In [1]:
```
import h5py
import openmc
import matplotlib.pyplot as plt
import numpy as np
%matplotlib notebook
```

In [2]:
```
# Load the statepoint file
sp_file = 'radial_sanscntrl_HastX_'
sp = openmc.StatePoint( sp_file + 'H.sp.h5')

tally = sp.get_tally(name='spatial flux')
print(tally)

heat = tally.get_slice(scores=['heating-local'])
print(heat)
```

```
Tally
        ID              =       1
        Name            =       spatial flux
        Filters         =       MeshFilter
        Nuclides        =       total
        Scores          =       ['heating-local']
        Estimator       =       tracklength
Tally
        ID              =       2
        Name            =       spatial flux
        Filters         =       MeshFilter
        Nuclides        =       total
        Scores          =       ['heating-local']
        Estimator       =       tracklength
```

In [3]:
```
H = float(np.mean(heat.mean)) #eV/source
H_prime = 1.60218E-19*H #J/source
num_plates = 90
dem = num_plates/2*4
P = 45.0E6/dem #W; total power of reactor slice
f = P/H_prime #source/s

print(f)
```

```
1.827399052436442e+16
```

In [4]:
```
sp = openmc.StatePoint( sp_file + 'F.sp.h5')
tally = sp.get_tally(scores=['flux'])
print(tally)

flux = tally.get_slice(scores=['flux'])
fission = tally.get_slice(scores=['fission'])
print(heat)
```

```
Tally
        ID              =       1
        Name            =       spatial flux
        Filters         =       MeshFilter
        Nuclides        =       total
        Scores          =       ['flux', 'fission', 'heating']
        Estimator       =       tracklength
Tally
        ID              =       2
        Name            =       spatial flux
        Filters         =       MeshFilter
        Nuclides        =       total
        Scores          =       ['heating-local']
        Estimator       =       tracklength
```

In [5]:
```
#restructure array
numcells_x = 500
numcells_y = 500
numcells_z = 30

heat_array = np.ones((numcells_x, numcells_y, numcells_z))
flux_array = np.ones((numcells_x, numcells_y, numcells_z))
fission_array = np.ones((numcells_x, numcells_y, numcells_z))
i = j = k = 0
for n in range(len(flux.mean)):
    flux_array[i][j][k] = float(flux.mean[n])
    fission_array[i][j][k] = float(fission.mean[n])
    i += 1
    if i >= numcells_x:
        i = 0
        j += 1
    if j >= numcells_y:
        j = 0
        k += 1
    if k >= numcells_z:
        k = 0
print('done')
```

```
done
```

In [6]:
```
#re-organize dimensions (x, y, z) -> (z, y, x)
flux_array_new = np.ones((numcells_z, numcells_y, numcells_x))
fission_array_new = np.ones((numcells_z, numcells_y, numcells_x))
for i in range(len(flux_array)):
    for j in range(len(flux_array[i])):
        for k in range(len(flux_array[i][j])):
            flux_array_new[k][j][i] = flux_array[i][j][k] #converted from particle-cm/source to particle/(cm^2*s)
            fission_array_new[k][j][i] = fission_array[i][j][k]
print('done')
```

```
done
```

In [7]:
```
vol_cell = (80.111/numcells_x)*(80.111/numcells_y)*(5.2/numcells_z) #cm^3/cell
P_fission = 3.171E-11 #J/fission_U233

flux_farray = np.ones((numcells_z, numcells_y, numcells_x))
power_farray = np.ones((numcells_z, numcells_y, numcells_x))
for n in range(len(flux_array_new)):
    for m in range(len(flux_array_new[n])):
        for p in range(len(flux_array_new[n][m])):
            flux_farray[n][m][p] = f*flux_array_new[n][m][p]/vol_cell #particle/(cm^2*s)
            power_farray[n][m][p] = f*fission_array_new[n][m][p]*P_fission/vol_cell #J/s/cm^3
print('done')
```

```
done
```

