A short introduction to compression with MGARD

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import numpy.linalg as la
```

In [38]:
```
def max_rerror(u0, u):
    return (np.abs(u0-u).max()/np.abs(u0).max())
```

In [39]:
```
def max_error(u0, u):
    return (np.abs(u0-u).max())
```

In [112]:
```
def franke(x1, x2, x3):
    term1 = 0.75 * np.exp(-(9*x1-2)**2/4 - (9*x2-2)**2/4);
    term2 = 0.75 * np.exp(-(9*x1+1)**2/49 - (9*x2+1)/10);
    term3 = 0.5 * np.exp(-(9*x1-7)**2/4 - (9*x2-3)**2/4);
    term4 = -0.2 *np.exp(-(9*x1-4)**2 - (9*x2-7)**2);

    return ( x3 + term1 + term2 + term3 + term4)
```

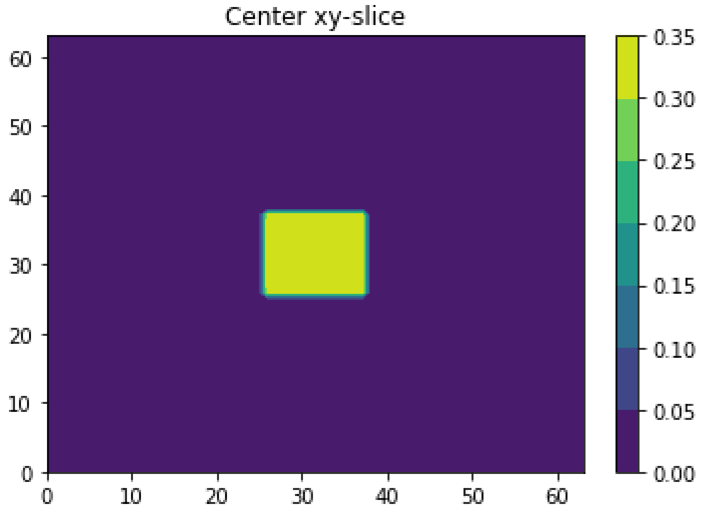
In [5]:
```
data_dir = "/home/user/share/build/tmp/MGARD/"
```

First let's load examine our data file:

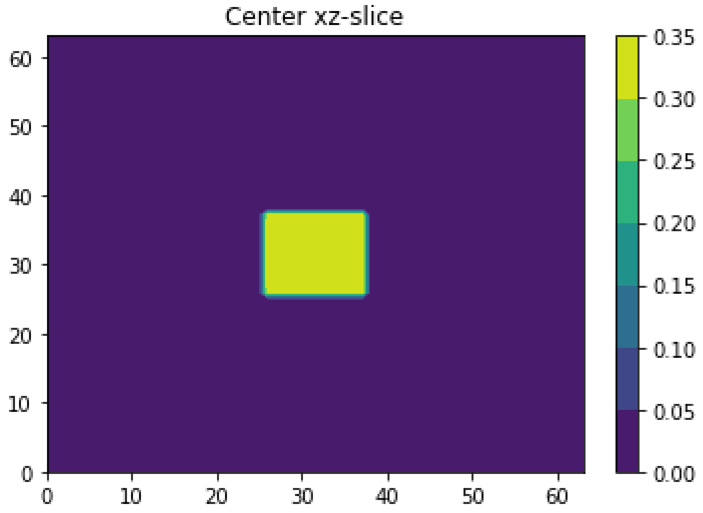
In [20]:
```
data_orig = np.fromfile(data_dir + "data.bin")
```

In [21]:
```
u = data_orig.reshape((64, 64, 64))
```

```
In [13]: plt.title("Center xy-slice")
         plt.contourf(u[:,:,32])
         plt.colorbar()

Out[13]: <matplotlib.colorbar.Colorbar at 0x7fa6109ea6d8>
```

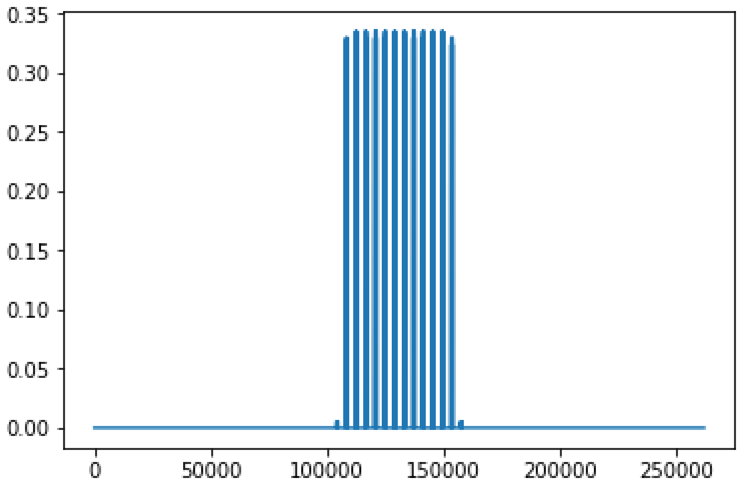


```
In [15]: plt.title("Center xz-slice")
         plt.contourf(u[:,32,:])
         plt.colorbar()

Out[15]: <matplotlib.colorbar.Colorbar at 0x7fa6106c0160>
```



Data consists of concentric cubes, where most of the outer region is equal zero. Whic is evideng from the 1-D plot below

```
In [17]: plt.plot(in_file)

Out[17]: [<matplotlib.lines.Line2D at 0x7fa6105c29b0>]
```



Now let's compress this data while preserving the relative L_∞ error. The relevant call to MGARD compressor in this case will be:

- mgard_compress(iflag, v.data(), out_size, nrow, ncol, nfib, tol);

  ** Here iflag is the data type (0-> float, 1-> double)

  ** v is the input data

  ** nrow, ncol, nfib : Dimension of input

  ** tol: The tolerance in L_∞ norm

  MGARD will return a pointer to the compressed data (unsigned char*) and the output size; out_size

Let's pick a tolerance of 10^{-3} and compress our data, decompress it and see what has happens.

```
In [22]: comp_data = np.fromfile(data_dir + "data_0.001000_infty.dat")
```

```
In [23]: ut_inf_m3 = comp_data.reshape((64,64,64))
```

Let's check the error

```
In [25]: max_error(u, ut_inf_m3)

Out[25]: 0.0005609203671535953
```

It seems the tolerance is met! Good. What if we wanted to preserve the error in derivative, say ∂_x? Then we call MGARD in the following manner:

- mgard_compress(iflag, v.data(), out_size, nrow, ncol, nfib, tol, s); with s = 1

Let's pick a tolerance of 10^{-6} and compress our data, decompress it and see what happens.

```
In [89]: comp_data = np.fromfile(data_dir + "data_1e-7_s1.dat")
```

```
In [90]: ut_s1_m7 = comp_data.reshape((64,64,64))
```

Let's compute the derivatives:

```
In [91]: dux  = np.gradient(u, axis=0)
         dutx = np.gradient(ut_s1_m7, axis=0)
```

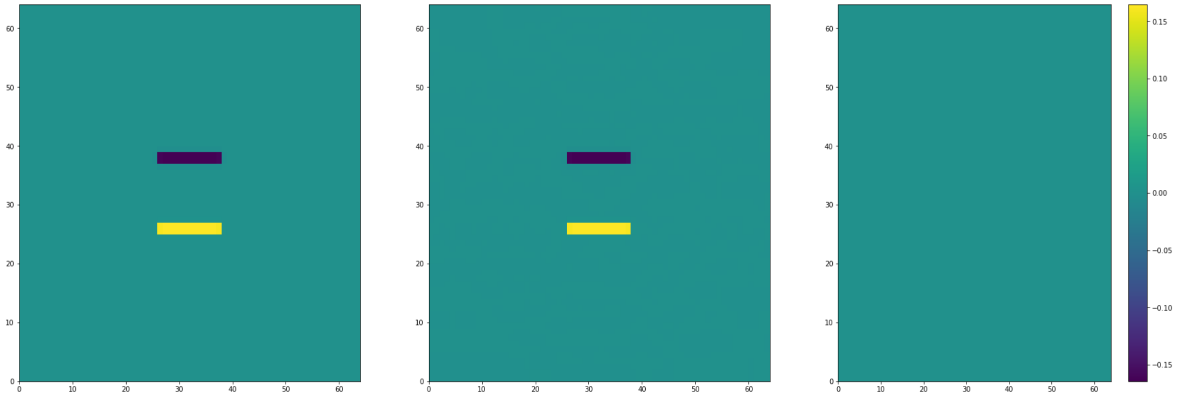
```
In [92]: max_error(dux, dutx)
```

```
Out[92]: 6.847788780461883e-09
```

Good, the tolerance holds! Let's see how the deriative looks like on the center:

In [93]:

```
v_min = dux.min()
v_max = dux.max()

fig = plt.figure(figsize=(30,10));
ax = fig.add_subplot(1,3, 1)
cx = ax.pcolor(dux[:,:,32], cmap='viridis',
                   vmin=v_min, vmax=v_max)

ax = fig.add_subplot(1,3, 2)
cx = ax.pcolor(dutx[:,:,32],cmap='viridis',
                   vmin=v_min, vmax=v_max )

ax = fig.add_subplot(1,3, 3)
cx = ax.pcolor(np.abs(dux[:,:,32] - dutx[:,:,32]),cmap='viridis',
                   vmin=v_min, vmax=v_max)

plt.colorbar(cx);
```



In [116]:

```
x = np.linspace(-1,1,71)
x1, x2, x3 = np.meshgrid(x,x,x)
ufz = franke(x1, x2, x3)
```

In [122]:

```
ufz.tofile(data_dir + "franke3.bin")
```

In [173]:

```
ff = np.fromfile(data_dir + "franke3_0.010000_infty.dat")
```

L_∞ compression, $\tau = 10^{-2}$

In [174]:

```
utfz = ff.reshape((71, 71, 71))
```

In [182]:

```
dufz  = np.gradient(ufz, axis=0)
dutfz = np.gradient(utfz, axis=0)
```

In [183]:

```
max_rerror(np.gradient(ufz, axis=0), np.gradient(utfz, axis=0))
```

Out[183]:

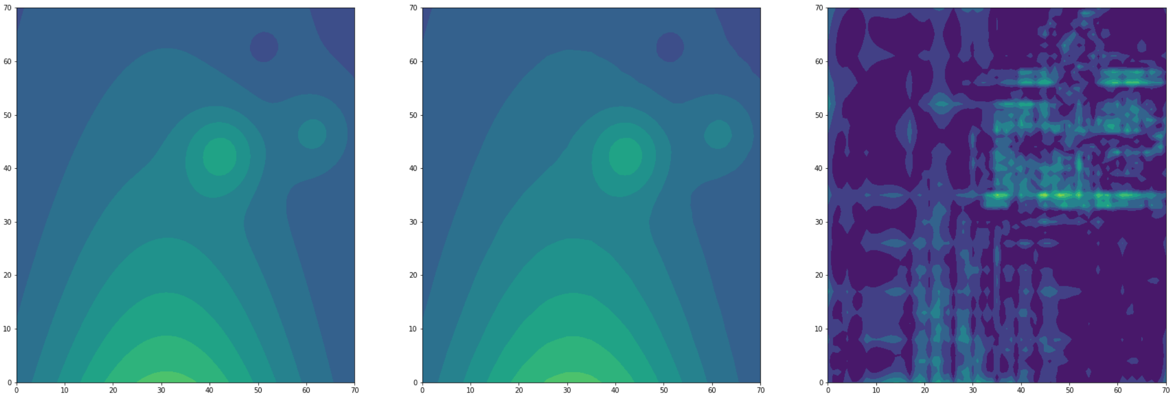
```
0.07376693520536426
```

In [188]:

```
v_min = ufz.min()
v_max = ufz.max()

fig = plt.figure(figsize=(30,10));
ax = fig.add_subplot(1,3, 1)
cx = ax.contourf(ufz[:,:,32], cmap='viridis',
                   vmin=v_min, vmax=v_max)

ax = fig.add_subplot(1,3, 2)
cx = ax.contourf(utfz[:,:,32],cmap='viridis',
                   vmin=v_min, vmax=v_max )

ax = fig.add_subplot(1,3, 3)
ax.contourf(np.abs(ufz[:,:,32] - utfz[:,:,32]),cmap='viridis')

#plt.colorbar(cx);
```

Out[188]:

```
<matplotlib.contour.QuadContourSet at 0x7fa5ff72fa90>
```

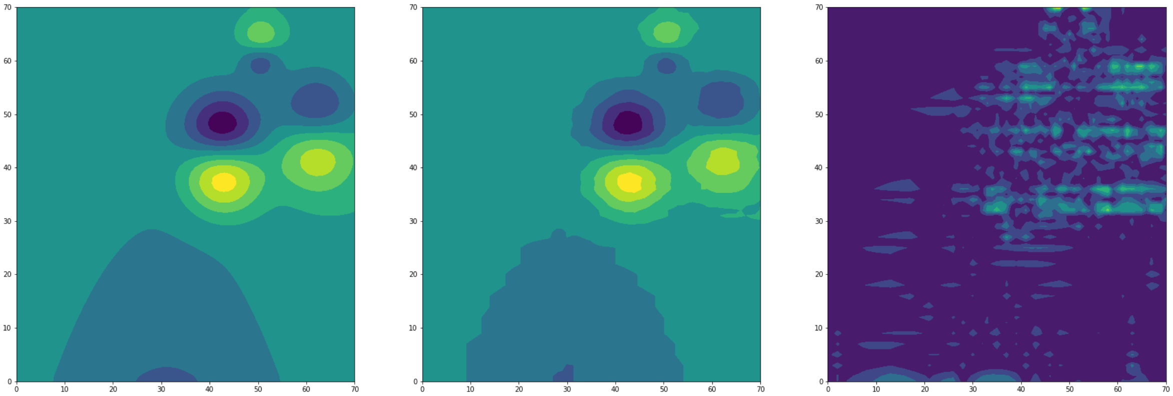


In [190]:

```
v_min = dufz.min()
v_max = dufz.max()

fig = plt.figure(figsize=(30,10));
ax = fig.add_subplot(1,3, 1)
cx = ax.contourf(dufz[:,:,32], cmap='viridis', vmin=v_min, vmax=v_max
)

ax = fig.add_subplot(1,3, 2)
cx = ax.contourf(dutfz[:,:,32],cmap='viridis', vmin=v_min, vmax=v_max
)

ax = fig.add_subplot(1,3, 3)
ax.contourf(np.abs(dufz[:,:,32] - dutfz[:,:,32]),cmap='viridis')

#plt.colorbar(cx);
```

Out[190]:

```
<matplotlib.contour.QuadContourSet at 0x7fa5ff98a3c8>
```



S1 compression, $\tau = 10^{-2}$

In [193]:

```
fs1 = np.fromfile(data_dir + "franke3_0.010000_s1.dat")
```

In [194]:

```
utfzs1 = fs1.reshape((71, 71, 71))
```

In [198]:

