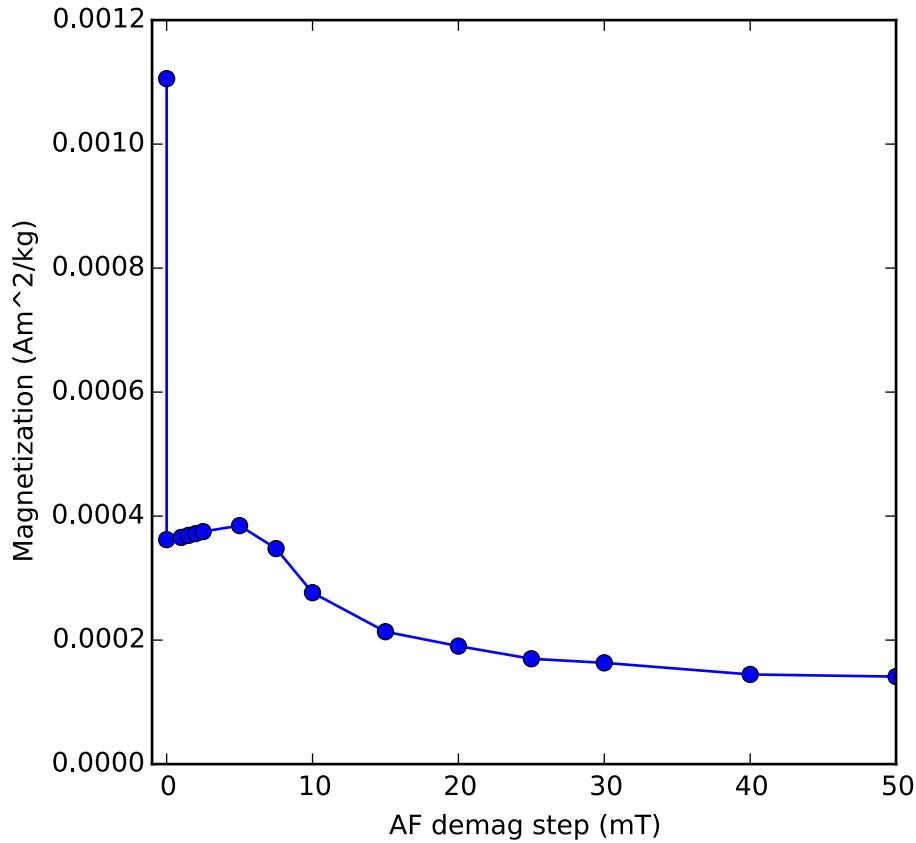
```
Out[55]:   specimen  step  intensity  dec_geo  inc_geo  dec_spec  inc_spec
0    PW10-7a  0.0  0.001106  287.293  58.389400  131.3    49.4
1    PW10-7a  0.0  0.000362  205.501  25.299800  331.1    65.1
2    PW10-7a  1.0  0.000366  204.858  24.588300  330.4    64.2
3    PW10-7a  1.5  0.000369  204.310  23.822200  330.0    63.3
4    PW10-7a  2.0  0.000372  203.611  22.952500  329.4    62.3
5    PW10-7a  2.5  0.000375  202.734  21.649100  328.9    60.8
6    PW10-7a  5.0  0.000385  197.607  13.624400  326.2    51.5
7    PW10-7a  7.5  0.000348  192.869   6.610280  323.9    43.2
8    PW10-7a 10.0  0.000276  186.925   0.377430  320.2    35.1
9    PW10-7a 15.0  0.000214  184.698  -2.601010  319.3    31.4
10   PW10-7a 20.0  0.000190  187.394  -1.962070  321.9    33.2
11   PW10-7a 25.0  0.000170  189.228  -1.561390  323.7    34.3
12   PW10-7a 30.0  0.000163  189.053  -1.088220  323.3    34.6
13   PW10-7a 40.0  0.000145  186.617   4.399750  317.7    38.6
14   PW10-7a 50.0  0.000141  184.853   0.728249  317.8    34.5
```

```
In [56]: PW10_7a_datablock_geo=[]
PW10_7a_datablock_spec=[]
PW10_7a_DI_geo=[]
PW10_7a_DI_spec=[]
for n in range(0,len(PW10_7a_demag)):
    PW10_7a_datablock_geo.append([PW10_7a_demag['step'][n],
                                PW10_7a_demag['dec_geo'][n],
                                PW10_7a_demag['inc_geo'][n],
                                PW10_7a_demag['intensity'][n]])
    PW10_7a_datablock_spec.append([PW10_7a_demag['step'][n],
                                    PW10_7a_demag['dec_spec'][n],
                                    PW10_7a_demag['inc_spec'][n],
                                    PW10_7a_demag['intensity'][n]])
    PW10_7a_DI_geo.append([PW10_7a_demag['dec_geo'][n],
                           PW10_7a_demag['inc_geo'][n]])
    PW10_7a_DI_spec.append([PW10_7a_demag['dec_spec'][n],
                            PW10_7a_demag['inc_spec'][n]])
```

Intensity plot

```
In [57]: plt.figure(figsize=(5,5))
plt.plot(PW10_7a_demag['step'],PW10_7a_demag['intensity'],marker='o')
plt.xlabel('AF demag step (mT)')
plt.ylabel('Magnetization (Am^2/kg)')
plt.xlim((-1,50))
plt.show()

print 'The remanence lost (in %) following low-temp cycling is:'
print ((PW10_7a_demag['intensity'][0]-PW10_7a_demag['intensity'][1])/PW10_7a_demag['intensity'][0])*100
```



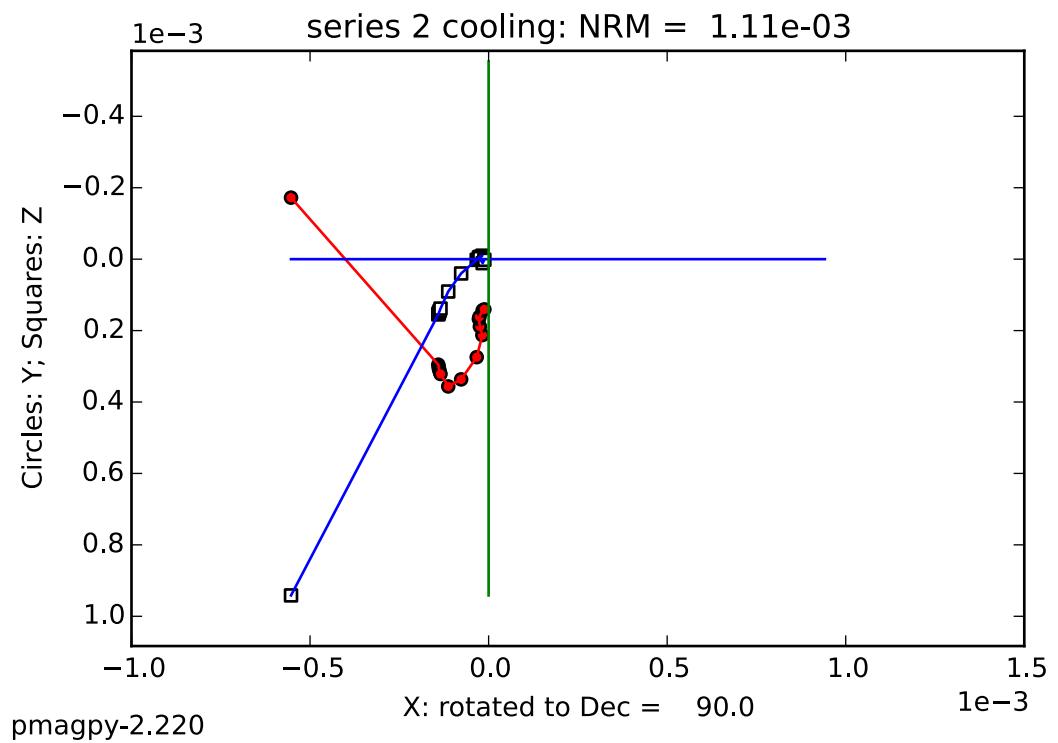
The remanence lost (in %) following low-temp cycling is:
67.2532472231

The large loss of remanence is associated with the 77K nitrogen bath.

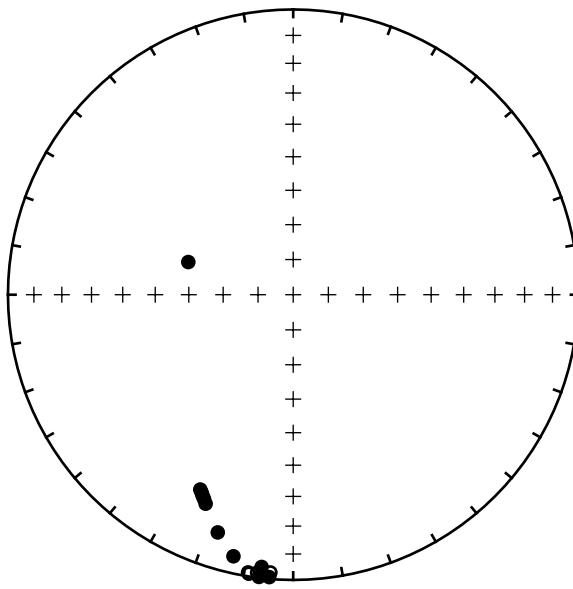
Plots in geographic coordinates

```
In [58]: #plot with pmagplotlib.plotZ(fignum,datablock,angle,s,norm)
pmagplotlib.plotZ(1,PW10_7a_datablock_geo,90,'series 2 cooling',0)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
plt.show()

plt.figure(figsize=(5,5))
pmagplotlib.plotNET(1)
IPmag.iplotDI(PW10_7a_DI_geo)
plt.title('Equal Area of demag')
plt.show()
```



Equal Area of demag



In the plots above, the large directional change and reduction in magnetization accompanies a low-temperature demagnetization step (i.e. 77 K nitrogen bath in a very low-field environment). Along with the following AF steps an overprint is removed that brings the data to the shallow southerly pointing vector which is consistent with other data from the ~1100 Ma Umkondo large igneous province.

```
In [59]: print "The angular change (in degrees) associated with the LTD step is:"  
        print pmag.angle(PW10_7a_DI_geo[0],PW10_7a_DI_geo[1])
```

```
The angular change (in degrees) associated with the LTD step is:  
[ 64.43078671]
```

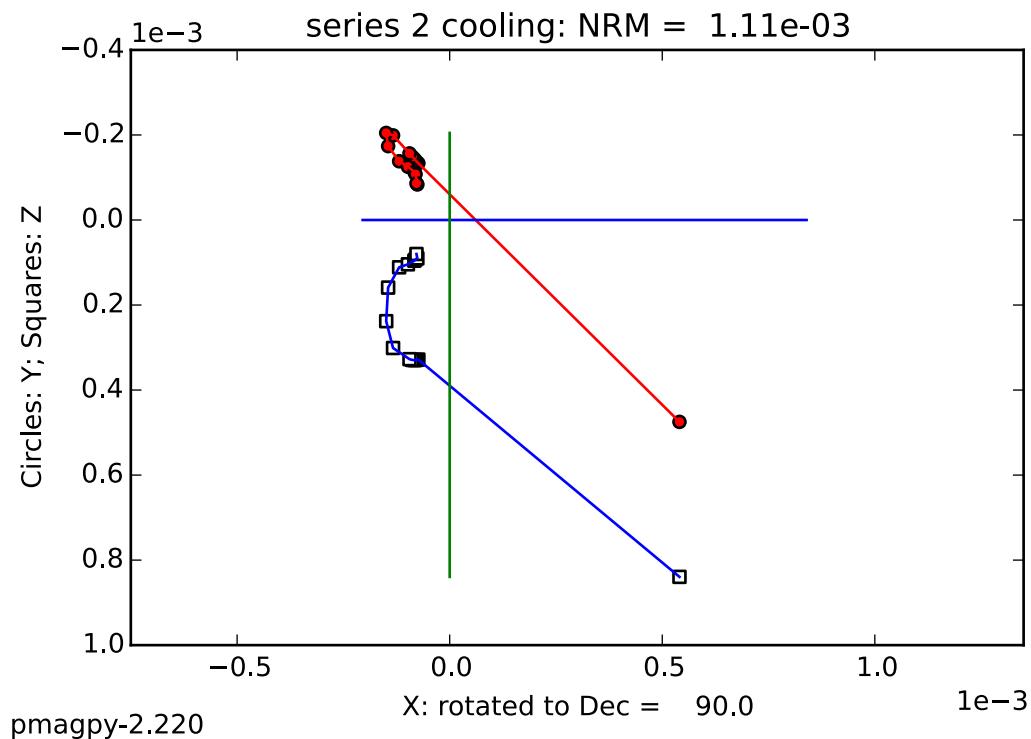
Specimen coordinates These data plotted in specimen coordinates are more directly comparable to the data acquired on another specimen from the same sample using the IRM-LTI.

```
In [60]: #plot with pmagplotlib.plotZ(fignum,datablock,angle,s,norm)  
        pmagplotlib.plotZ(1,PW10_7a_datblock_spec,90,'series 2 cooling',0)  
        plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')  
        plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')  
        plt.savefig('code_output/Umk_Pw10-7a_specimen_zplot.pdf')  
        plt.show()
```