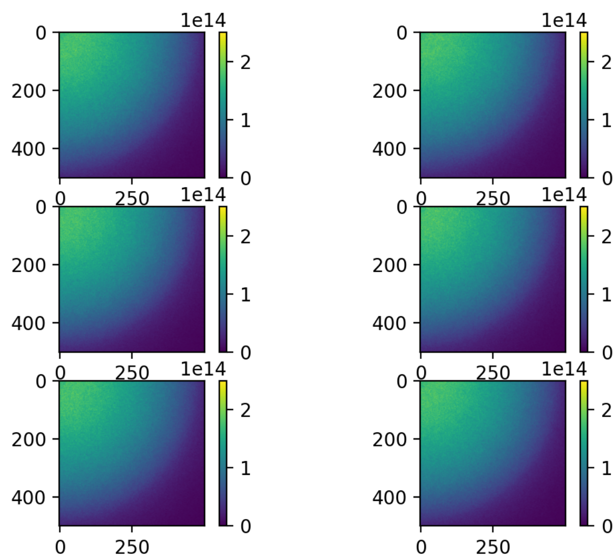
In [8]:
```
print(sp_file)
print(np.mean(flux_farray), 'particles/(cm^2*s)')
print(np.mean(fission_array_new), 'fissions/source')
print(np.mean(power_farray), 'W/cm^3')
print('flux/source',np.mean(flux_array_new), 'particle-cm/source')
```

```
radial_sanscntrl_HastX_
81322604347261.19 particles/(cm^2*s)
5.692323567019851e-08 fissions/source
7.412980025246132 W/cm^3
flux/source 1.9801781732634628e-05 particle-cm/source
```

In [37]:

```
#true flux
fig11 = plt.subplot(321)
im11 = fig11.imshow(flux_farray[0],vmin = 0,vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im11)
fig21 = plt.subplot(322)
im21 = fig21.imshow(flux_farray[5],vmin = 0, vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im21)
fig31 = plt.subplot(323)
im31 = fig31.imshow(flux_farray[13],vmin = 0, vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im31)
fig41 = plt.subplot(324)
im41 = fig41.imshow(flux_farray[15],vmin = 0, vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im41)
fig51 = plt.subplot(325)
im51 = fig51.imshow(flux_farray[17],vmin = 0, vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im51)
fig61 = plt.subplot(326)
im61 = fig61.imshow(flux_farray[29],vmin = 0, vmax = 2.5e14) #particle/(cm^2*s)
plt.colorbar(im61)
```

```
<IPython.core.display.Javascript object>
```

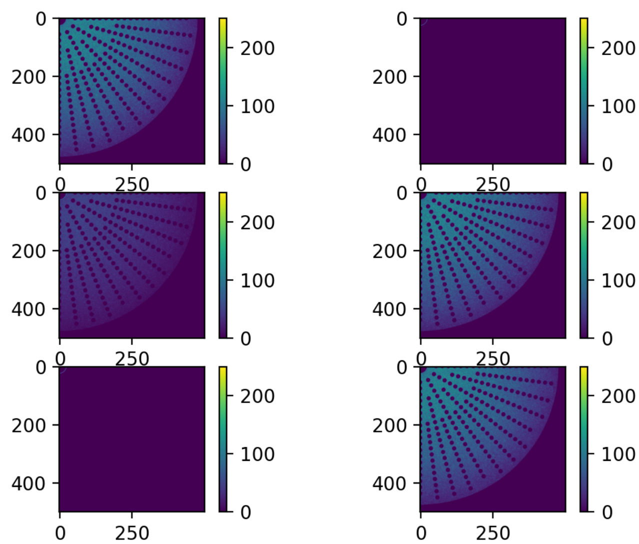


Out[37]: `<matplotlib.colorbar.Colorbar at 0x7f3910914e80>`

In [38]:

```
#power
fig12 = plt.subplot(321)
im12 = fig12.imshow(power_farray[0], vmin=0,vmax=250) #W
plt.colorbar(im12)
fig22 = plt.subplot(322)
im22 = fig22.imshow(power_farray[5],vmin=0,vmax=250) #W
plt.colorbar(im22)
fig32 = plt.subplot(323)
im32 = fig32.imshow(power_farray[13],vmin=0,vmax=250) #W
plt.colorbar(im32)
fig42 = plt.subplot(324)
im42 = fig42.imshow(power_farray[15],vmin=0,vmax=250) #W
plt.colorbar(im42)
fig52 = plt.subplot(325)
im52 = fig52.imshow(power_farray[17],vmin=0,vmax=250) #W
plt.colorbar(im52)
fig62 = plt.subplot(326)
im62 = fig62.imshow(power_farray[29],vmin=0,vmax=250) #W
plt.colorbar(im62)
```