```
dutfzs1 = np.gradient(utfzs1, axis=0)
```

In [195]:

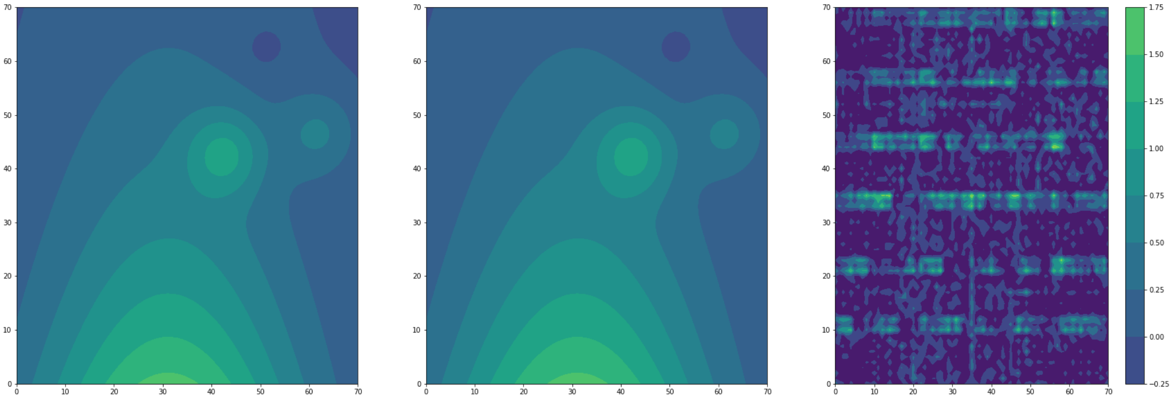
```
max_rerror(np.gradient(ufz, axis=0), np.gradient(utfzs1, axis=0))
```

Out[195]:

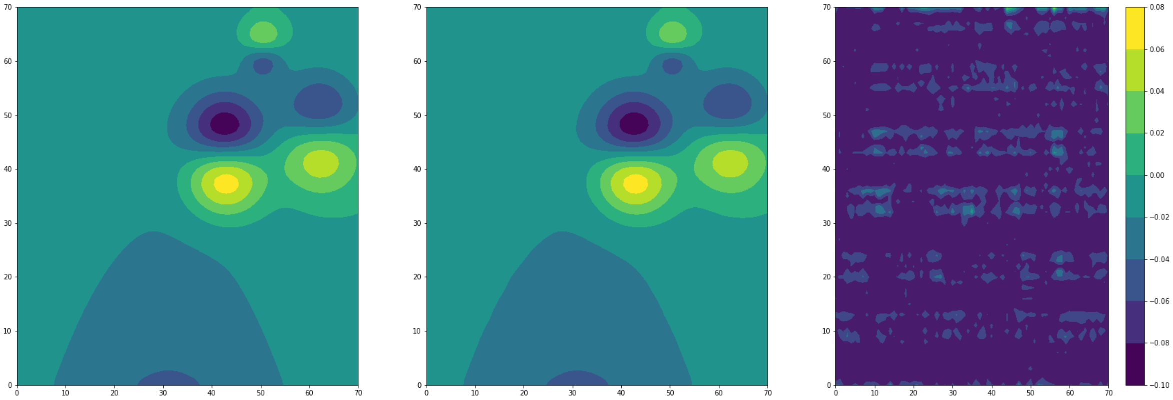
```
0.005602861419883853
```

In [197]:

```
v_min = ufz.min()
v_max = ufz.max()

fig = plt.figure(figsize=(30,10));
ax = fig.add_subplot(1,3, 1)
cx = ax.contourf(ufz[:,:,32], cmap='viridis',
                   vmin=v_min, vmax=v_max)

ax = fig.add_subplot(1,3, 2)
cx = ax.contourf(utfzs1[:,:,32],cmap='viridis',
                   vmin=v_min, vmax=v_max )

ax = fig.add_subplot(1,3, 3)
ax.contourf(np.abs(ufz[:,:,32] - utfzs1[:,:,32]),cmap='viridis',
                  )

plt.colorbar(cx);
```



In [200]:
```
v_min = dufz.min()
v_max = dufz.max()

fig = plt.figure(figsize=(30,10));
ax = fig.add_subplot(1,3, 1)
cx = ax.contourf(dufz[:,:,32], cmap='viridis',
                   vmin=v_min, vmax=v_max)

ax = fig.add_subplot(1,3, 2)
cx = ax.contourf(dutfzs1[:,:,32],cmap='viridis',
                   vmin=v_min, vmax=v_max )

ax = fig.add_subplot(1,3, 3)
ax.contourf(np.abs(dufz[:,:,32] - dutfzs1[:,:,32]),cmap='viridis',
                   )

plt.colorbar(cx);
```



In [ ]:
```
```