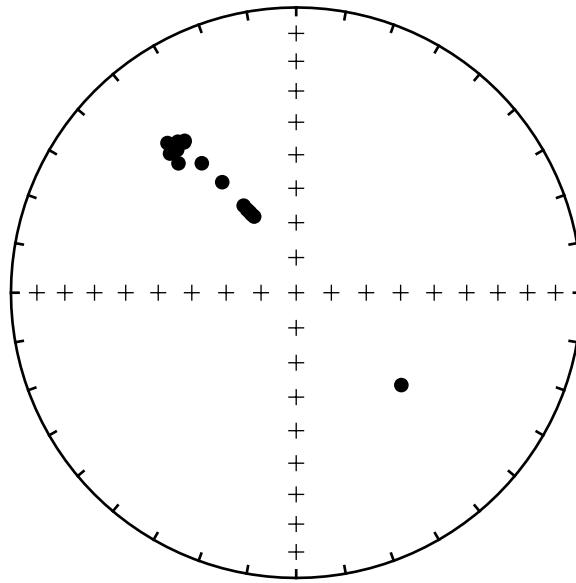
```

plt.figure(figsize=(5,5))
pmagplotlib.plotNET(1)
IPmag.iplotDI(PW10_7a_DI_spec)
plt.title('Equal Area of demag')
plt.savefig('code_output/Umk_Pw10-7a_specimen_eqaplot.pdf')
plt.show()

```



Equal Area of demag



7.2.2 MPMS experiments

```
In [61]: PW10_MPMS=pd.read_csv('..../Data/Umkondo/PW10/PW10_7c_MPMS.csv',header=2)
pd.set_option('display.max_columns', None)
PW10_MPMS.head(5)
```

```
Out[61]:      T [K]  Bapp [T]  M [Am2/kg]    reg fit          timestamp  Unnamed: 5 \
0  299.9879      2.5  2.885055  0.990173  2/5/2013 4:13:32 PM      NaN
1  294.2503      2.5  2.899802  0.989840  2/5/2013 4:16:42 PM      NaN
2  288.9702      2.5  2.914910  0.989691  2/5/2013 4:17:28 PM      NaN
3  283.9853      2.5  2.931989  0.989543  2/5/2013 4:18:06 PM      NaN
4  278.9876      2.5  2.951009  0.989169  2/5/2013 4:18:40 PM      NaN

      T [K].1  Bapp [T].1  M [Am2/kg].1  reg fit.1          timestamp.1 \
0  19.99881        0     0.457038   0.990071  2/5/2013 5:27:50 PM
1  25.59370        0     0.434746   0.989292  2/5/2013 5:29:09 PM
2  30.62943        0     0.415489   0.989056  2/5/2013 5:30:10 PM
3  35.51034        0     0.401149   0.988457  2/5/2013 5:31:10 PM
4  40.58522        0     0.389925   0.988874  2/5/2013 5:32:11 PM
```

```

    Unnamed: 11      T [K].2      Bapp [T].2      M [Am2/kg].2      reg fit.2  \
0           NaN   19.99765          0     0.483920   0.988106
1           NaN   25.55709          0     0.465696   0.987919
2           NaN   30.55221          0     0.451775   0.988061
3           NaN   35.59132          0     0.442285   0.987936
4           NaN   40.54948          0     0.434388   0.988107

      timestamp.2      Unnamed: 17      T [K].3      Bapp [T].3      M [Am2/kg].3  \
0  2/5/2013 7:43:34 PM      NaN  300.0001          0     0.292515
1  2/5/2013 7:44:54 PM      NaN  294.4174          0     0.292170
2  2/5/2013 7:45:55 PM      NaN  289.0140          0     0.291212
3  2/5/2013 7:46:57 PM      NaN  283.8555          0     0.290107
4  2/5/2013 7:47:56 PM      NaN  279.0740          0     0.288958

      reg fit.3      timestamp.3      Unnamed: 23      T [K].4      Bapp [T].4  \
0  0.989795  2/5/2013 8:53:56 PM      NaN  19.99720          0
1  0.989635  2/5/2013 8:56:51 PM      NaN  25.65808          0
2  0.989510  2/5/2013 8:57:42 PM      NaN  30.53962          0
3  0.989398  2/5/2013 8:58:22 PM      NaN  35.66172          0
4  0.989000  2/5/2013 8:58:58 PM      NaN  40.49764          0

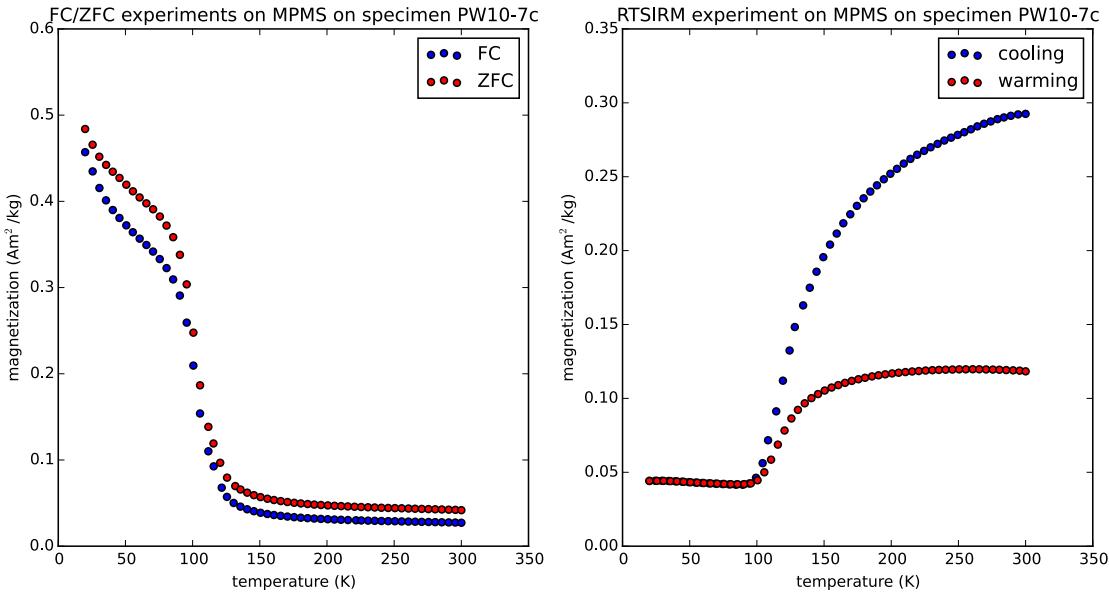
      M [Am2/kg].4      reg fit.4      timestamp.4
0      0.044147      0.987200  2/5/2013 10:03:51 PM
1      0.044205      0.987814  2/5/2013 10:05:12 PM
2      0.044195      0.987583  2/5/2013 10:06:11 PM
3      0.044043      0.987072  2/5/2013 10:07:14 PM
4      0.043838      0.987979  2/5/2013 10:08:12 PM

```

```

In [62]: fig=plt.figure(figsize=(12,6),dpi=160)
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)
ax1.scatter(PW10_MPMS['T [K].1'],PW10_MPMS['M [Am2/kg].1'], c='b', label='FC')
ax1.scatter(PW10_MPMS['T [K].2'],PW10_MPMS['M [Am2/kg].2'], c='r', label='ZFC')
ax2.scatter(PW10_MPMS['T [K].3'],PW10_MPMS['M [Am2/kg].3'], c='b', label='cooling')
ax2.scatter(PW10_MPMS['T [K].4'],PW10_MPMS['M [Am2/kg].4'], c='r', label='warming')
ax1.set_xlabel('temperature (K)')
ax1.set_ylabel('magnetization (Am$^2$/kg)')
ax1.legend()
ax1.set_title('FC/ZFC experiments on MPMS on specimen PW10-7c')
ax2.set_xlabel('temperature (K)')
ax2.set_ylabel('magnetization (Am$^2$/kg)')
ax2.legend()
ax2.set_title('RTSIRM experiment on MPMS on specimen PW10-7c')
plt.savefig('code_output/PW10-7c_MPMS.svg')
plt.show()

```



7.2.3 IRM-LTI experiments

Experiment on 07/21/2013 on specimen PW10-7d (measurement series 1 and 2): This experiment was designed to be low-temperature cycling of the specimens natural remanent magnetization (NRM).

Load the sample with arrow (down core z-axis) pointing out of SRM (old 2G at IRM) and make measurement (measurement series #1, measurement #1,2,3,4). The direction should be approx. the same as the measurement of the sister “a-series” specimen.

Twist the sample screw tight as is needed for running in low-temperature insert which results in sample being rotated about the z-axis (measurement series #1, measurement #5).

Cover the sample holder with the sapphire radiation shield and vacuum shroud ((measurement series #1, measurement #5) and then begin cooling. Measurements are made upon cooling (measurement series #2).

Once at minimum temperature (slightly below 20K) warm the probe back up progressively making measurements as it warms (measurement series #2).

Import data file which has the format:

```
TEMPERATURE (K)
M_x [Am²]
M_y [Am²]
M_z [Am²]
M_total [Am²]
MEASUREMENT SERIES
```

```
In [63]: data_file='..../Data/Umkondo/PW10/PW10-7d_rotation.txt'
data = np.genfromtxt(data_file, skip_header=1)
```

```
temp=data[:,0]
Mx=data[:,1]
My=data[:,2]
Mz=data[:,3]
Mtotal=data[:,4]
```

```
In [64]: full_vector_oriented=[]
full_vector_rotated=[]
```

```

#data point 3 is oriented (remember python starts at 0)
for n in range(3,4):
    Mx_full,My_full,Mz_full=Mx[n],My[n],Mz[n]
    full_vector_oriented.append([Mx_full,My_full,Mz_full])

#data point 4 is rotated about z (remember python starts at 0)
for n in range(4,5):
    Mx_full,My_full,Mz_full=Mx[n],My[n],Mz[n]
    full_vector_rotated.append([Mx_full,My_full,Mz_full])

oriented_directions=pmag.cart2dir(full_vector_oriented)
rotated_directions=pmag.cart2dir(full_vector_rotated)

#rotate rotated directions about the z-axis to correspond to the oriented measurement
angle = 360 + oriented_directions.item(0) - rotated_directions.item(0)

x_prime = Mx[4]*np.cos(np.radians(angle)) - My[4]*np.sin(np.radians(angle))
y_prime = Mx[4]*np.sin(np.radians(angle)) + My[4]*np.cos(np.radians(angle))
z_prime = Mz[4]

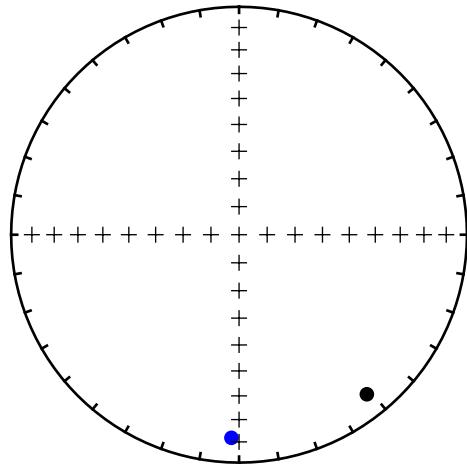
full_vector_rotated_prime= [[x_prime,y_prime,z_prime]]
rotated_directions_prime=pmag.cart2dir(full_vector_rotated_prime)

#plot the oriented measurement and the rotated measurement
fignum = 1
plt.figure(num=fignum, figsize=(4,4))
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(oriented_directions, 'k')
IPmag.iplotDI(rotated_directions, 'b')
plt.title('PW10-7d NRM oriented (black) and rotated (blue);');

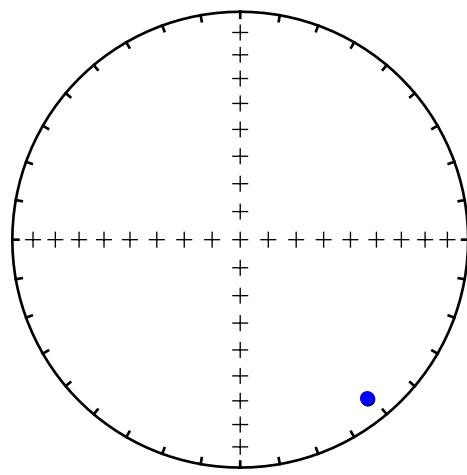
#plot the oriented measurement and the corrected rotated measurement
fignum = 2
plt.figure(num=fignum, figsize=(4,4))
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(oriented_directions, 'k')
IPmag.iplotDI(rotated_directions_prime, 'b')
plt.title('PW10-7d NRM oriented (black) and rotated (corrected; blue);');

```

PW10-7d NRM oriented (black) and rotated (blue)



PW10-7d NRM oriented (black) and rotated (corrected; blue)



```

In [65]: data_file_cooling='..../Data/Umkondo/PW10/PW10-7d_cooling.txt'
          data_cooling = np.genfromtxt(data_file_cooling, skip_header=1)

          temp_cooling=data_cooling[:,0]
          Mx_cooling=data_cooling[:,1]
          My_cooling=data_cooling[:,2]
          Mz_cooling=data_cooling[:,3]
          Mtotal_cooling=data_cooling[:,4]

          data_file_warming='..../Data/Umkondo/PW10/PW10-7d_warming.txt'
          data_warming = np.genfromtxt(data_file_warming, skip_header=1)

          temp_warming=data_warming[:,0]
          Mx_warming=data_warming[:,1]
          My_warming=data_warming[:,2]
          Mz_warming=data_warming[:,3]
          Mtotal_warming=data_warming[:,4]

In [66]: Mx_cooling_bsub = background_sub(blank1_data_cooling['Mx[Am2]'],
                                         blank1_data_cooling['measurement_T[K]'],
                                         Mx_cooling,temp_cooling)

          My_cooling_bsub = background_sub(blank1_data_cooling['My[Am2]'],
                                         blank1_data_cooling['measurement_T[K]'],
                                         My_cooling,temp_cooling)

          Mz_cooling_bsub = background_sub(blank1_data_cooling['Mz[Am2]'],
                                         blank1_data_cooling['measurement_T[K]'],
                                         Mz_cooling,temp_cooling)

          Mx_warming_bsub = background_sub(blank1_data_warming['Mx[Am2]'],
                                         blank1_data_warming['measurement_T[K]'],
                                         Mx_warming[:-8],temp_warming[:-8])

          My_warming_bsub = background_sub(blank1_data_warming['My[Am2]'],
                                         blank1_data_warming['measurement_T[K]'],
                                         My_warming[:-8],temp_warming[:-8])

          Mz_warming_bsub = background_sub(blank1_data_warming['Mz[Am2]'],
                                         blank1_data_warming['measurement_T[K]'],
                                         Mz_warming[:-8],temp_warming[:-8])

          fignum=1
          plt.figure(num=fignum,figsize=(10,10),dpi=160)

          ax=plt.subplot(211)
          plt.plot(temp_cooling, Mx_cooling, 'r*-',label="Mx cooling")
          plt.plot(temp_cooling, My_cooling, 'b*-',label="My cooling")
          plt.plot(temp_cooling, Mz_cooling, 'g*-',label="Mz cooling")
          plt.plot(temp_warming, Mx_warming, 'rv-',label="Mx warming")
          plt.plot(temp_warming, My_warming, 'bv-',label="My warming")
          plt.plot(temp_warming, Mz_warming, 'gv-',label="Mz warming")
          plt.xlabel('Temperature (K)')
          plt.ylabel('Magnetic moment (Am$^2$)')

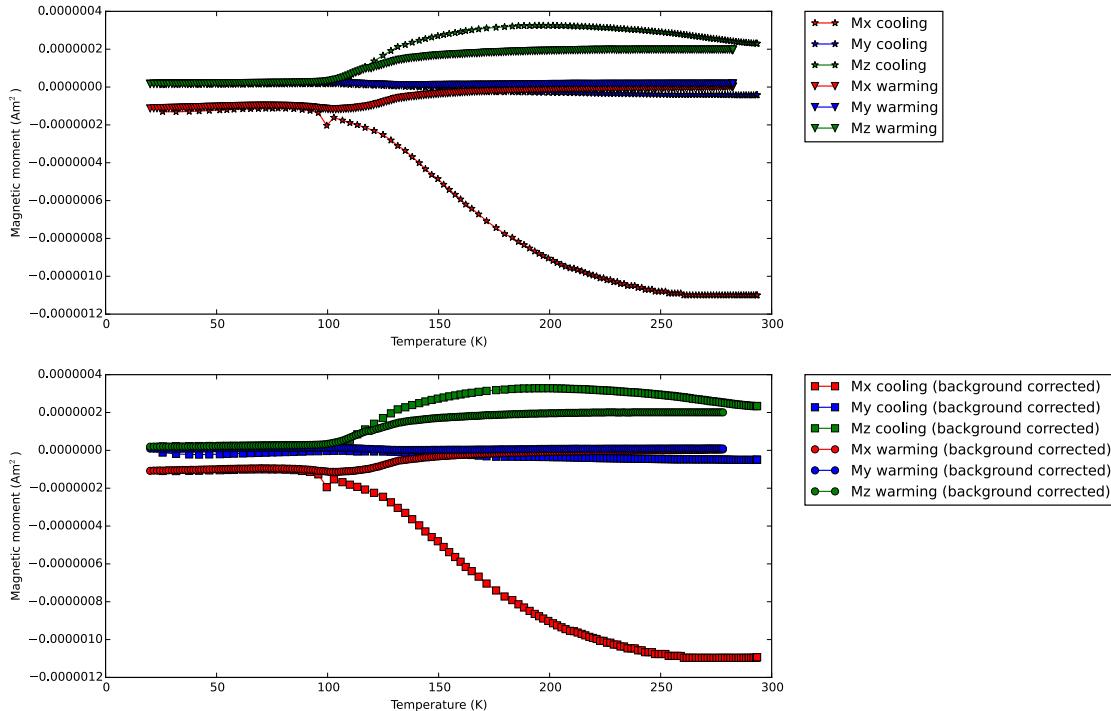
```

```

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

ax=plt.subplot(212)
plt.plot(temp_cooling, Mx_cooling_bsub, 'rs-',
          label="Mx cooling (background corrected)")
plt.plot(temp_cooling, My_cooling_bsub, 'bs-',
          label="My cooling (background corrected)")
plt.plot(temp_cooling, Mz_cooling_bsub, 'gs-',
          label="Mz cooling (background corrected)")
plt.plot(temp_warming[:-8], Mx_warming_bsub, 'ro-',
          label="Mx warming (background corrected)")
plt.plot(temp_warming[:-8], My_warming_bsub, 'bo-',
          label="My warming (background corrected)")
plt.plot(temp_warming[:-8], Mz_warming_bsub, 'go-',
          label="Mz warming (background corrected)")
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()

```



```

In [67]: full_vector_cooling=[]
    for n in range(0,len(temp_cooling)):
        full_vector_cooling.append([Mx_cooling[n],My_cooling[n],Mz_cooling[n]])
full_vector_warming=[]
    for n in range(0,len(temp_warming)):
        full_vector_warming.append([Mx_warming[n],My_warming[n],Mz_warming[n]])

cooling_directions=pmag.cart2dir(full_vector_cooling)

```

```

warming_directions=pmag.cart2dir(full_vector_warming)

fignum=3
plt.figure(num=fignum,figsize=(4,4),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(cooling_directions,'b')
IPmag.iplotDI(warming_directions,'r')
plt.title('LT-cycling of NRM (rotated; no background subtraction)')
#plt.savefig('PW10-7d_3axisLTD_eqarea.svg')
plt.show()

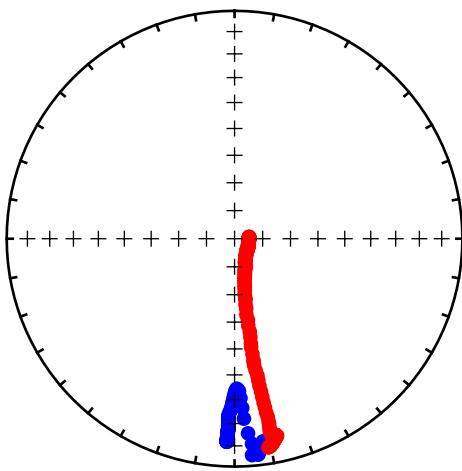
full_vector_cooling=[]
for n in range(0,len(temp_cooling)):
    full_vector_cooling.append([Mx_cooling_bsub[n],My_cooling_bsub[n],Mz_cooling_bsub[n]])
full_vector_warming=[]
for n in range(0,len(Mx_warming_bsub)):
    full_vector_warming.append([Mx_warming_bsub[n],My_warming_bsub[n],Mz_warming_bsub[n]])

cooling_directions=pmag.cart2dir(full_vector_cooling)
warming_directions=pmag.cart2dir(full_vector_warming)

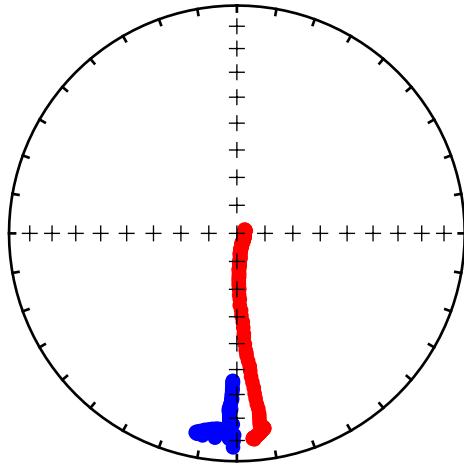
fignum=3
plt.figure(num=fignum,figsize=(4,4),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(cooling_directions,'b')
IPmag.iplotDI(warming_directions,'r')
plt.title('LT-cycling of NRM (rotated; background subtraction)')
#plt.savefig('PW10-7d_3axisLTD_eqarea.svg')
plt.show()

```

LT-cycling of NRM (rotated; no background subtraction)



LT-cycling of NRM (rotated; background subtraction)



These plots show that the background subtraction is imperfect along the y-axis and is thereby adding an artifact to the data that is making it so that the end cooling values are not matching the start warming values. As can be seen in the data for the empty probe above, the y-axis has the highest magnetization in the blank probe runs with significant temperature dependent change. Given that the majority of the specimens directional change is in the z-axis and the z-axis data is minimally effected by the background subtraction, let's proceed with plotting the data that is not corrected for the holder, but is rotated back into specimen coordinates (e.g. Mz_cooling rather than Mz_cooling_bsub).

```
In [68]: Mx_cooling_prime = Mx_cooling*np.cos(np.radians(angle)) - My_cooling*np.sin(np.radians(angle))
My_cooling_prime = Mx_cooling*np.sin(np.radians(angle)) + My_cooling*np.cos(np.radians(angle))
Mz_cooling_prime = Mz_cooling
full_vector_cooling=[]
for n in range(0,len(temp_cooling)):
    full_vector_cooling.append([Mx_cooling_prime[n],My_cooling_prime[n],Mz_cooling_prime[n]])

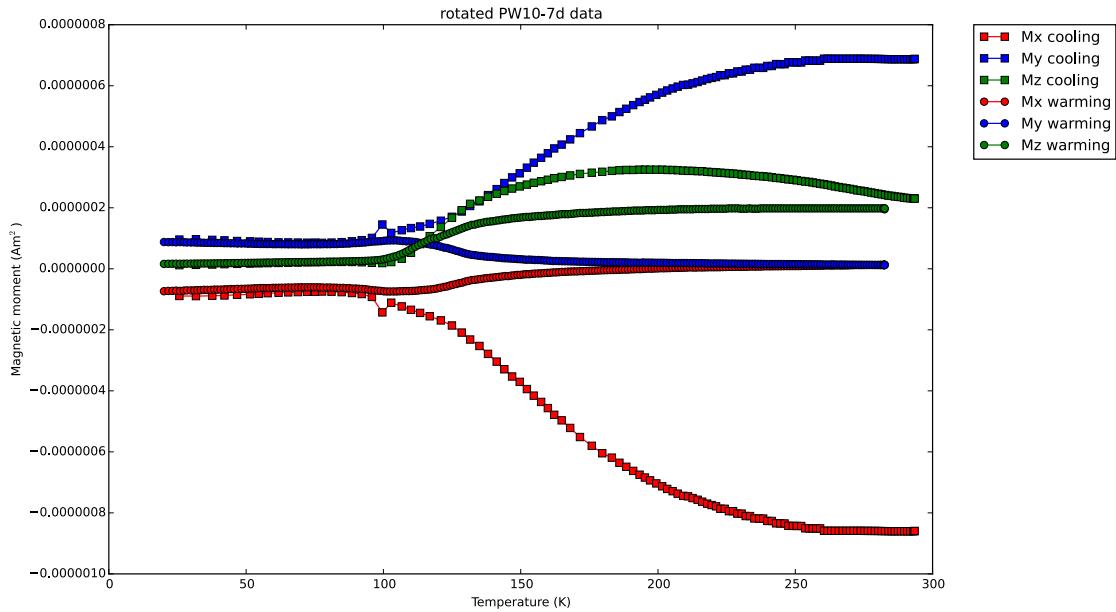
Mx_warming_prime = Mx_warming*np.cos(np.radians(angle)) - My_warming*np.sin(np.radians(angle))
My_warming_prime = Mx_warming*np.sin(np.radians(angle)) + My_warming*np.cos(np.radians(angle))
Mz_warming_prime = Mz_warming
full_vector_warming=[]
for n in range(0,len(temp_warming)):
    full_vector_warming.append([Mx_warming_prime[n],My_warming_prime[n],Mz_warming_prime[n]])

fignum=2
plt.figure(num=fignum,figsize=(12,8),dpi=160)
```

```

plt.plot(temp_cooling, Mx_cooling_prime, 'rs-',label="Mx cooling")
plt.plot(temp_cooling, My_cooling_prime, 'bs-',label="My cooling")
plt.plot(temp_cooling, Mz_cooling_prime, 'gs-',label="Mz cooling")
plt.plot(temp_warming, Mx_warming_prime, 'ro-',label="Mx warming")
plt.plot(temp_warming, My_warming_prime, 'bo-',label="My warming")
plt.plot(temp_warming, Mz_warming_prime, 'go-',label="Mz warming")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.title('rotated PW10-7d data')
plt.savefig('code_output/PW10-7d_3axisLTD.svg')

```



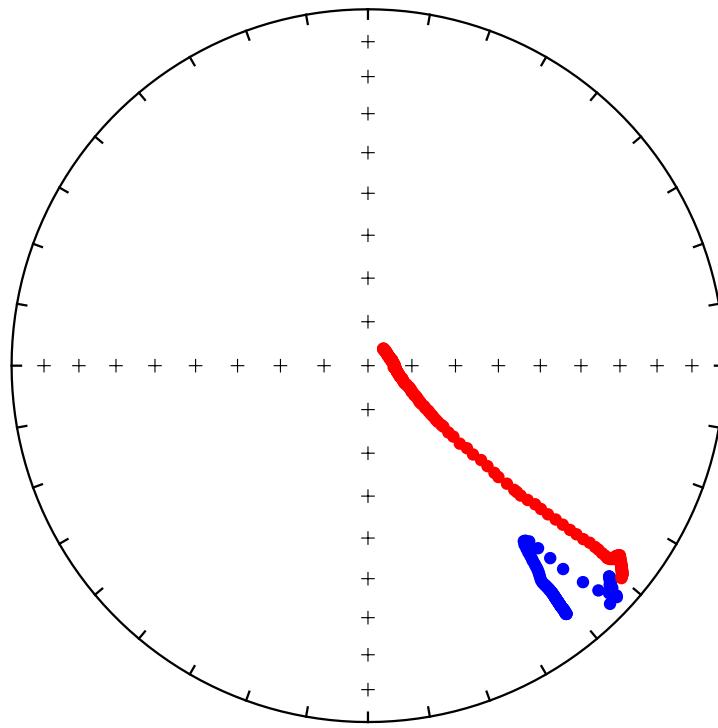
```

In [69]: cooling_directions=pmag.cart2dir(full_vector_cooling)
warming_directions=pmag.cart2dir(full_vector_warming)

figure=3
plt.figure(num=figure,figsize=(7,7),dpi=160)
pmagplotlib.plotNET(figure)
IPmag.iplotDI(cooling_directions,'b')
IPmag.iplotDI(warming_directions,'r')
plt.title('PW10-7d NRM upon cooling and warming (specimen coordinates)')
plt.savefig('code_output/PW10-7d_3axisLTD_eqarea.svg')
plt.show()

```

PW10-7d NRM upon cooling and warming (specimen coordinates)



```
In [70]: print "The angular change from NRM to the end of LT-cycling is:"  
print pmag.angle(cooling_directions[0],warming_directions[240])
```

The angular change from NRM to the end of LT-cycling is:
[84.7946601]

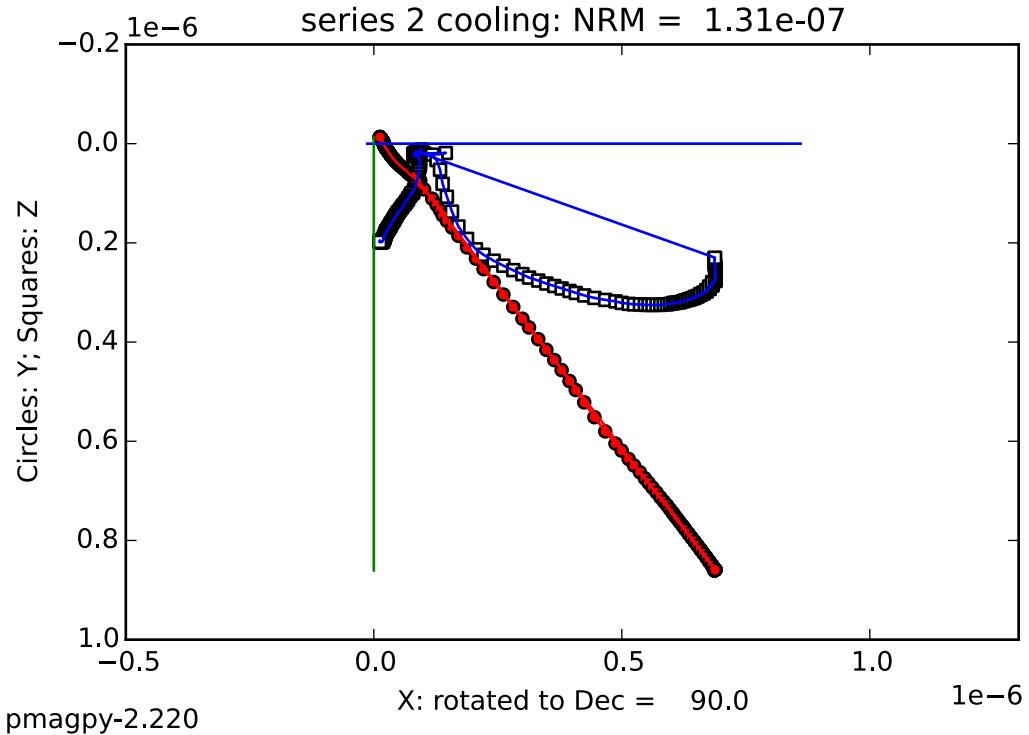
```
In [71]: PW10_7d_datablock=[]  
  
for n in range(0,len(cooling_directions)):  
    PW10_7d_datablock.append([temp_cooling[n],  
                            cooling_directions[n][0],  
                            cooling_directions[n][1],  
                            cooling_directions[n][2]])  
  
for n in range(0,len(warming_directions)):  
    PW10_7d_datablock.append([temp_warming[n],
```

```

warming_directions[n][0],
warming_directions[n][1],
warming_directions[n][2]]))

#plot with pmagplotlib.plotZ(fignum, datablock, angle, s, norm)
pmagplotlib.plotZ(1,PW10_7d_datablock,90,'series 2 cooling',0)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
plt.savefig('code_output/Umk_Pw10-7a_specimen_zplot.pdf')
plt.show()

```



7.3 PW15-4

This sample is from the interior (~ 11 meters from margin) of an Umkondo Large Igneous Province diabase sill (the Mogatelwane 2 sill) and displayed a large loss in remanence associated with a liquid nitrogen treatment step that was applied during initial demagnetization of the NRM. Due to this behavior the sample was targeted for experiments on the low-temperature probe.

7.3.1 NRM demagnetization

```
In [72]: PW15_4a_demag = pd.read_csv('../Data/Umkondo/PW15/PW15-4a.csv')
PW15_4a_demag
```

```
Out[72]:   specimen  step(mT)  intensity  dec_geo  inc_geo  dec_spec  inc_spec
0      PW15-4        0    0.034587   200.2     -7.8    354.3     72.1
1      PW15-4        0    0.016562   200.1     -9.1    354.2     70.8
2      PW15-4        5    0.015866   200.1     -8.8    354.3     71.1
```

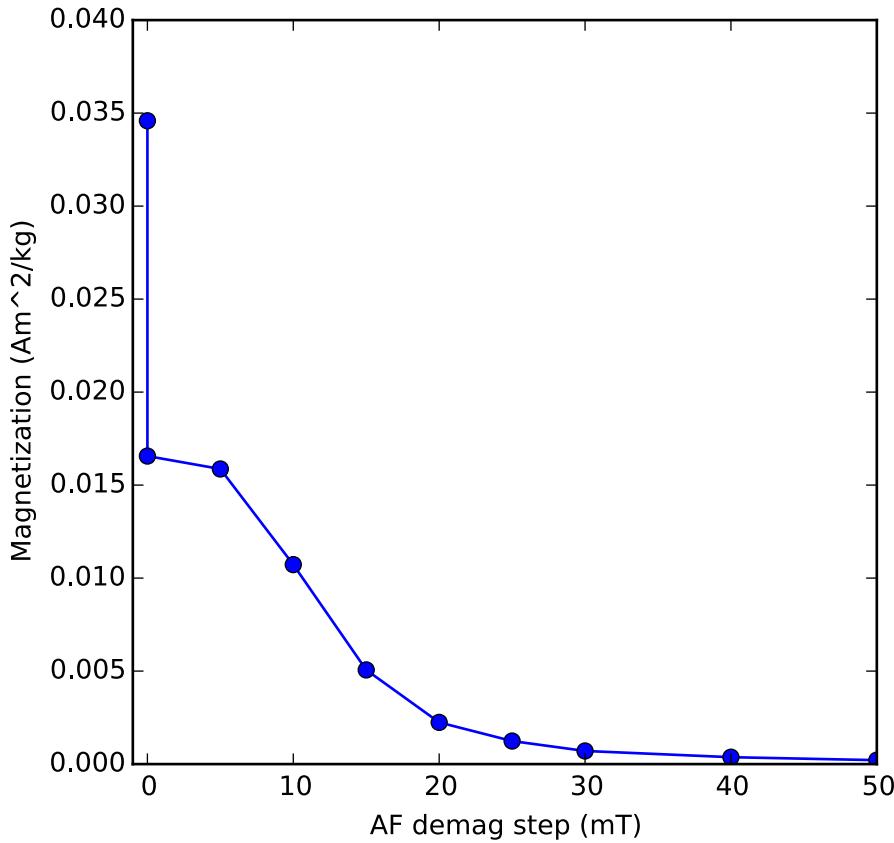
3	PW15-4	10	0.010725	199.8	-8.1	352.9	71.7
4	PW15-4	15	0.005063	198.5	-9.9	349.8	69.8
5	PW15-4	20	0.002243	196.5	-12.2	346.0	67.2
6	PW15-4	25	0.001239	194.6	-11.6	341.1	67.2
7	PW15-4	30	0.000706	194.5	-11.6	340.7	67.1
8	PW15-4	40	0.000372	194.5	-12.4	341.4	66.4
9	PW15-4	50	0.000212	194.1	-13.3	341.2	65.4
10	PW15-4	60	0.000135	193.5	-14.0	340.5	64.6
11	PW15-4	70	0.000090	192.0	-12.6	336.1	65.3
12	PW15-4	80	0.000075	192.5	-15.6	339.6	62.7
13	PW15-4	90	0.000058	192.7	-11.5	336.4	66.6
14	PW15-4	100	0.000046	188.5	-10.8	326.9	65.3

```
In [73]: PW15_4a_datablock_geo=[]
PW15_4a_datablock_spec=[]
PW15_4a_DI_geo=[]
PW15_4a_DI_spec=[]
for n in range(0,len(PW15_4a_demag)):
    PW15_4a_datablock_geo.append([PW15_4a_demag['step(mT)'][n],
                                    PW15_4a_demag['dec_geo'][n],
                                    PW15_4a_demag['inc_geo'][n],
                                    PW15_4a_demag['intensity'][n]])
    PW15_4a_datablock_spec.append([PW15_4a_demag['step(mT)'][n],
                                    PW15_4a_demag['dec_spec'][n],
                                    PW15_4a_demag['inc_spec'][n],
                                    PW15_4a_demag['intensity'][n]])
    PW15_4a_DI_geo.append([PW15_4a_demag['dec_geo'][n],
                           PW15_4a_demag['inc_geo'][n]])
    PW15_4a_DI_spec.append([PW15_4a_demag['dec_spec'][n],
                            PW15_4a_demag['inc_spec'][n]])
```

Intensity plot

```
In [74]: plt.figure(figsize=(5,5))
plt.plot(PW15_4a_demag['step(mT)'],PW15_4a_demag['intensity'],marker='o')
plt.xlabel('AF demag step (mT)')
plt.ylabel('Magnetization (Am^2/kg)')
plt.xlim((-1,50))
plt.ylim((0,.04))
plt.savefig('code_output/PW15_4a_intensity.pdf')
plt.show()

print 'The remanence lost (in %) following low-temp cycling is:'
print ((PW15_4a_demag['intensity'][0]-PW15_4a_demag['intensity'][1])/PW15_4a_demag['intensity'][1]) * 100
```



The remanence lost (in %) following low-temp cycling is:

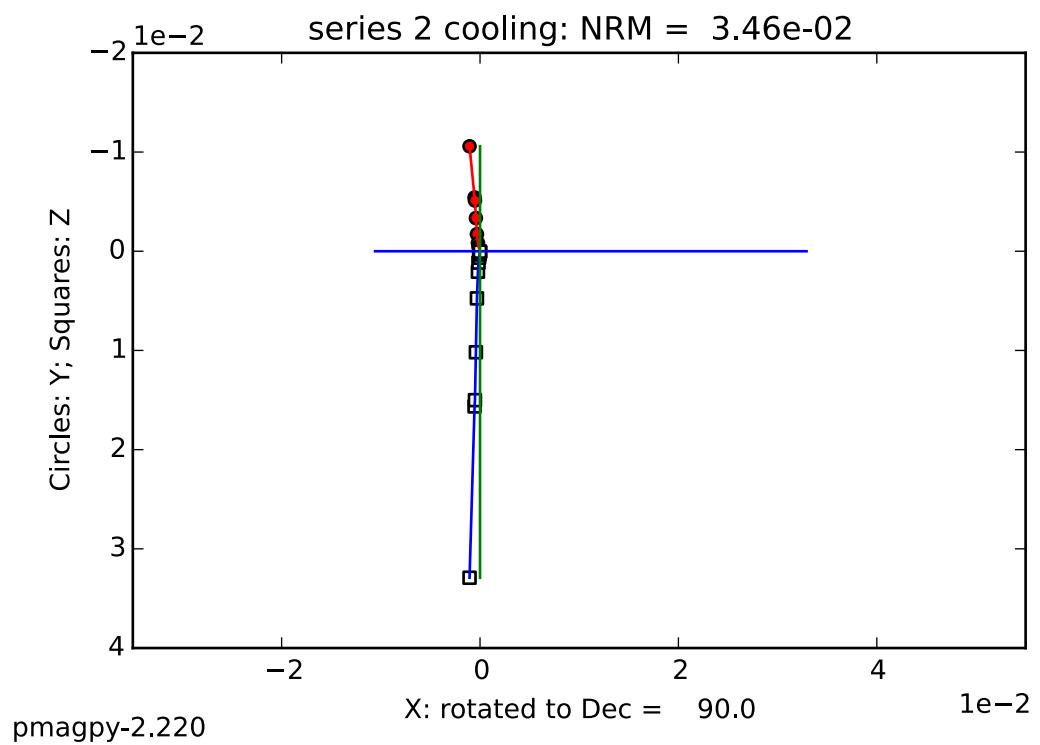
52.116666763

The large loss of remanence is associated with a low-temperature nitrogen bath in near-zero field.

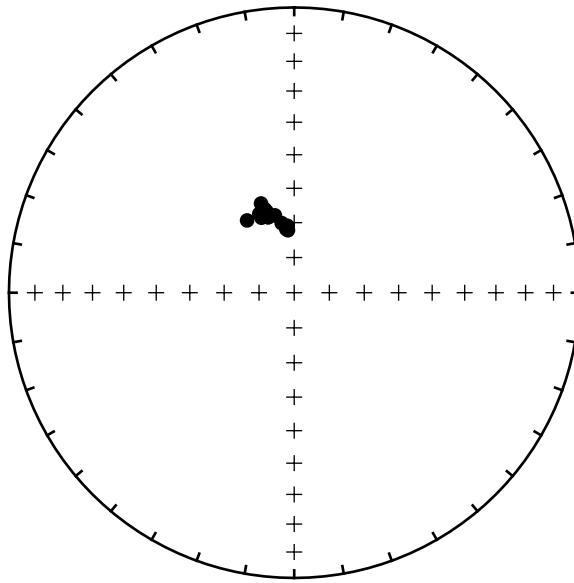
Z-plot and equal area in specimen coordinates

```
In [75]: #plot with pmagplotlib.plotZ(fignum,datablock,angle,s,norm)
    pmagplotlib.plotZ(1,PW15_4a_datablock_spec,90,'series 2 cooling',0)
    plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
    plt.ticklabel_format(style='sci', scilimits=(0,0), axis='x')
    plt.savefig('code_output/PW15_4a_specimen_zplot.pdf')
    plt.show()

    plt.figure(figsize=(5,5))
    pmagplotlib.plotNET(1)
    IPmag.iplotDI(PW15_4a_DI_spec)
    plt.title('Equal Area of demag')
    plt.savefig('code_output/PW15_4a_specimen_eqaplot.pdf')
    plt.show()
```



Equal Area of demag



7.4 IRM-LTI experiments

The experiments conducted on the samples were as follows.

Load the sample with arrow (down core z-axis) pointing out of SRM (old 2G at IRM) and make measurement (measurement series #1, measurement #1). The direction should be approx. the same as the measurement of the sister "a-series" specimen.

Twist the sample screw tight as is needed for running in low-temperature insert which results in sample being rotated about the z-axis (measurement series #1, measurement #2).

Cover the sample holder with the sapphire radiation shield and vacuum shroud and then begin cooling. Measurements are made upon cooling (measurement series #2).

Once at minimum temperature (slightly below 20K) warm the probe back up progressively making measurements as it warms (measurement series #2).

7.4.1 LT-cycling of NRM

Import data file which has the format:

TEMPERATURE (K)

M_x [Am2]

M_y [Am2]

M_z [Am2]

M_total [Am2]
MEASUREMENT SERIES

```
In [76]: PW15_data = pd.read_table('..../Data/Umkondo/PW15/PW15-4d_LTI_experiments.txt')
PW15_data.head()
```

```
Out[76]:   measurement_T[K]    Mx[Am2]    My[Am2]    Mz[Am2]    moment[Am2]  \
0            293.03  0.000007  0.000000  0.000024  0.000025
1            292.90 -0.000002 -0.000007  0.000024  0.000025
2            293.30 -0.000002 -0.000007  0.000024  0.000025
3            293.59 -0.000002 -0.000007  0.000024  0.000025
4            293.26 -0.000002 -0.000007  0.000024  0.000025
```

	Measurement_series
0	1
1	1
2	2
3	2
4	2

```
In [77]: full_vector_oriented=[]
full_vector_rotated=[]
```

```
#data point 0 is oriented
for n in range(0,1):
    full_vector_oriented.append([PW15_data['Mx[Am2]'][n],
                                PW15_data['My[Am2]'][n],
                                PW15_data['Mz[Am2]'][n]])
```



```
#data point 1 is rotated about z
for n in range(1,2):
    full_vector_rotated.append([PW15_data['Mx[Am2]'][n],
                                PW15_data['My[Am2]'][n],
                                PW15_data['Mz[Am2]'][n]])
```

```
oriented_directions=pmag.cart2dir(full_vector_oriented)
rotated_directions=pmag.cart2dir(full_vector_rotated)
```

```
#rotate the rotated directions about the z-axis
#to correspond to the oriented measurement
angle = 360 + oriented_directions.item(0) - rotated_directions.item(0)
```

```
x_prime = PW15_data['Mx[Am2]'][1]*np.cos(np.radians(angle)) - PW15_data['My[Am2]'][1]*np.sin(np.radians(angle))
y_prime = PW15_data['Mx[Am2]'][1]*np.sin(np.radians(angle)) + PW15_data['My[Am2]'][1]*np.cos(np.radians(angle))
z_prime = PW15_data['Mz[Am2]'][1]
```

```
full_vector_rotated_prime= [[x_prime,y_prime,z_prime]]
rotated_directions_prime=pmag.cart2dir(full_vector_rotated_prime)
```

```
#plot the oriented measurement and the rotated measurement
fignum = 1
plt.figure(num=fignum, figsize=(4,4), dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(oriented_directions, 'k')
```

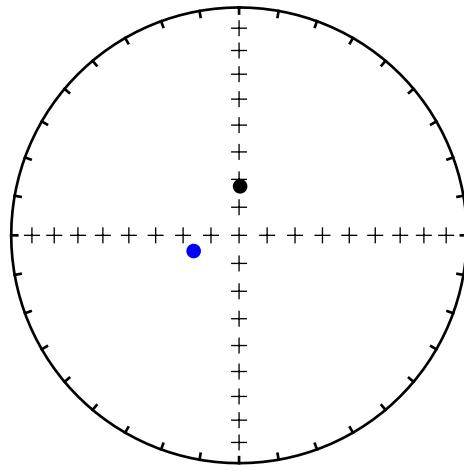
```

IPmag.iplotDI(rotated_directions,'b')
plt.title('PW15-4d NRM oriented (black) and rotated (blue);')

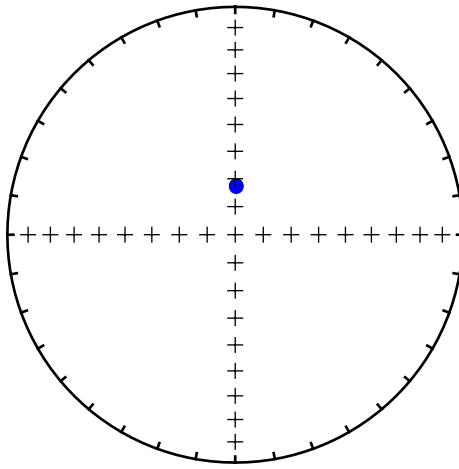
#plot the oriented measurement and the corrected rotated measurement
fignum = 2
plt.figure(num=fignum,figsize=(4,4),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(oriented_directions,'k')
IPmag.iplotDI(rotated_directions_prime,'b')
plt.title('PW15-4d NRM oriented (black) and rotated (corrected; blue)');

```

PW15-4d NRM oriented (black) and rotated (blue)



PW15-4d NRM oriented (black) and rotated (corrected; blue)



It will be more convenient to have the data for measurement series 2 (LT-cycling) split into cooling and warming.

```
In [78]: PW15_s2_cooling = PW15_data[2:121]
PW15_s2_cooling.reset_index(inplace='True')
PW15_s2_warming = PW15_data[129:374]
PW15_s2_warming.reset_index(inplace='True')

PW15_s2_warming.tail()

Out[78]:    index  measurement_T[K]      Mx [Am2]      My [Am2]      Mz [Am2]  moment [Am2]  \
240      369          284.16 -0.000001 -0.000003  0.000012  0.000013
241      370          284.47 -0.000001 -0.000003  0.000012  0.000013
242      371          284.39 -0.000001 -0.000003  0.000012  0.000013
243      372          284.46 -0.000001 -0.000003  0.000012  0.000013
244      373          284.52 -0.000001 -0.000003  0.000012  0.000013

Measurement_series
240              2
241              2
242              2
243              2
244              2
```

The data need to be rotated into the oriented specimen coordinates. These rotated data will be called 'prime'.

```
In [79]: PW15_s2_cooling['Mx_prime[Am2]'] = PW15_s2_cooling['Mx[Am2]']*np.cos(np.radians(angle)) - PW15_
PW15_s2_cooling['My_prime[Am2]'] = PW15_s2_cooling['My[Am2]']*np.sin(np.radians(angle)) + PW15_
```

```

PW15_s2_cooling['Mz_prime[Am2]'] = PW15_s2_cooling['Mz[Am2']]

full_vector_cooling_prime=[]
for n in range(len(PW15_s2_cooling)):
    full_vector_cooling_prime.append([PW15_s2_cooling['Mx_prime[Am2]'][n],
                                      PW15_s2_cooling['My_prime[Am2]'][n],
                                      PW15_s2_cooling['Mz_prime[Am2]'][n]])

PW15_s2_warming['Mx_prime[Am2]'] = PW15_s2_warming['Mx[Am2']]*np.cos(np.radians(angle)) - PW15_s2_cooling['Mx_prime[Am2]']
PW15_s2_warming['My_prime[Am2]'] = PW15_s2_warming['Mx[Am2']]*np.sin(np.radians(angle)) + PW15_s2_cooling['My_prime[Am2]']
PW15_s2_warming['Mz_prime[Am2]'] = PW15_s2_warming['Mz[Am2']]

full_vector_warming_prime=[]
for n in range(len(PW15_s2_warming)):
    full_vector_warming_prime.append([PW15_s2_warming['Mx_prime[Am2]'][n],
                                      PW15_s2_warming['My_prime[Am2]'][n],
                                      PW15_s2_warming['Mz_prime[Am2]'][n]])

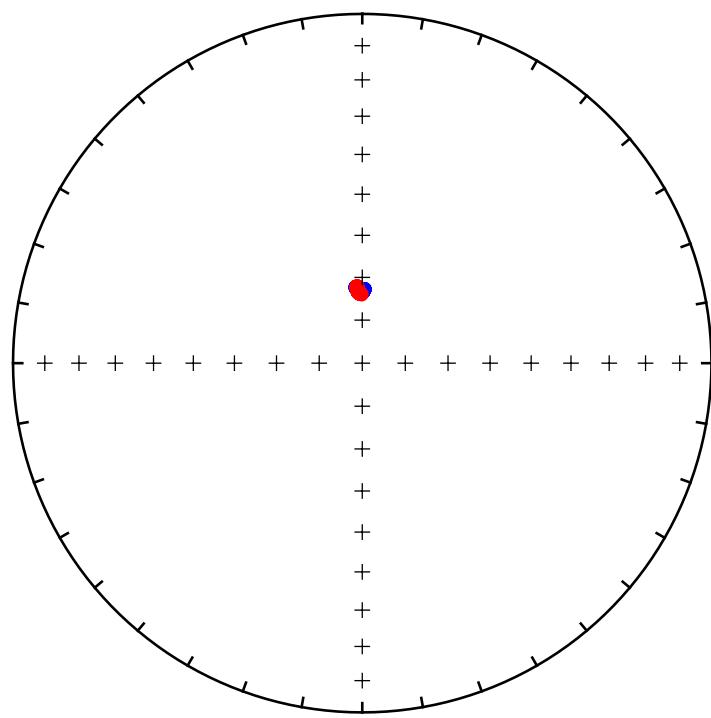

In [80]: cooling_directions_prime=pmag.cart2dir(full_vector_cooling_prime)
          warming_directions_prime=pmag.cart2dir(full_vector_warming_prime)

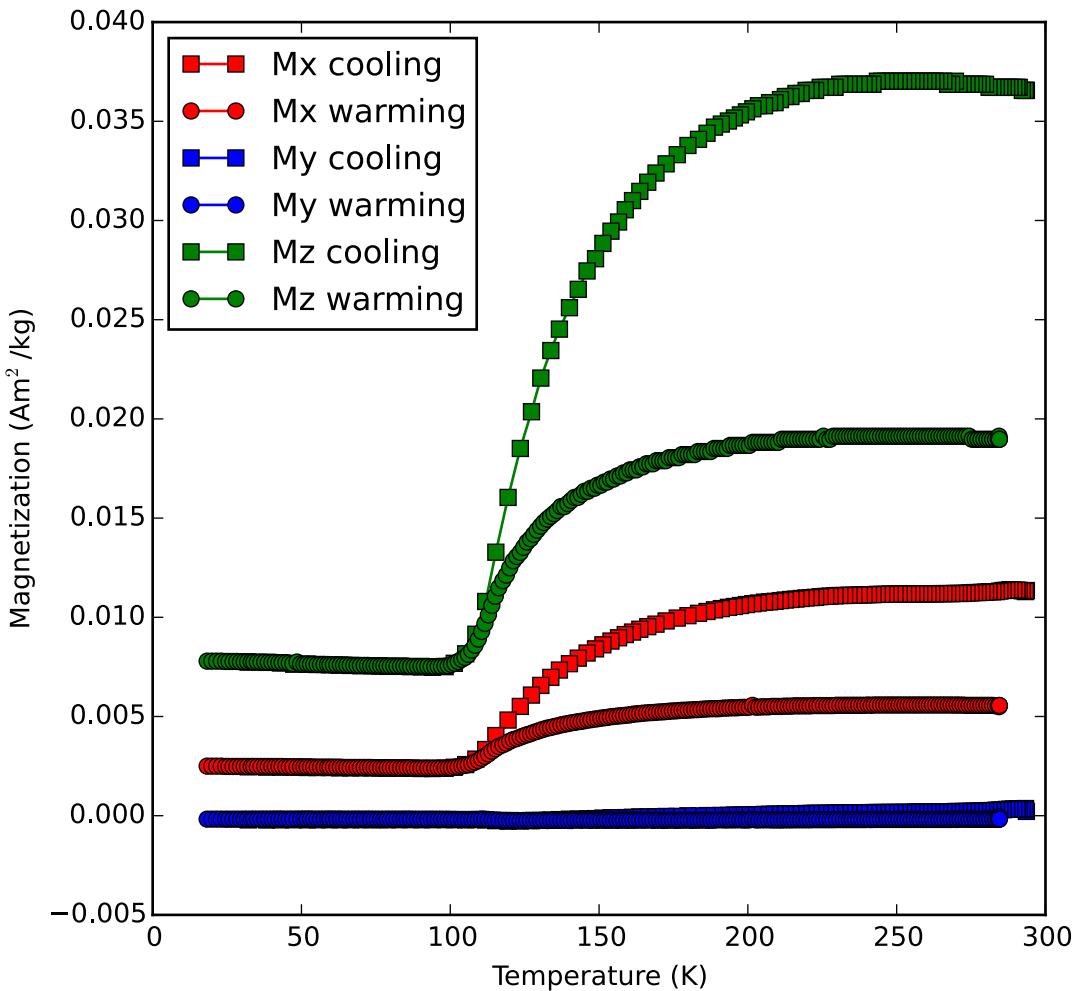
          figure(num=1,figsize=(6,6),dpi=160)
          pmagplotlib.plotNET(figure)
          IPmag.iplotDI(cooling_directions_prime,'b')
          IPmag.iplotDI(warming_directions_prime,'r')
          plt.title('PW15-4d NRM upon cooling and warming (specimen coordinates)')
          plt.savefig('code_output/PW15-4d_equalarea.pdf')
          plt.show()

#the specimen has a mass of .6483 g
figure(num=2,figsize=(6,6),dpi=160)
plt.plot(PW15_s2_cooling['measurement_T[K'],
                           PW15_s2_cooling['Mx_prime[Am2]']/(.6483/1000), 'rs-')
plt.plot(PW15_s2_warming['measurement_T[K'],
                           PW15_s2_warming['Mx_prime[Am2]']/(.6483/1000), 'ro-')
plt.plot(PW15_s2_cooling['measurement_T[K'],
                           PW15_s2_cooling['My_prime[Am2]']/(.6483/1000), 'bs-')
plt.plot(PW15_s2_warming['measurement_T[K'],
                           PW15_s2_warming['My_prime[Am2]']/(.6483/1000), 'bo-')
plt.plot(PW15_s2_cooling['measurement_T[K'],
                           PW15_s2_cooling['Mz_prime[Am2]']/(.6483/1000), 'gs-')
plt.plot(PW15_s2_warming['measurement_T[K'],
                           PW15_s2_warming['Mz_prime[Am2]']/(.6483/1000), 'go-')
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetization (Am$^2$/kg)')
plt.legend(('Mx cooling','Mx warming','My cooling',
           'My warming','Mz cooling','Mz warming'),loc=0)
plt.savefig('code_output/PW15-4d_3axis.svg')
plt.show()

```

PW15-4d NRM upon cooling and warming (specimen coordinates)





```
In [81]: moment_290 = cooling_directions_prime[1,2]
moment_20 = cooling_directions_prime[118,2]
moment_postcycle = warming_directions_prime[244,2]

print "Percent remanence remaining at 20K"
print moment_20/moment_290*100
print "Percent remanence remaining after thermal cycling"
print moment_postcycle/moment_290*100
```

```
Percent remanence remaining at 20K
21.234697426
Percent remanence remaining after thermal cycling
51.6463378548
```

7.4.2 ARM/pARM acquisition and low-temperature cycling

This experiment was designed to impart an ARM to the sample and a perpendicular pARM (to low coercivities) to the sample in order to ascertain the behavior of the low-coercivity ARM and resolve directional change during low-temperature cycling.

Give the specimen a negative z-axis ARM (0.03 mT bias field in 200 mT AF field) and measure (measurement 1 in measurement series #3) (after the measurement the decision was made to increase the bias field in order to increase the moment)

Give the specimen a negative z-axis ARM (0.1 mT bias field in 200 mT AF field) and measure (measurement 2 in measurement series #3)

Give the specimen a negative x-axis ARM (0.1 mT bias field in 5 mT AF field) and measure (measurement 3 in measurement series #3) (after the measurement the decision was made to increase the of this pARM bias field in order to increase the moment)

Give the specimen a negative x-axis ARM (0.2 mT bias field in 5 mT AF field) and measure (measurement 4 in measurement series #3)

Tightening the sample holder for the cryogenic experiment requires rotating the sample an unknown amount about the z-axis (measurement 5 in measurement series #3)

Cover the sample holder with the sapphire radiation shield and vacuum shroud and then begin cooling. Measurements are made upon cooling and subsequent warming (measurement series #4).

ARM/pARM acquisition

```
In [82]: zARM_oriented=[]
zARMxARM_oriented=[]
zARMxARM_rotated=[]

#negative z-axis ARM (0.1 mT bias field in 200 mT AF field)
for n in range(375,376):
    Mx_full=PW15_data['Mx[Am2]'][n]
    My_full=PW15_data['My[Am2]'][n]
    Mz_full=PW15_data['Mz[Am2]'][n]
    zARM_oriented.append([Mx_full,My_full,Mz_full])

#negative z-axis ARM (0.1 mT bias field in 200 mT AF field) + negative x-axis ARM (0.2 mT bias
for n in range(377,378):
    Mx_full=PW15_data['Mx[Am2]'][n]
    My_full=PW15_data['My[Am2]'][n]
    Mz_full=PW15_data['Mz[Am2]'][n]
    zARMxARM_oriented.append([Mx_full,My_full,Mz_full])

#negative z-axis ARM (0.1 mT bias field in 200 mT AF field) + negative x-axis ARM (0.2 mT bias
for n in range(378,379):
    Mx_full=PW15_data['Mx[Am2]'][n]
    My_full=PW15_data['My[Am2]'][n]
    Mz_full=PW15_data['Mz[Am2]'][n]
    zARMxARM_rotated.append([Mx_full,My_full,Mz_full])

zARM_oriented_directions=pmag.cart2dir(zARM_oriented)
zARMxARM_oriented_directions=pmag.cart2dir(zARMxARM_oriented)
zARMxARM_rotated_directions=pmag.cart2dir(zARMxARM_rotated)

#rotate the rotated directions about the z-axis to correspond to the oriented measurement
angle = zARMxARM_oriented_directions.item(0) - zARMxARM_rotated_directions.item(0)

x_prime = PW15_data['Mx[Am2]'][378]*np.cos(np.radians(angle)) - PW15_data['My[Am2]'][378]*np.s
y_prime = PW15_data['Mx[Am2]'][378]*np.sin(np.radians(angle)) + PW15_data['My[Am2]'][378]*np.c
z_prime = PW15_data['Mz[Am2]'][378]

zARMxARM_rotated_prime = [[x_prime,y_prime,z_prime]]
```

```

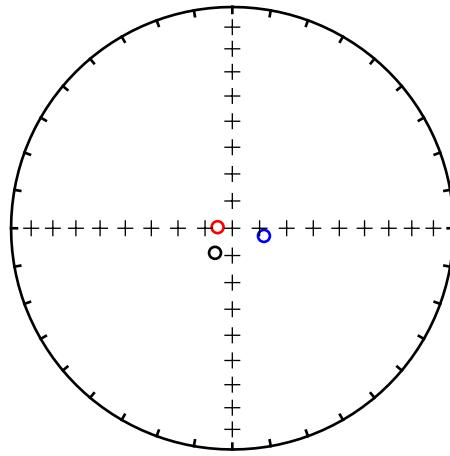
zARMxARM_rotated_directions_prime=pmag.cart2dir(zARMxARM_rotated_prime)

#plot the oriented measurements and the rotated measurement
fignum = 1
plt.figure(num=fignum,figsize=(4,4),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(zARM_oriented_directions,'r')
IPmag.iplotDI(zARMxARM_oriented_directions,'k')
IPmag.iplotDI(zARMxARM_rotated_directions,'b')
plt.title('zARM oriented (red) z&xARM oriented (black) and rotated (blue)');
plt.savefig('code_output/PW15-4d_eqarea_ARMex.svg')

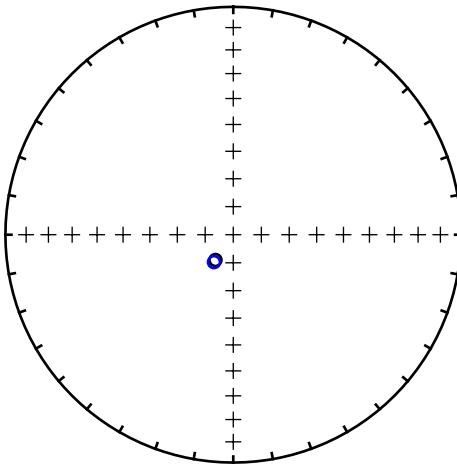
#plot the oriented measurement and the corrected rotated measurement
fignum = 2
plt.figure(num=fignum,figsize=(4,4),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(zARMxARM_oriented_directions,'k')
IPmag.iplotDI(zARMxARM_rotated_directions_prime,'b')
plt.title('z&xARM oriented (black) and rotated (corrected; blue)');

```

zARM oriented (red) z&xARM oriented (black) and rotated (blue)



z&xARM oriented (black) and rotated (corrected; blue)



```
In [83]: #these values were needed to plot with the low-temp cycling data
#on the vector component and equal area plot
print zARM_oriented_directions
print zARMAxARM_rotated_directions_prime

[[ 2.74201770e+02 -8.45885759e+01  7.45321764e-07]]
[[ 2.15129124e+02 -7.80651803e+01  7.16487969e-07]]
```

Low-temperature cycling

```
In [84]: data_file_cooling='../../Data/Umkondo/PW15/Pw15-4d_ARMex_cooling.txt'
data_cooling = np.genfromtxt(data_file_cooling, skip_header=1)

temp_cooling=data_cooling[:,0]
Mx_cooling=data_cooling[:,1]
My_cooling=data_cooling[:,2]
Mz_cooling=data_cooling[:,3]
Mtotal_cooling=data_cooling[:,4]

data_file_warming='../../Data/Umkondo/PW15/Pw15-4d_ARMex_warming.txt'
data_warming = np.genfromtxt(data_file_warming, skip_header=1)

temp_warming=data_warming[:,0]
Mx_warming=data_warming[:,1]
My_warming=data_warming[:,2]
Mz_warming=data_warming[:,3]
Mtotal_warming=data_warming[:,4]
```

```

#For the interpolate function to work the x values (temperature) need to be ascending.
#The blank_hem_data_cooling can be sorted so that temperatures are ascending
blank2_data_cooling = blank2_data_cooling.sort(['measurement_T[K]'])

Mx_cooling_bsub = background_sub(blank2_data_cooling['Mx[Am2]'],
                                  blank2_data_cooling['measurement_T[K]'],
                                  Mx_cooling,temp_cooling)

My_cooling_bsub = background_sub(blank2_data_cooling['My[Am2]'],
                                  blank2_data_cooling['measurement_T[K]'],
                                  My_cooling,temp_cooling)

Mz_cooling_bsub = background_sub(blank2_data_cooling['Mz[Am2]'],
                                  blank2_data_cooling['measurement_T[K]'],
                                  Mz_cooling,temp_cooling)

Mx_warming_bsub = background_sub(blank2_data_warming['Mx[Am2]'],
                                  blank2_data_warming['measurement_T[K]'],
                                  Mx_warming,temp_warming)

My_warming_bsub = background_sub(blank2_data_warming['My[Am2]'],
                                  blank2_data_warming['measurement_T[K]'],
                                  My_warming,temp_warming)

Mz_warming_bsub = background_sub(blank2_data_warming['Mz[Am2]'],
                                  blank2_data_warming['measurement_T[K]'],
                                  Mz_warming,temp_warming)

Mtotal_cooling_bsub=[]
Mtotal_warming_bsub=[]

for n in range(0,len(temp_cooling)):
    Mtotal_cooling_bsub.append(np.sqrt(Mx_cooling_bsub[n]**2+My_cooling_bsub[n]**2+Mz_cooling_bsub[n]**2))

for n in range(0,len(temp_warming)):
    Mtotal_warming_bsub.append(np.sqrt(Mx_warming_bsub[n]**2+My_warming_bsub[n]**2+Mz_warming_bsub[n]**2))

fignum=1
plt.figure(num=fignum,figsize=(10,12),dpi=160)

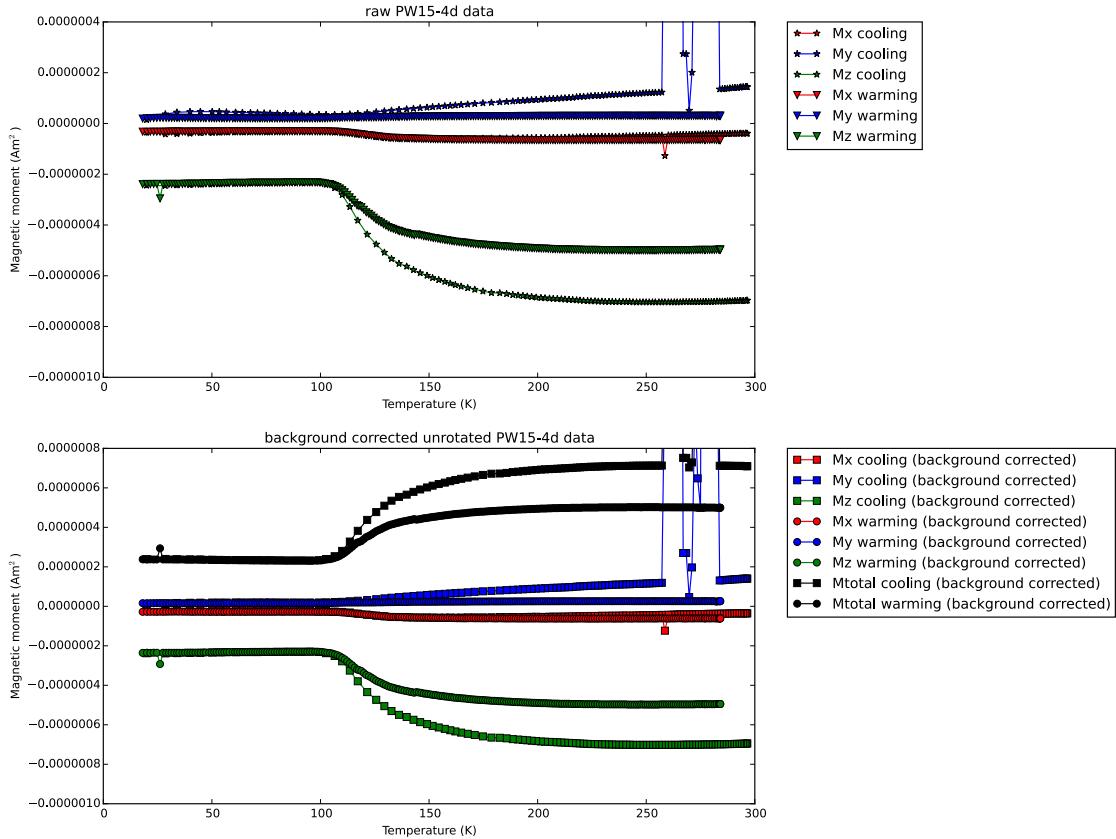
ax=plt.subplot(211)
plt.plot(temp_cooling, Mx_cooling, 'r*-',label="Mx cooling")
plt.plot(temp_cooling, My_cooling, 'b*-',label="My cooling")
plt.plot(temp_cooling, Mz_cooling, 'g*-',label="Mz cooling")
plt.plot(temp_warming, Mx_warming, 'rv-',label="Mx warming")
plt.plot(temp_warming, My_warming, 'bv-',label="My warming")
plt.plot(temp_warming, Mz_warming, 'gv-',label="Mz warming")
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.axis([0, 300, -0.000001, 0.000004])
plt.title('raw PW15-4d data')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

```

```

ax=plt.subplot(212)
plt.plot(temp_cooling, Mx_cooling_bsub, 'rs-',
         label="Mx cooling (background corrected)")
plt.plot(temp_cooling, My_cooling_bsub, 'bs-',
         label="My cooling (background corrected)")
plt.plot(temp_cooling, Mz_cooling_bsub, 'gs-',
         label="Mz cooling (background corrected)")
plt.plot(temp_warming, Mx_warming_bsub, 'ro-',
         label="Mx warming (background corrected)")
plt.plot(temp_warming, My_warming_bsub, 'bo-',
         label="My warming (background corrected)")
plt.plot(temp_warming, Mz_warming_bsub, 'go-',
         label="Mz warming (background corrected)")
plt.plot(temp_cooling, Mtotal_cooling_bsub, 'ks-',
         label="Mtotal cooling (background corrected)")
plt.plot(temp_warming, Mtotal_warming_bsub, 'ko-',
         label="Mtotal warming (background corrected)")
plt.xlabel('Temperature (K)')
plt.ylabel('Magnetic moment (Am$^2$)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('background corrected unrotated PW15-4d data')
plt.axis([0, 300, -0.0000001, 0.0000008])
plt.show()

```



The fan turning on and off on another instrument in the lab resulted in flux jumps during a portion of time during initial cooling. These data need to be filtered out of the data set which is done below. The data also need to be rotated into the specimen's core coordinates.

```
In [85]: Mx_cooling_prime = Mx_cooling_bsub*np.cos(np.radians(angle)) - My_cooling_bsub*np.sin(np.radians(angle))
My_cooling_prime = Mx_cooling_bsub*np.sin(np.radians(angle)) + My_cooling_bsub*np.cos(np.radians(angle))
Mz_cooling_prime = Mz_cooling_bsub
Mx_cooling_prime_nf = []
My_cooling_prime_nf = []
Mz_cooling_prime_nf = []
Mtotal_cooling_prime_nf = []
full_vector_cooling=[]
temp_cooling_prime = []
temp_cooling_prime_nf = []

for n in range(0,len(temp_cooling)):
    if My_cooling[n] < 1.4500000e-07 and temp_cooling[n] != 269.77999999999997:
        #this filter is instituted to remove data points that are spurious due to flux jumps
        Mx_cooling_prime_nf.append(Mx_cooling_prime[n])
        My_cooling_prime_nf.append(My_cooling_prime[n])
        Mz_cooling_prime_nf.append(Mz_cooling_prime[n])
        temp_cooling_prime_nf.append(temp_cooling[n])
        Mtotal_cooling_prime_nf.append(np.sqrt(Mx_cooling_prime[n]**2+My_cooling_prime[n]**2+Mz_cooling_prime[n]**2))
        full_vector_cooling.append([Mx_cooling_prime[n],My_cooling_prime[n],Mz_cooling_prime[n]])

Mx_warming_prime = Mx_warming_bsub*np.cos(np.radians(angle)) - My_warming_bsub*np.sin(np.radians(angle))
My_warming_prime = Mx_warming_bsub*np.sin(np.radians(angle)) + My_warming_bsub*np.cos(np.radians(angle))
Mz_warming_prime = Mz_warming_bsub

Mtotal_warming_prime= []
full_vector_warming=[]

for n in range(0,len(temp_warming)):
    full_vector_warming.append([Mx_warming_prime[n],My_warming_prime[n],Mz_warming_prime[n]])
    Mtotal_warming_prime.append(np.sqrt(Mx_warming_prime[n]**2+My_warming_prime[n]**2+Mz_warming_prime[n]**2))
```

These rotated, background corrected data are plotted below along with data that shows the change in magnetocrystalline anisotropy as a function of temperature for the sake of comparision with the patterns of demagnetization observed for the z-axis component (full ARM up to 200 mT except 0 to 5 mT) and the x-axis pARM (applied to low coercivity grains with a 0 to 5 mT AF field).

```
In [86]: fignum=2
plt.figure(num=fignum, figsize=(5,10), dpi=160)
ax=plt.subplot(311)
plt.plot(temp_cooling_prime_nf, np.negative(Mtotal_cooling_prime_nf),
         'ks-',label="M$_{\text{total}}$ cooling")
plt.plot(temp_warming, np.negative(Mtotal_warming_prime),
         'ko-',label="M$_{\text{total}}$ warming")
plt.plot(temp_cooling_prime_nf, Mx_cooling_prime_nf, 'rs-',label="M$_x$ cooling")
plt.plot(temp_warming, Mx_warming_prime, 'ro-',label="M$_x$ warming")
#plt.plot(temp_cooling_prime_nf, My_cooling_prime_nf, 'bs-',label="M$_y$ cooling")
#plt.plot(temp_warming, My_warming_prime, 'bo-',label="M$_y$ warming")
plt.plot(temp_cooling_prime_nf, Mz_cooling_prime_nf, 'gs-',label="M$_z$ cooling")
plt.plot(temp_warming, Mz_warming_prime, 'go-',label="M$_z$ warming")
plt.vlines(130, 1e-7, -8e-7, colors='b')
```

```

plt.vlines(122, 1e-7,-8e-7, colors='c')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.ylabel('Magnetic moment (Am$^2$)')
# plt.title('background corrected and rotated PW6-5f data')
plt.ylim(1e-7,-8e-7)
plt.savefig('code_output/PW15-4d_3axisLTD_ARMex.svg')

ax=plt.subplot(312)
plt.plot(temp_cooling_prime_nf, Mx_cooling_prime_nf/Mx_cooling_prime_nf[0],
         'rs-',label="Mx cooling")
plt.plot(temp_warming, Mx_warming_prime/Mx_cooling_prime_nf[0],
         'ro-',label="Mx warming")
plt.plot(temp_cooling_prime_nf, Mz_cooling_prime_nf/Mz_cooling_prime_nf[0],
         'gs-',label="Mx cooling")
plt.plot(temp_warming, Mz_warming_prime/Mz_cooling_prime_nf[0],
         'go-',label="Mx warming")
plt.vlines(130, -.2, 1.2, colors='b')
plt.vlines(122, -.2, 1.2, colors='c')
# plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.ylabel('Magnetic moment (normalized to initial)')
# plt.title('background corrected and rotated PW6-5f data')
plt.ylim(-.2, 1.2)
plt.savefig('code_output/PW15-4d_3axisLTD_ARMex.svg')

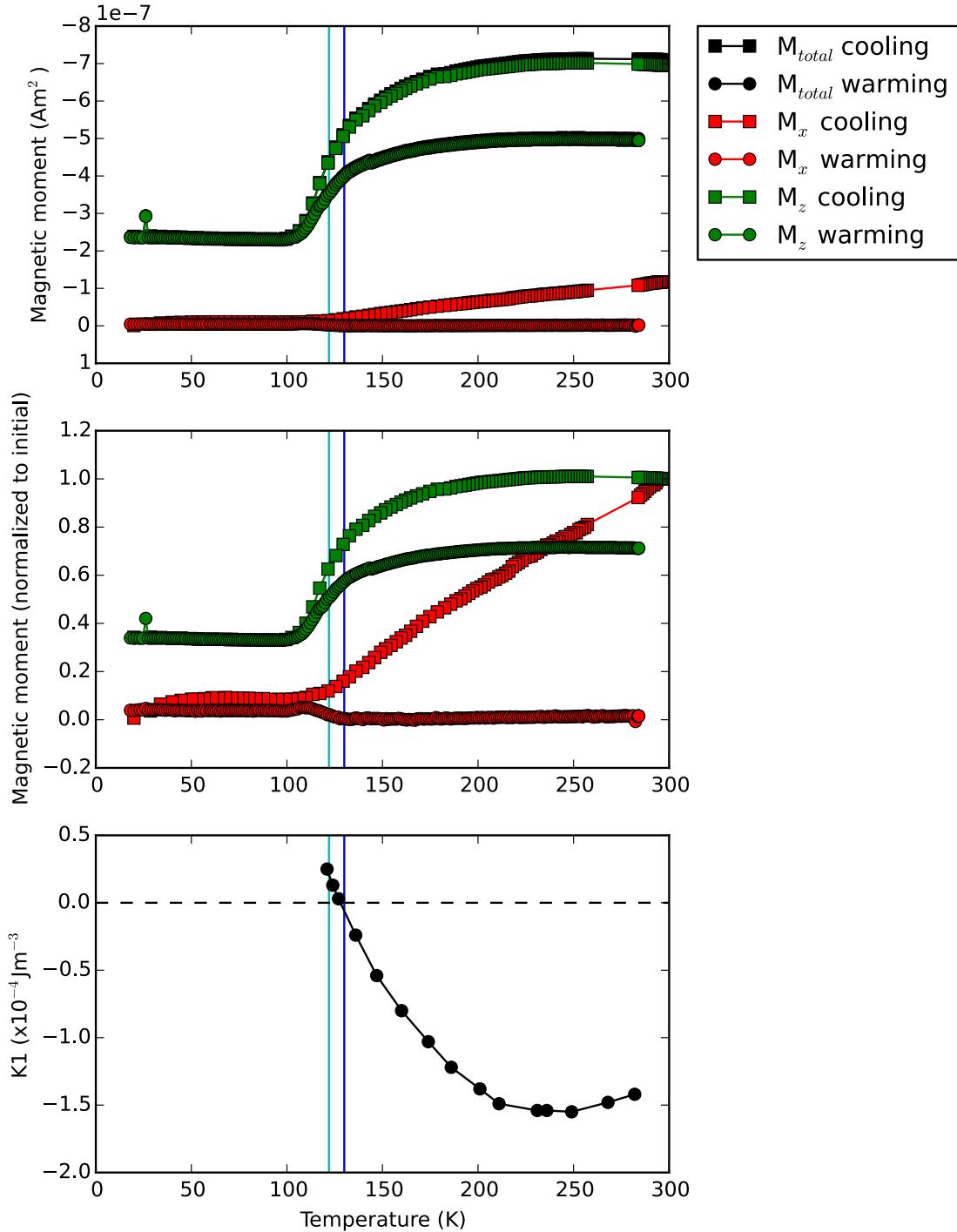
Bickford_data = pd.read_csv('..../Data/Umkondo/Bickford1957a_data.csv')

ax=plt.subplot(313)
plt.plot(Bickford_data['temp'],Bickford_data['Kone'], 'ko-')
plt.vlines(130, -2, 0.5, colors='b')
plt.vlines(122, -2, 0.5, colors='c')
plt.hlines(0, 0, 300, colors='k',linestyles='--')
plt.xlim(0,300)
plt.ylim(-2,0.5)
plt.xlabel('Temperature (K)')
plt.ylabel('K1 (x10$^{-4}$Jm$^{-3}$)')

cooling_directions=pmag.cart2dir(full_vector_cooling)
warming_directions=pmag.cart2dir(full_vector_warming)

plt.savefig('code_output/PW15-4d_3axisLTD_ARMex.pdf')

```



```
In [87]: Mz_initial = np.negative(Mz_cooling_prime_nf[0])
Mz_end = np.negative(Mz_warming_prime[243])
```

```
print 'Percent of initial remanence remaining along z axis'
print Mz_end/Mz_initial*100
```

```
Percent of initial remanence remaining along z axis  
71.2291884462
```

Let's export the data in order to make a space-delimited file to be plotted using the PmagPy program zeq.py outside of this IPython notebook.

```
In [88]: np.savetxt("code_output/PW15-4d_cooling.txt", cooling_directions, delimiter=" ")  
np.savetxt("code_output/PW15-4d_coolingtemp.txt", temp_cooling_prime_nf, delimiter=" ")  
np.savetxt("code_output/PW15-4d_warming.txt", warming_directions, delimiter=" ")  
np.savetxt("code_output/PW15-4d_warmingtemp.txt", temp_warming, delimiter=" ")  
lowT_endpoint = warming_directions[len(warming_directions)-1]  
print lowT_endpoint
```

[2.68489944e+02 -8.23278382e+01 4.99277242e-07]

Another figure to make is to compare the direction of the z-axis ARM with the z-axis ARM + x-axis pARM with the end point following low-temperature cycling.

```
In [89]: fignum = 1  
plt.figure(num=fignum, figsize=(6,6), dpi=160)  
pmagplotlib.plotNET(fignum)  
IPmag.iplotDI(zARM_oriented_directions, 'r')  
IPmag.iplotDI(zARMxARM_rotated_directions_prime, 'k')  
IPmag.iplotDI([[268.4,-82.3]], 'b')  
plt.title('PW15-4d zARM oriented (red) z&xARM oriented (black) and following lowT-demag (blue)')  
plt.savefig('code_output/PW15-4d_eqarea_ARMex.svg')
```

PW15-4d zARM oriented (red) z&xARM oriented (black) and following lowT-demag (blue)