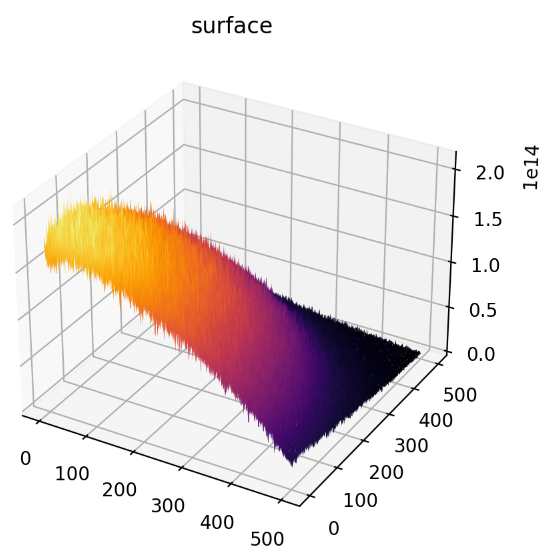
<IPython.core.display.Javascript object>



Out[38]: <matplotlib.colorbar.Colorbar at 0x7f391e387668>

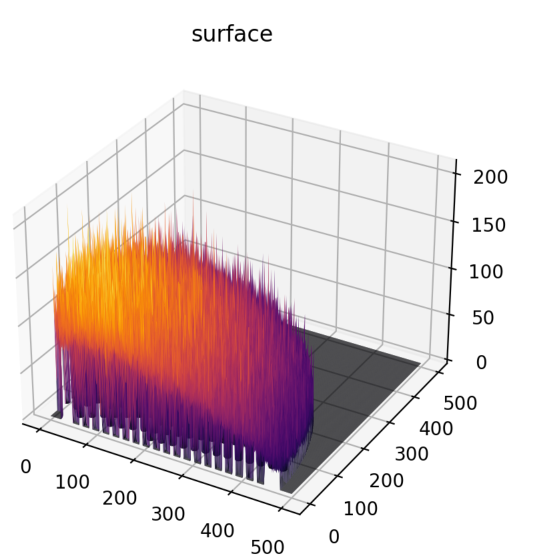
In [23]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

x = np.linspace(0, 500, 500)
y = np.linspace(0, 500, 500)

X, Y = np.meshgrid(x, y)
Z = flux_farray[15]
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1
                ,
                cmap='inferno', edgecolor='none')
ax.set_title('surface');
```

```
<IPython.core.display.Javascript object>
```



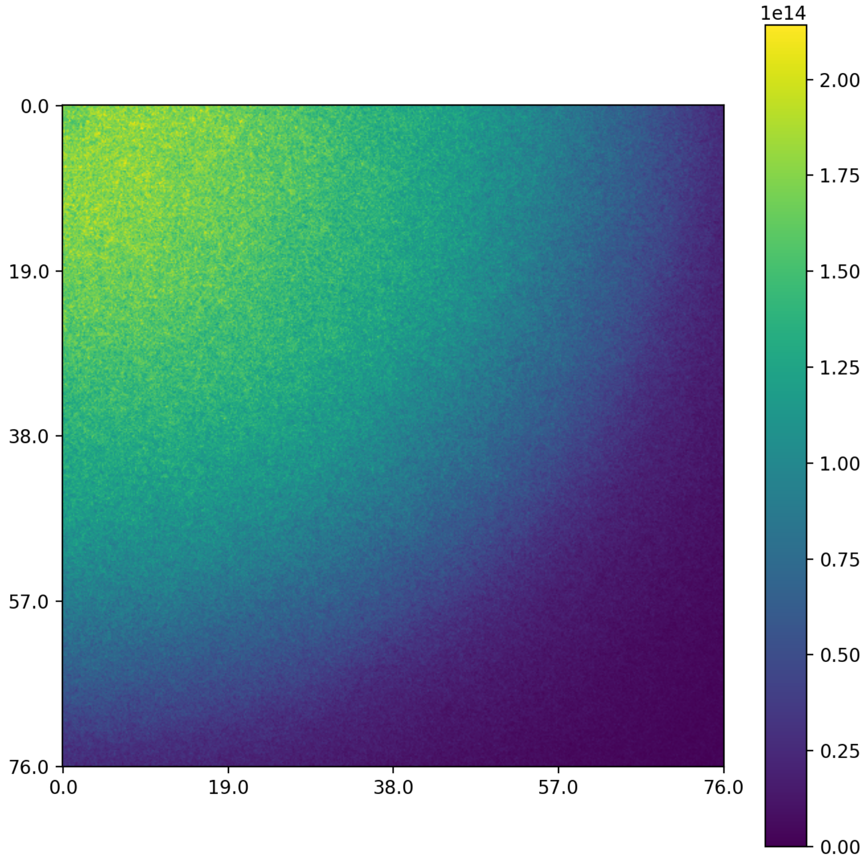
In [24]:
```
PP = power_farray[15]
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, PP, rstride=1, cstride=1
                ,
                cmap='inferno', edgecolor='none')
ax.set_title('surface');
```

