MATools Data Processing Example

This example shows how to perform common operations on ARPES image data using the functions of the MATools toolkit.

For a more in-depth explanation of potential in- and output of those functions, consult the UserGuide.

Reading in Data

A datafile in the HDF5 or SP format can be read using the *ReadARPES* function. The function will return all data from the specified file in the order shown in the command below.

The process is confirmed in the MATLAB command window by displaying "-Reading in data" as shown.

```
[Angle,Energy,Scan,Data,ep,Note]=ReadARPES('Data_Ex1.h5');
```

```
- Reading in data
```

The 'Note' contains information about the ARPES setup. The *disp* command below displays this information in the MATLAB command window.

Note how in this case, hv is given as a range in MATLAB syntax. This will be the case for whatever quantity was measured in Scan.

```
disp(Note)
```

```
hv      = 290:100:1090
Pol     = C+
Slit    = 50
Mode    = LAD
Epass   = 200
Ek/Eb   = -5
dt      = 60
Sweeps  = 2
Theta   = -0.30592
Tilt    = 2.3
Azimuth = 59.0018
Comment =
```

General Processing

Use *CurveCorr* to get the curvature corrected energy scale. Note that this step is obsolete for data taken after 2014 (see UserGuide). The action is confirmed in the command window.

```
ECorr=CurveCorr(Angle,Energy,ep);
```

```
- Curvature correction
```

Align the energy scale at the Fermi level of the different frames using *AlignEF*. The action is again confirmed in the command window. EAlign is the proper energy scale that will be used from here on out whenver a function requires it as an input.

```
EAlign=AlignEF(Data,ECorr);
```

- EF alignment

Displaying the Data

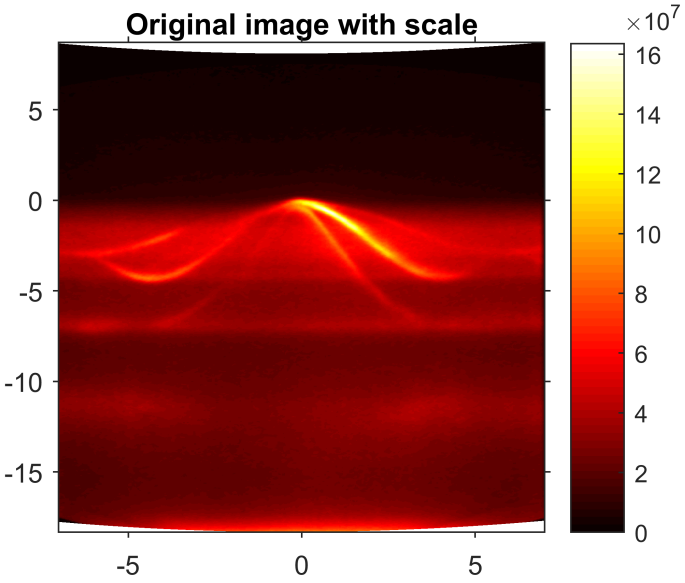
First we select one of the frames of constant hv, the seventh in this example, and then display it using the *ImData* function and the colorscale option 'hot'. This function will be used throughout this example to create heatmap-style plots. Since it creates a regular MATLAB figure via the *pcolor* function, all figure-modifying commands can be used on its output.

Note the slight curvature of the energy scale due to the previous correction.

A scale can be included using the 'colorbar' command.

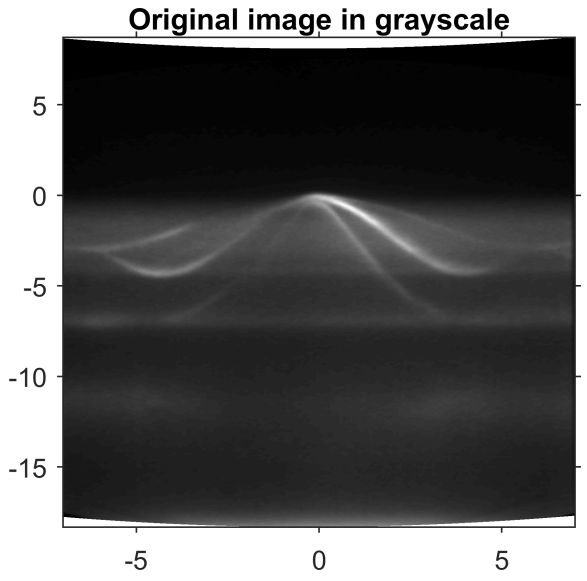
```
nFrame=7; EFrame=EAlign(:,:,nFrame); DataFrame=Data(:,:,nFrame);

figure; ImData(Angle,EFrame,DataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out');
title('Original image with scale');colorbar;set(gcf,'Position',[0,0,380,300])
```



Grayscale images can be produced using the colormap option 'gray'.

```
figure; ImData(Angle,EFrame,DataFrame,'Flat');
colormap gray; set(gca,'TickDir','Out');
title('Original image in grayscale');set(gcf,'Position',[0,0,330,300])
```



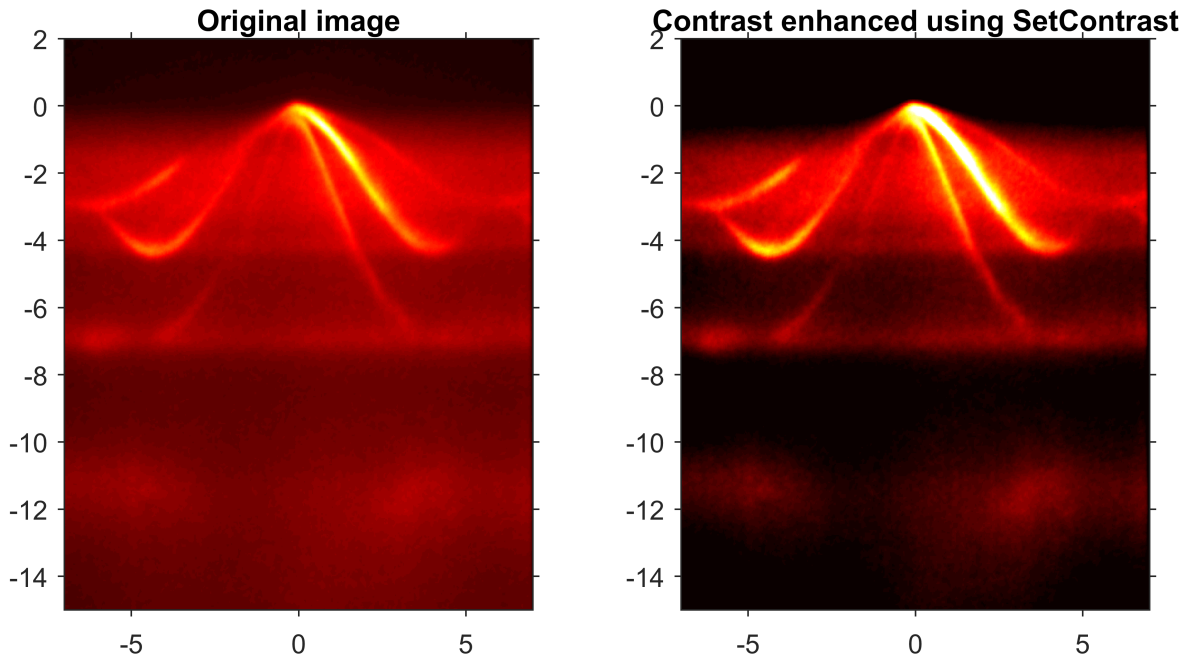
Processing the Image

We use the function *SetContrast* to set the value of the least intense 45% of points to 0 and the intensity of the most intense 0.2% to 1 for the selected frame and display the result in comparison to the original. This results in enhanced contrast as shown in the images below.

```
ContDataFrame=SetContrast(DataFrame,0.45,0.998);

figure;subplot(121);ImData(Angle,EFrame,DataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); limit2=[-15,2]; ylim(limit2)
title('Original image')

subplot(122);ImData(Angle,EFrame,ContDataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); ylim(limit2)
title('Contrast enhanced using SetContrast');set(gcf,'Position',[0,0,700,350])
```

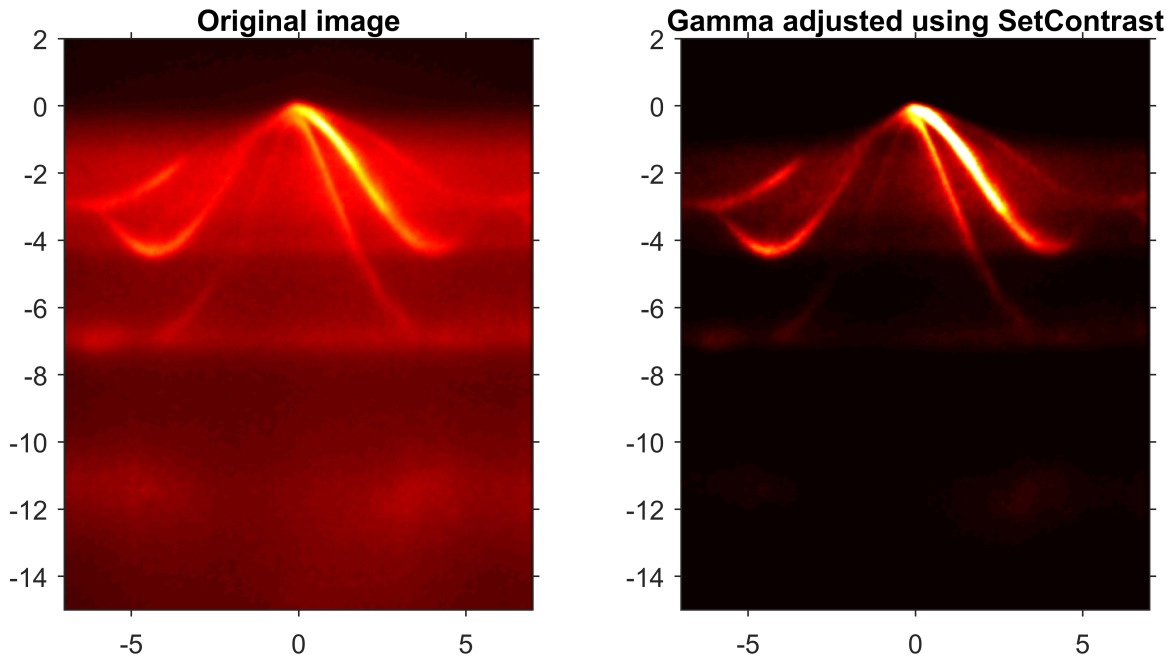


The contrast may also be increased by changing the gamma value from its default of 1 by using a third argument to *SetContrast.*

```
ContDataFrame=SetContrast(DataFrame,0.45,0.998,1.8);

figure;subplot(121);ImData(Angle,EFrame,DataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); ylim(limit2);
title('Original image')

subplot(122); ImData(Angle,EFrame,ContDataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); ylim(limit2);
title('Gamma adjusted using SetContrast');set(gcf,'Position',[0,0,700,350])
```

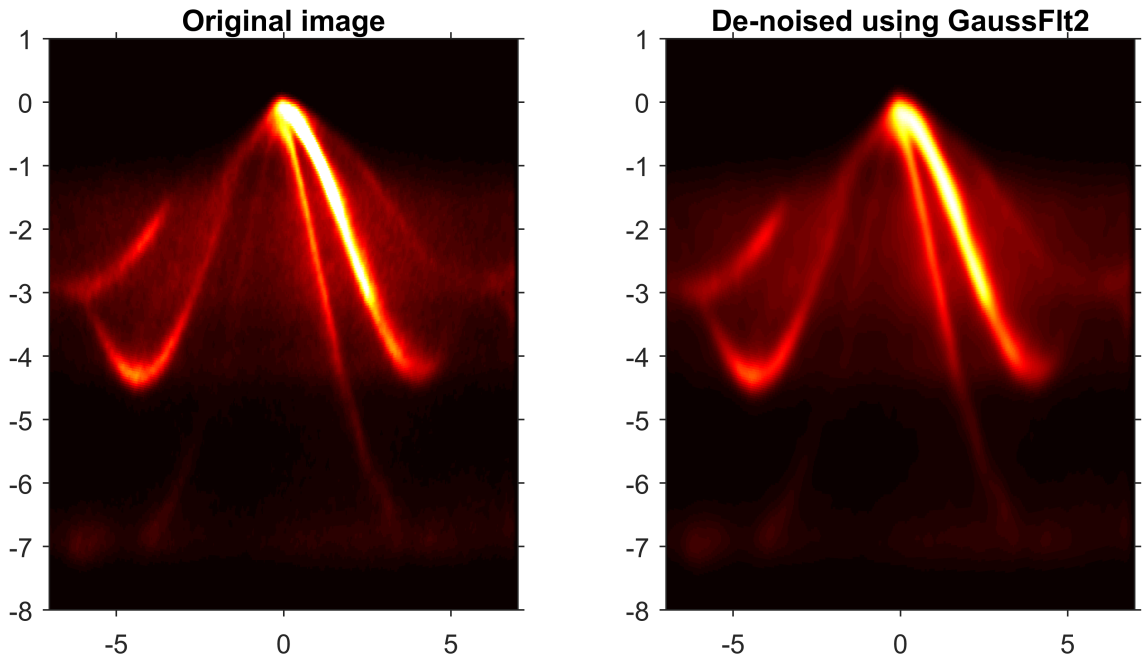


The function *GaussFlt2* applies a Gaussian low-pass filter to the image with four parameters defining HWHM and support in x and y direction.

```
DNDataFrame=GaussFlt2(ContDataFrame,15,15,40,40);

figure;subplot(121);ImData(Angle,EFrame,ContDataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); limit=[-8,1]; ylim(limit);
title('Original image')

subplot(122); ImData(Angle,EFrame,DNDataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out'); ylim(limit)
title('De-noised using GaussFlt2'); set(gcf,'Position',[0,0,700,350])
```



Subtracting an angle-integrated spectrum removes non-dispersive features from the data. The integration can be performed using *IntAngle.*

Note that the contrast was also adjusted in the example below.

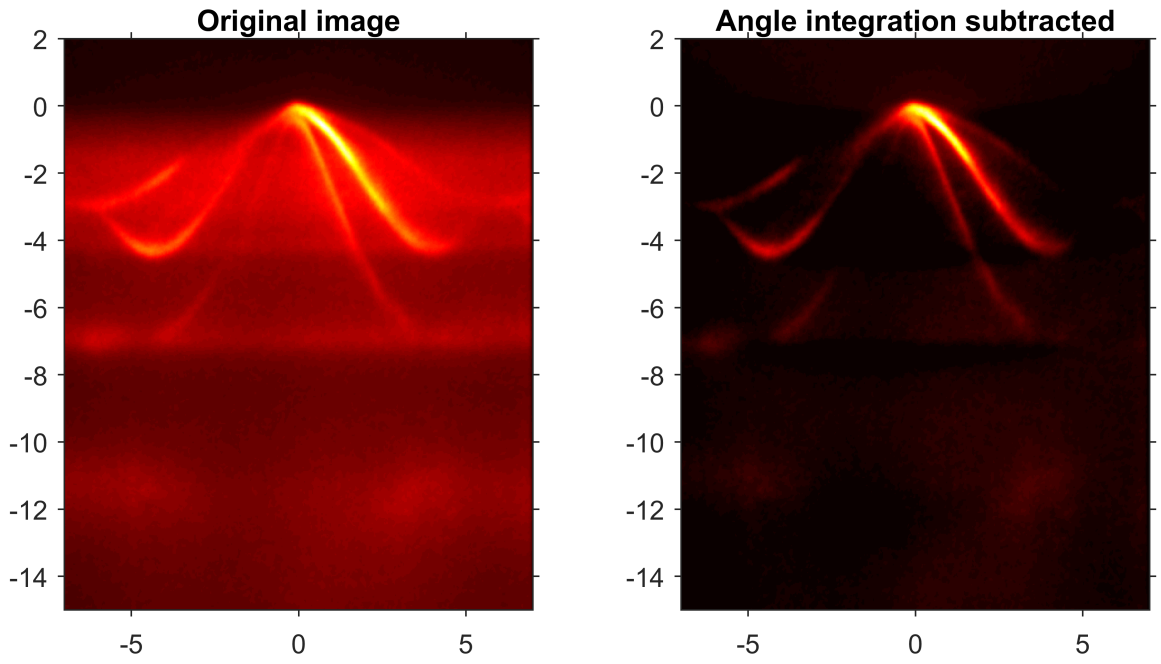
```
DataInt=IntAngle(DataFrame,Angle,EFrame);
```

```
 - Angle integrating
```

```
NormData=DataFrame-DataInt;

NormData=SetContrast(NormData,0.15,1.);
figure;subplot(121);ImData(Angle,EFrame,DataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out');ylim(limit2)

title('Original image');
subplot(122);ImData(Angle,EFrame,NormData,'Flat');
colormap hot; ylim(limit2)
title('Angle integration subtracted'); set(gcf,'Position', [0,0,700,350])
```



Cuts and Slices

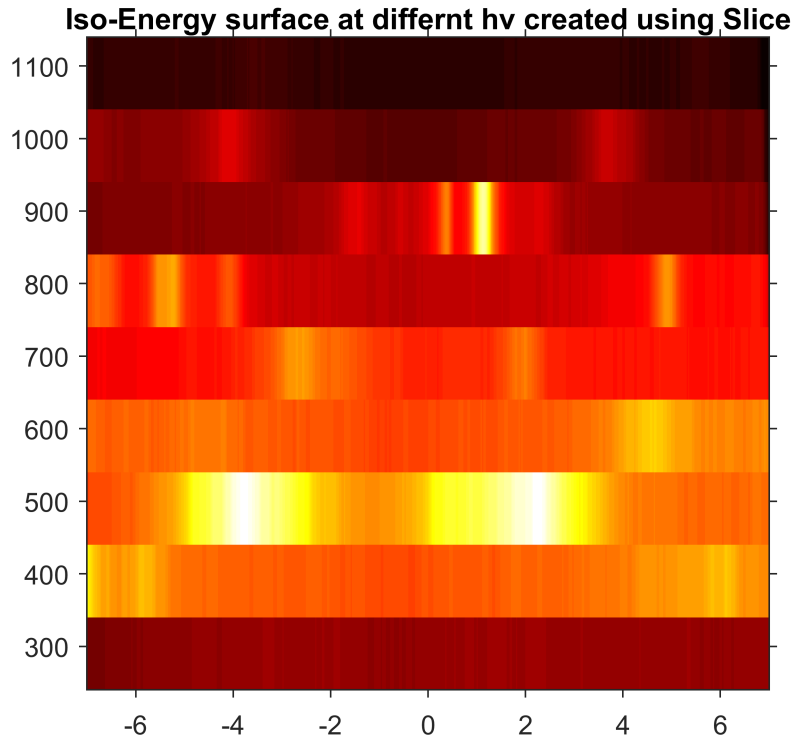
MATools also provides functions for cutting and slicing the data along planes of both constant energy or angle/k.

Below we create and display an iso-energy surface for the different hv. Iso-angle slices can be performed using the 'IsoK' option.

```
EWindow=[-1.05 -0.95];
[ASlice, XSlice]=Slice(Angle,EAlign,Data,'IsoE',EWindow);
```

```
 - Data slice formation
```

```
figure; ImData(XSlice,Scan,ASlice,'Flat');
colormap hot; set(gca,'TickDir','Out');set(gcf,'Position',[0,0,440,400])
title('Iso-Energy surface at differnt hv created using Slice')
```



Performing cuts is accomplished using the *Cut* function in very similar syntax to *Slice*. Note however that *Cut* requires 2D inputs and returns 1D output. Here, we perform a constant k cut through our previously selected frame and display it using the standard *plot* function. Energy cuts may be performed using the 'MDC' option.

```
KWindow=[0.5,0.55]; [ACut, XCut] = Cut(Angle,EFrame,DataFrame,'EDC',KWindow);
```

```
- Data cut formation
```

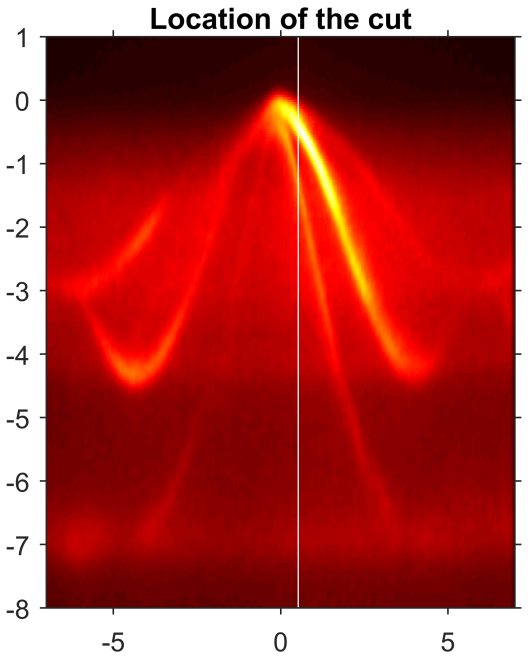
```
figure;subplot(122);plot(ACut, XCut,'r');
title('Constant-angle cut performed using Cut');axis square; xlim(limit)
```

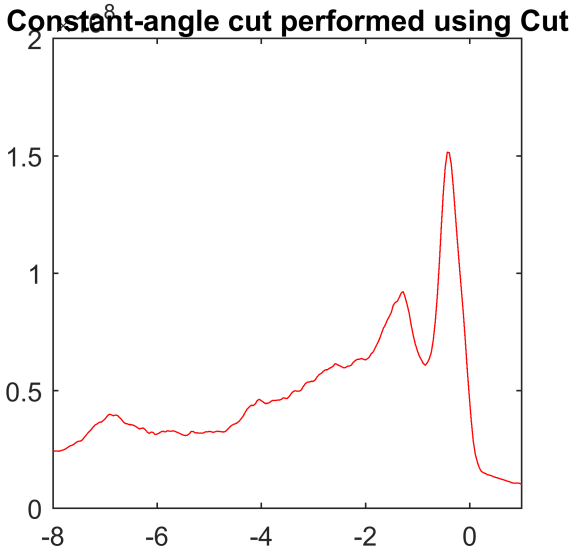
Additional features may be drawn into the *ImData* figure using *hold* as usual. Here, we include the location of the cut in white on our original frame.

```
subplot(121); ImData(Angle,EFrame,DataFrame,'Flat'); ylim(limit)
colormap hot; set(gca,'TickDir','Out'); hold;
```

```
Current plot held
```

```
plot([mean(KWindow),mean(KWindow)], [min(EFrame(:)),max(EFrame(:))],'w-');
title('Location of the cut'); set(gcf, 'Position',[0,0,700,350])
```





Edge Detection

MATools provides two methods of edge detection in the data.

The first, *LaplaceFlt2*, calculates the Laplacian of '2nd' (default) or '4th' order of the data.

```
LpFrame = LaplaceFlt2(DataFrame);

figure; subplot(311); ImData(Angle,EFrame,DataFrame,'Flat');
colormap hot; set(gca,'TickDir','Out');ylim(limit)
title('Original image');

subplot(312);ImData(Angle, EFrame, LpFrame,'Flat');
colormap hot; set(gca,'TickDir','Out');ylim(limit)
title('Edge detection using LaplaceFlt2');
```

The second, *CurvatureFlt2*, calculates the '1D' or '2D' curvature of the data. Note that noise filtering might be necessary both before and after using *CurvatureFlt2* to produce usable results.

```
CrvFrame=GaussFlt2(DataFrame,100,100,500,500);
CrvFrame=SetContrast(CrvFrame,0.4,0.999,1.5);

CrvFrame=CurvatureFlt2(CrvFrame,'2D',1,1);

CrvFrame=GaussFlt2(CrvFrame,1,1,10,10);
CrvFrame=SetContrast(CrvFrame,0.2,0.999);

subplot(313);ImData(Angle, EFrame, CrvFrame,'Flat');
colormap hot; set(gca,'TickDir','Out');ylim(limit);
title('Edge detection using CurvatureFlt2');
```

```
set(gcf, 'Position', [0, 0, 500, 1000])
```