```
<IPython.core.display.Javascript object>
```



In [45]:
```
plt.figure(figsize=(8,8))
x = np.linspace(0,500,5)
y = x * 76/500
fig1 = plt.subplot(111)
im1 = fig1.imshow(flux_farray[15], vmin = 0) #vmax = 2.0e14) #particle/(cm^2*s)
plt.colorbar(im1)
plt.xticks(x,y)
plt.yticks(x,y)
plt.show()
```

```
<IPython.core.display.Javascript object>
```



In [28]:
```
#peaking factors
#axial peaking factor
fluxmean = 0
fluxpeak = 0
for i in range(len(flux_farray)-1):
    temp_mean = np.mean(flux_farray[i])
    fluxmean += temp_mean/(len(flux_farray)-1)
    if(temp_mean > fluxpeak):
        fluxpeak = temp_mean
print(fluxmean)
print(fluxpeak)
axial_peak = fluxpeak/fluxmean
print(axial_peak)
```

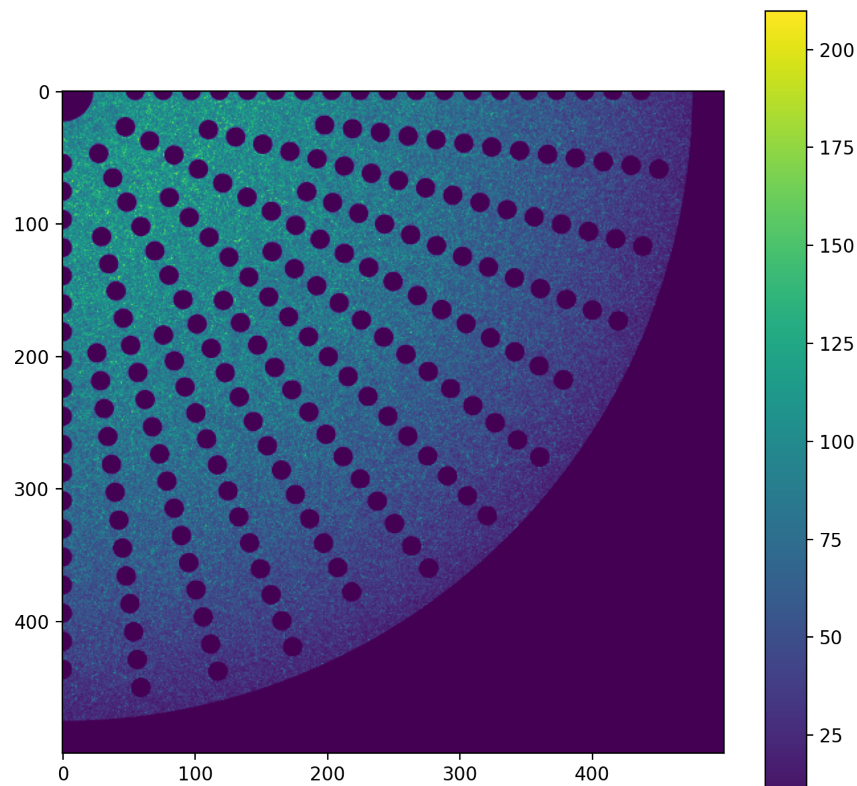
```
81317202139970.45
81563207683046.06
1.0030252583291315
```

In [30]:
```
#radial peaking factor
fluxmean = 0
fluxpeak = 0
for i in range(500):
    tempmean = np.mean(flux_farray[:-1,0,i])
    fluxmean += tempmean/500
    if tempmean > fluxpeak:
        fluxpeak = tempmean
print(fluxmean)
print(fluxpeak)
radial_peak = fluxpeak/fluxmean
print(radial_peak)
```

```
124291942601183.06
179969896434692.4
1.4479610879698277
```

In [33]:
```
#power
plt.figure(figsize=(8,8))
fig1 = plt.subplot(111)
im1 = fig1.imshow(power_farray[15]) #W
plt.colorbar(im1)
```

```
<IPython.core.display.Javascript object>
```



Out[33]:
```
<matplotlib.colorbar.Colorbar at 0x7f391e3f03c8>
```

In [ ]: