c3_radial_sens_2d

October 13, 2023

[1]:
```
import numpy as np
import serpentTools as st
from utilitities import*
from numpy.linalg import norm
import matplotlib.image as mpimg
```

[2]:
```
coreresFile = '/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_coreres.
 ↪main_res.m'
coreres = st.read(coreresFile)

ringresFile = '/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_ringres.
 ↪main_res.m'
ringres = st.read(ringresFile)

elemresFile = '/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_elemres.
 ↪main_res.m'
elemres = st.read(elemresFile)
```

```
SERPENT Serpent 2.2.1 found in
/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_coreres.main_res.m, but
version 2.1.31 is defined in settings
  Attemping to read anyway. Please report strange behaviors/failures to
developers.
SERPENT Serpent 2.2.1 found in
/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_ringres.main_res.m, but
version 2.1.31 is defined in settings
  Attemping to read anyway. Please report strange behaviors/failures to
developers.
SERPENT Serpent 2.2.1 found in
/Users/isaacnaupaaguirre/Downloads/s82d_ac_c3_gcu_elemres.main_res.m, but
version 2.1.31 is defined in settings
  Attemping to read anyway. Please report strange behaviors/failures to
developers.
```

[3]:
```
ref2DFile = '/Users/isaacnaupaaguirre/Downloads/s82d_c3_18G_default70_1.
 ↪main_res.m'
ref2Dres = st.read(ref2DFile)
```

```
SERPENT Serpent 2.1.32 found in
/Users/isaacnaupaaguirre/Downloads/s82d_c3_18G_default70_1.main_res.m, but
version 2.1.31 is defined in settings
  Attemping to read anyway. Please report strange behaviors/failures to
developers.
```

0.0.1 System Analysis

[4]:
```
coreUni = coreres.universes['100', 0, 0, 0]
rootUni = coreres.universes['0', 0, 0, 0]
```

[5]:
```
# rootUni.infExp.keys()
```

[6]:
```
# ax = rootUni.plot('infTot', labels=['infAbs - system'])
# ax.grid()
# coreUni.plot('infTot', ax=ax, labels = ['infAbs - core'], legend='right')
```

[7]:
```
def condense(universe, key, useInvFlux = False):
    cond = None
    if useInvFlux:
        invFlux = np.zeros(len(universe.infExp[key]))
        for i in range(0, len(invFlux)):
            invFlux[i] = 1/universe.infExp[key][i]
        cond = np.sum(np.multiply(universe.infExp[key], invFlux)/np.
 ↪sum(invFlux))
    else:
        cond = np.sum(np.multiply(universe.infExp[key], universe.
 ↪infExp['infFlx']))/np.sum(universe.infExp['infFlx'])
    return cond
```

[8]:
```
fluxweight_coreTranspxs = condense(coreUni, 'infTranspxs')
invfluxweight_coreTranspxs = condense(coreUni, 'infTranspxs', useInvFlux=True)

fluxweight_coreDiff = 1/(3*fluxweight_coreTranspxs)
invfluxweight_coreDiff = 1/(3*invfluxweight_coreTranspxs)

print("fluxweighted infTranspxs : {:.3f}".format(fluxweight_coreTranspxs))
print("invfluxweighted infTranspxs : {:.3f}".format(invfluxweight_coreTranspxs))

print("fluxweighted infDiff : {:.3f}".format(fluxweight_coreDiff))
print("invfluxweighted infDiff : {:.3f}".format(invfluxweight_coreDiff))

coreAbs = condense(coreUni, 'infAbs')

fluxweight_coreDiffLen = np.sqrt(fluxweight_coreDiff/coreAbs)
invfluxweight_coreDiffLen = np.sqrt(invfluxweight_coreDiff/coreAbs)
```

```
print("fluxweighted infDiffLen : {:.3f}".format(fluxweight_coreDiffLen))
print("invfluxweighted infDiffLen : {:.3f}".format(invfluxweight_coreDiffLen))
```

```
fluxweighted infTranspxs : 0.487
invfluxweighted infTranspxs : 0.358
fluxweighted infDiff : 0.684
invfluxweighted infDiff : 0.932
fluxweighted infDiffLen : 4.287
invfluxweighted infDiffLen : 5.002
```

0.0.2 Radial GCU Resolution Study 2D

Make Sure Cases are unbiased torwards statistics on few-group data

[9]:
```
maxs = []
max = None
for uni in coreres.universes:
    maxs.append(np.max(coreres.universes[uni].infUnc['infTot']))
print(np.max(maxs))
```

```
0.00327
```

[10]:
```
maxs = []
max = None
for uni in ringres.universes:
    maxs.append(np.max(ringres.universes[uni].infUnc['infTot']))
print(np.max(maxs))
```

```
0.00685
```

[11]:
```
maxs = []
max = None
for uni in elemres.universes:
    maxs.append(np.max(elemres.universes[uni].infUnc['infTot']))
print(np.max(maxs))
```

```
0.00895
```

Keff Comparison

[12]:
```
reffKeff = elemres.resdata['absKeff']
print("Reference Serpent Keff: {:.5}, pcm: {}".
  ↪format(reffKeff[0],reffKeff[1]*1e5))
```

```
Reference Serpent Keff: 1.142, pcm: 4.6
```

[13]:
```
ringResDF = postProcess('/Users/isaacnaupaaguirre/Documents/GitHub/
  ↪SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
  ↪c3_radial_sens_2d/s82d_ac_c3_gcu_ringres_out.csv', isSteady=True)
coreResDF = postProcess('/Users/isaacnaupaaguirre/Documents/GitHub/
  ↪SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
  ↪c3_radial_sens_2d/s82d_ac_c3_gcu_coreres_out.csv', isSteady=True)
```

```
elemResDF = postProcess('/Users/isaacnaupaaguirre/Documents/GitHub/
 ↪SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
 ↪c3_radial_sens_2d/s82d_ac_c3_gcu_elemres_out.csv', isSteady=True)
```

[14]:
```
def kToPCM(k):
    return 1e5*((k-1)/k)
```

[15]:
```
griff_ckeff = coreResDF['eigenvalue'][1]
griff_rkeff = ringResDF['eigenvalue'][1]
griff_ekeff = elemResDF['eigenvalue'][1]

print("core resolution keff: {:.5f}, pcmDiff: {:.1f}".format(griff_ckeff,␣
 ↪kToPCM(griff_ckeff) - kToPCM(reffKeff[0])))
print("ring resolution keff: {:.5f}, pcmDiff: {:.1f}".format(griff_rkeff,␣
 ↪kToPCM(griff_rkeff) - kToPCM(reffKeff[0])))
print("elem resolution keff: {:.5f}, pcmDiff: {:.1f}".format(griff_ekeff,␣
 ↪kToPCM(griff_ekeff) - kToPCM(reffKeff[0])))
```

```
core resolution keff: 1.14162, pcmDiff: -25.8
ring resolution keff: 1.14170, pcmDiff: -20.1
elem resolution keff: 1.14175, pcmDiff: -15.8
```

1. PCM difference decreases with increase in spatial resolution as espected

Verification of Few Group Spectrum

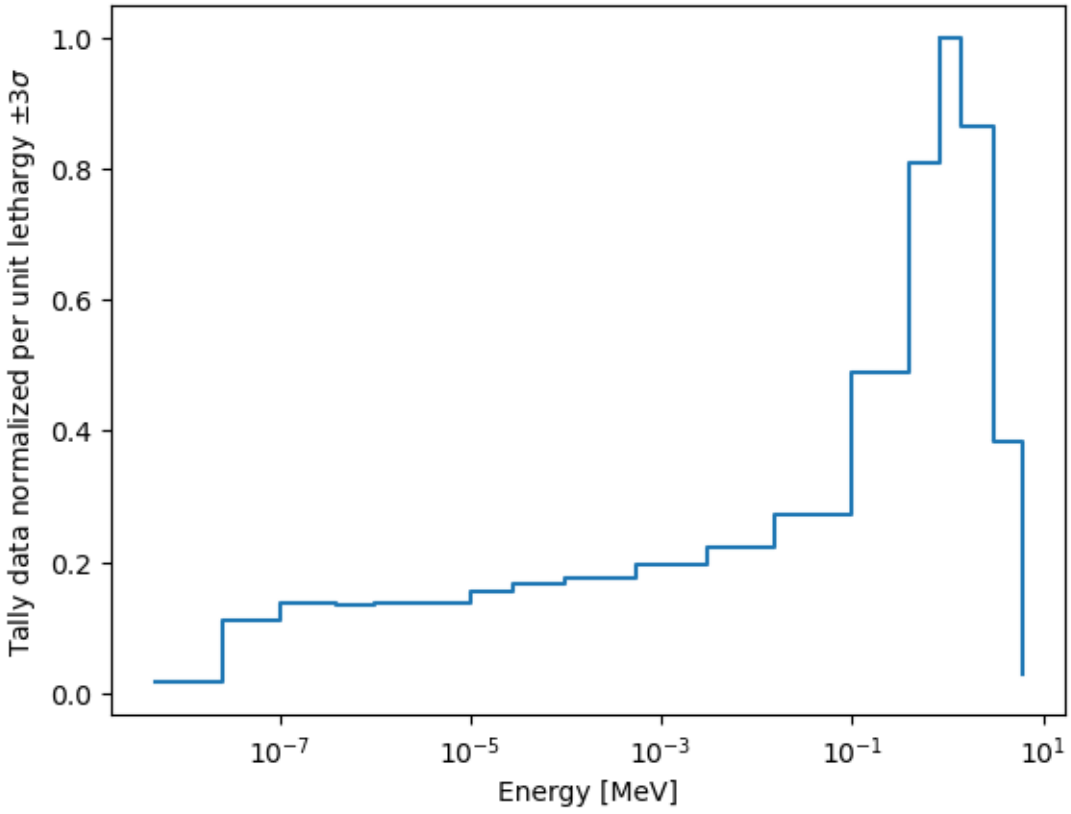
[16]:
```
fgs_hr18 = [5.0000E-09, 2.5000E-08, 1.0000E-07, 4.0000E-07, 9.9600E-07, 3.
 ↪0000E-06,
9.8770E-06, 2.7700E-05, 1.0000E-04, 5.5000E-04, 3.0000E-03, 1.5030E-02,
1.0000E-01, 4.0000E-01, 8.2100E-01, 1.3530E+00, 3.0000E+00, 6.0655E+00,
2.0000E+01]
```

[17]:
```
refFGSFile = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_coreres.main_det0.m'
refFGSDet = st.read(refFGSFile)
refFGS = refFGSDet['fgs_spec']
```

[18]:
```
def normZeroToOne(arr):
    normArr = np.zeros(len(arr))
    min = np.min(arr)
    max = np.max(arr)
    for i in range(0, len(arr)):
        normArr[i] = (arr[i] - min)/(max - min)
    return normArr
```

[19]:
```
refFGS.spectrumPlot()
```

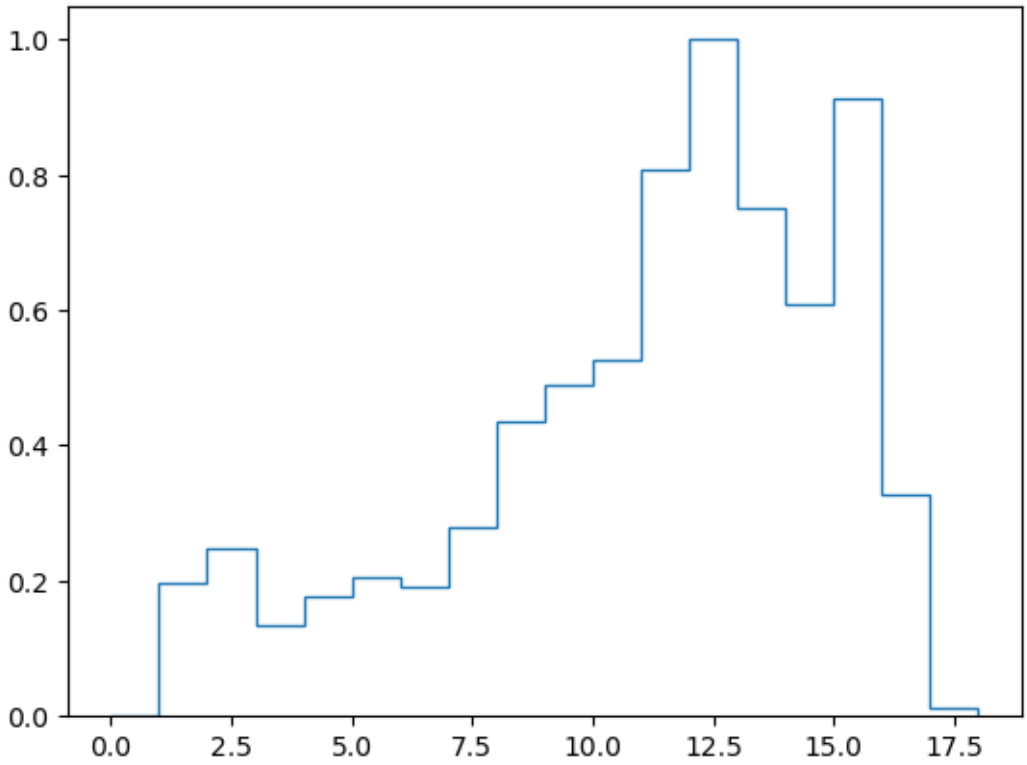
[19]: <AxesSubplot:xlabel='Energy [MeV]', ylabel='Tally data normalized per unit lethargy $\\pm3\\sigma$'>



[20]:
```
normRefFGS = normZeroToOne(refFGS.tallies)
```

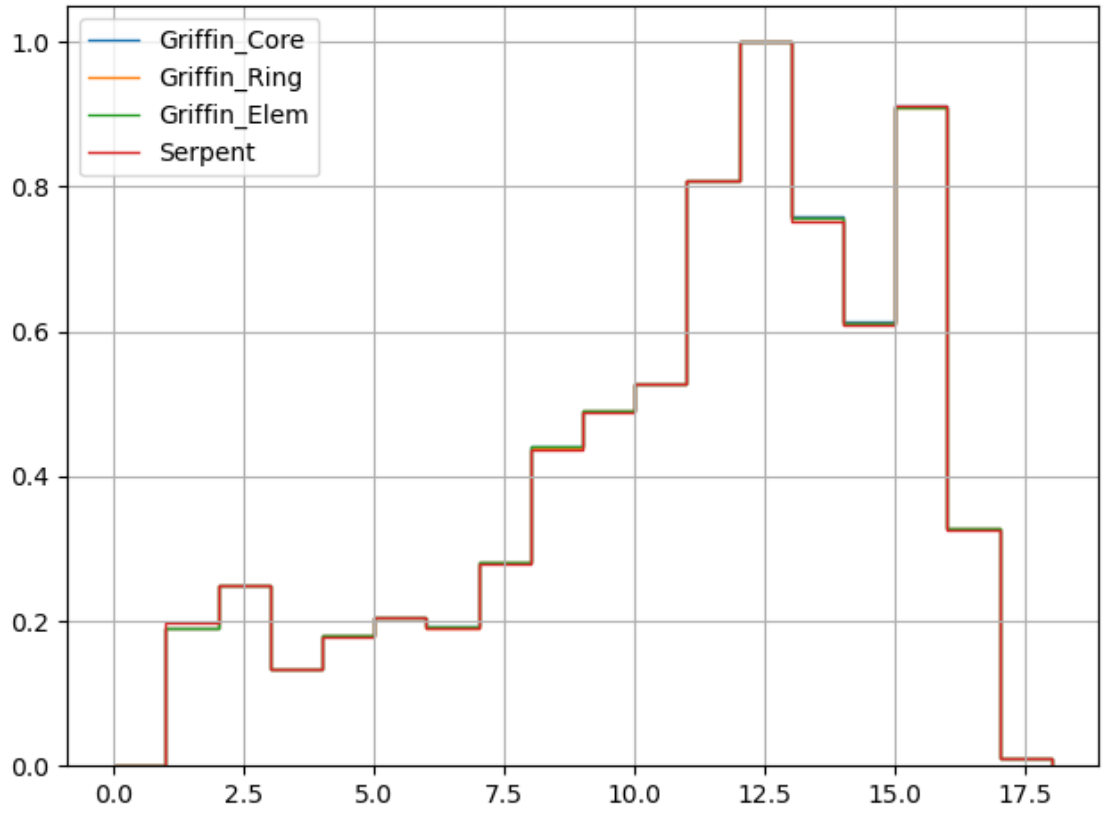
[21]:
```
plt.stairs(normRefFGS)
```

[21]: <matplotlib.patches.StepPatch at 0x7fa7e49bcd30>



[22]:
```
#collectFGS
nFewGroups = 18
griff_coreResFGS = []
griff_ringResFGS = []
griff_elemResFGS = []
for i in range(nFewGroups-1, -1, -1):
    griff_coreResFGS.append(coreResDF['Flux{}'.format(i+1)][1])
    griff_ringResFGS.append(ringResDF['Flux{}'.format(i+1)][1])
    griff_elemResFGS.append(elemResDF['Flux{}'.format(i+1)][1])

normGriff_ringResFGS = normZeroToOne(griff_ringResFGS)
normGriff_coreResFGS = normZeroToOne(griff_coreResFGS)
normGriff_elemResFGS = normZeroToOne(griff_elemResFGS)
```

[23]:
```
plt.stairs(normGriff_coreResFGS, label = "Griffin_Core")
plt.stairs(normGriff_ringResFGS, label = "Griffin_Ring")
plt.stairs(normGriff_elemResFGS, label = "Griffin_Elem")
plt.stairs(normRefFGS, label = "Serpent")
plt.legend(loc='upper left')
plt.tight_layout()
plt.grid()
```



[24]:
```
def calcL2NormDiffPerc(ref,  comp):
    diff = np.subtract(ref, comp)
    diffNorm  = norm(diff)
    base = norm(ref)
    return (diffNorm/base)*100

def calcPercentRelativeError(ref, comp):
    relErr = []
    diff = np.abs(np.subtract(ref, comp))

    for i in range(0, len(diff)):
        if diff[i] != 0:
            #relErr.append(2*(diff[i]/(np.abs(ref[i])+np.abs(comp[i]))))
            relErr.append(100*diff[i]/ref[i])
        else:
            relErr.append(0)
    return relErr
```

[25]:
```
coreL2NormDiffFGS = calcL2NormDiffPerc(normRefFGS, normGriff_coreResFGS)
ringL2NormDiffFGS = calcL2NormDiffPerc(normRefFGS, normGriff_ringResFGS)
elemL2NormDiffFGS = calcL2NormDiffPerc(normRefFGS, normGriff_elemResFGS)
```

[26]:
```
print("core resolution FGS l2normDiff (%): {:.3f}".format(coreL2NormDiffFGS))
print("ring resolution FGS l2normDiff (%): {:.3f}".format(ringL2NormDiffFGS))
print("elem resolution FGS l2normDiff (%): {:.3f}".format(elemL2NormDiffFGS))
```

```
core resolution FGS l2normDiff (%): 0.624
ring resolution FGS l2normDiff (%): 0.607
elem resolution FGS l2normDiff (%): 0.609
```

[27]:
```
def createDetectors(unis):
    detStr = ""
    for i in range(0, len(unis)):
        detStr = detStr + 'det nuFissRate{} dr -7 void du {}\n'.format(unis[i],␣
 ↪unis[i])
        detStr = detStr + 'det capRate{} dr -2 void du {}\n'.format(unis[i],␣
 ↪unis[i])

    return detStr
```

Flux Map Comparison

[28]:
```
coreRefMapFile = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_coreres.main_detMap.m'
coreRefMapDet = st.read(coreRefMapFile, reader='det')

ringRefMapFile = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_ringres.main_det0.m'
ringRefMapDet = st.read(ringRefMapFile, reader='det')

elemRefMapFile = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_elemres.main_det0.m'
elemRefMapDet = st.read(elemRefMapFile, reader='det')
```

[29]:
```
def griffinFluxMapReader(path):
    keys = ['volume', 'nufiss', 'power', 'absorption', 'scalar']
    block = []
    vol = []
    nufiss = []
    pow = []
    abs = []
    scalar = []
    with open(path, "r") as f:
        lines = f.readlines()
        f.close()

    bidx = None
```

```
    eidx = None
    hasBegun = False
    for ldx, line in enumerate(lines):
        if "Block average" in line:
            bidx = ldx+2
            hasBegun = True
        if (line == "\n") & hasBegun:
            eidx = ldx
            break

    data = lines[bidx:eidx]

    dicts = []

    for i in range(0, len(data)):
        vals  = data[i].split()
        block.append(vals[0])
        vol.append(float(vals[1]))
        nufiss.append(float(vals[2]))
        pow.append(float(vals[3]))
        abs.append(float(vals[4]))
        scalar.append(float(vals[5]))

        dset = [vol[i], nufiss[i], pow[i], abs[i], scalar[i]]
        dicts.append(dict(zip(keys, dset)))


    map = dict(zip(block, dicts))
    return map
```

[30]:
```
coreGriffBlockMap = griffinFluxMapReader('/Users/isaacnaupaaguirre/Documents/
 ↪GitHub/SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
 ↪c3_radial_sens_2d/core_flux_map.txt')
ringGriffBlockMap = griffinFluxMapReader('/Users/isaacnaupaaguirre/Documents/
 ↪GitHub/SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
 ↪c3_radial_sens_2d/ring_flux_map.txt')
elemGriffBlockMap = griffinFluxMapReader('/Users/isaacnaupaaguirre/Documents/
 ↪GitHub/SNAP-REACTORS/snapReactors/reactor_models/AutomatedSerpentModels/GCU/
 ↪c3_radial_sens_2d/elem_flux_map.txt')
```

[31]:
```
def getBlock2UniMap(df, blockMap):
    uniMap = {}
    blocks = list(df['Block Name'])
    unis = list(df['material_id'])

    bkeyu = {}
```

```
    for bdx, block in enumerate(blocks):
        bkeyu[block.replace("block_", "")] = str(int(unis[bdx]))

    for block in blockMap:
        uniMap[bkeyu[block]] = blockMap[block]

    return uniMap
```

[32]:
```
def getUni2BlockMap(df, uniMap):
    blockMap = {}

    blocks = list(df['Block Name'])
    unis = list(df['material_id'])

    ukeyb = {}

    for bdx, block in enumerate(blocks):
        ukeyb[str(int(unis[bdx]))] = block.replace("block_", "")

    for uni in uniMap:
        # uniMap[bkeyu[block]] = blockMap[block]

        blockMap[ukeyb[uni]] = uniMap[str(uni)]
    return blockMap
```

[33]:
```
corePointData = pd.read_csv('snapReactors/reactor_models/AutomatedSerpentModels/
 ↪GCU/c3_radial_sens_2d/s82d_ac_c3_gcu_coreres_cellPointdata.csv')
ringPointData = pd.read_csv('snapReactors/reactor_models/AutomatedSerpentModels/
 ↪GCU/c3_radial_sens_2d/s82d_ac_c3_gcu_ringres_cellPointdata.csv')
elemPointData = pd.read_csv('snapReactors/reactor_models/AutomatedSerpentModels/
 ↪GCU/c3_radial_sens_2d/s82d_ac_c3_gcu_elemres_cellPointdata.csv')
```

[34]:
```
coreGriffMap = getBlock2UniMap(corePointData, coreGriffBlockMap)
ringGriffMap = getBlock2UniMap(ringPointData, ringGriffBlockMap)
elemGriffMap = getBlock2UniMap(elemPointData, elemGriffBlockMap)
```

[35]:
```
# print(coreGriffMap)
# print(ringGriffMap)
# print(elemGriffMap)
```

[36]:
```
coreUnis = list(coreGriffMap.keys())
ringUnis = list(ringGriffMap.keys())
elemUnis = list(elemGriffMap.keys())

# print(createDetectors(coreUnis))
# print(createDetectors(ringUnis))
# print(createDetectors(elemUnis))
```

[37]:
```
def getUniValsList(map, attr):
    vals = []
    for uni in map:
        vals.append(map[uni][attr])
    return vals
```

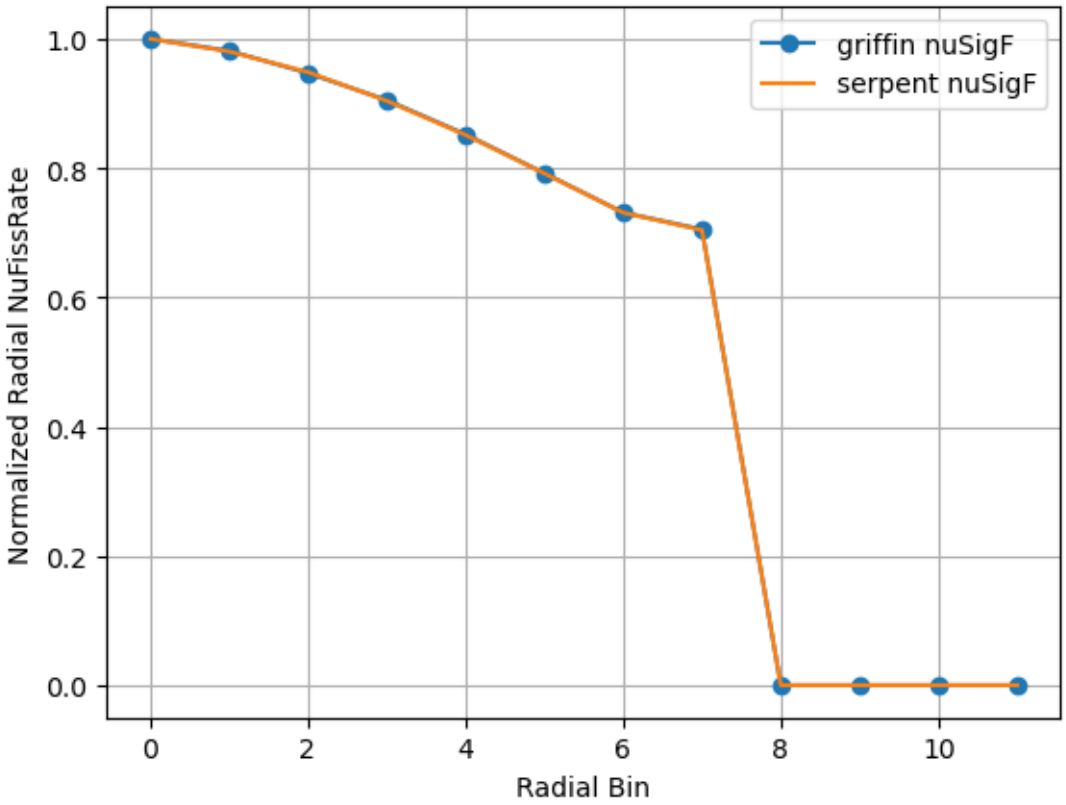
[38]:
```
coreNuFissMap = {}
coreCapMap = {}

coreNuFiss = []
coreCap = []

ringNuFissMap = {}
ringCapMap = {}

ringNuFiss = []
ringCap = []

elemNuFissMap = {}
elemCapMap = {}

elemNuFiss = []
elemCap = []

for uni in coreUnis:
    coreNuFissMap[uni] = coreRefMapDet["nuFissRate{}".format(uni)]
    coreCapMap[uni] = coreRefMapDet["capRate{}".format(uni)]

for uni in ringUnis:
    ringNuFissMap[uni] = ringRefMapDet["nuFissRate{}".format(uni)]
    ringCapMap[uni] = ringRefMapDet["capRate{}".format(uni)]

for uni in elemUnis:
    elemNuFissMap[uni] = elemRefMapDet["nuFissRate{}".format(uni)]
    elemCapMap[uni] = elemRefMapDet["capRate{}".format(uni)]
```

[39]:
```
for uni in coreNuFissMap:
    coreNuFiss.append(coreNuFissMap[uni].tallies/coreGriffMap[uni]['volume'])
    coreCap.append(coreCapMap[uni].tallies/coreGriffMap[uni]['volume'])

for uni in ringNuFissMap:
    ringNuFiss.append(ringNuFissMap[uni].tallies/ringGriffMap[uni]['volume'])
    ringCap.append(ringCapMap[uni].tallies/ringGriffMap[uni]['volume'])

for uni in elemNuFissMap:
    elemNuFiss.append(elemNuFissMap[uni].tallies/elemGriffMap[uni]['volume'])
    elemCap.append(elemCapMap[uni].tallies/elemGriffMap[uni]['volume'])
```

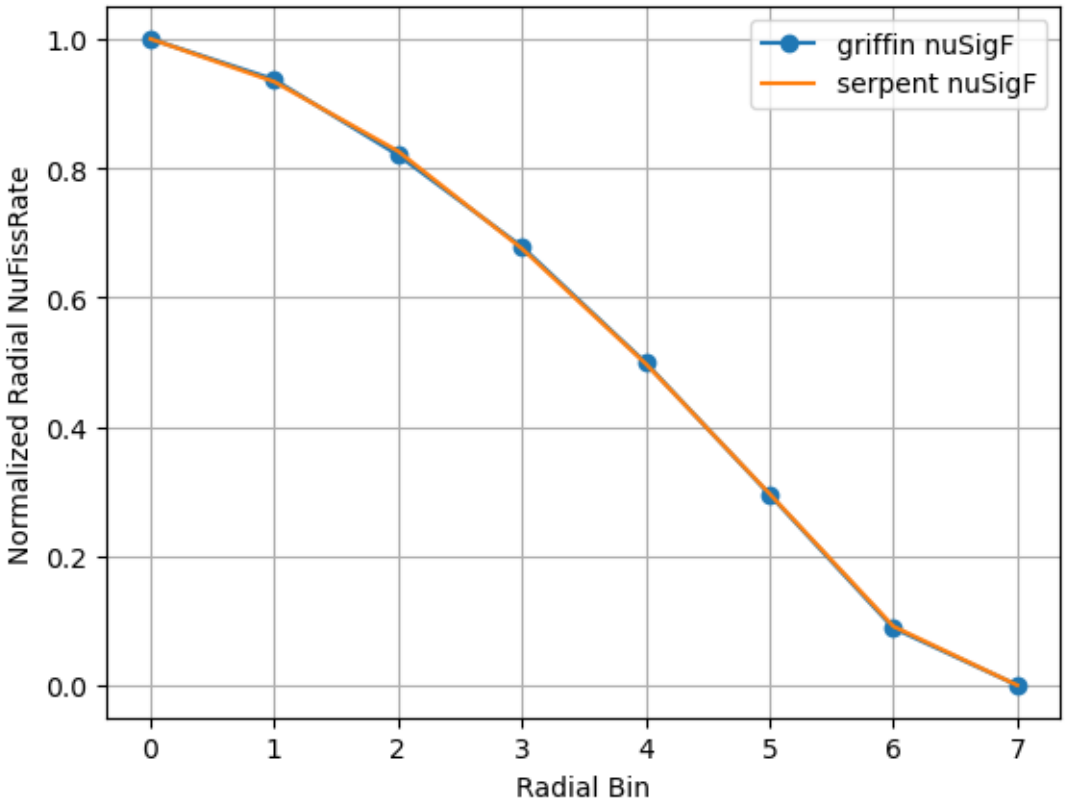
[40]:
```
elemNuFissBlockMap = getUni2BlockMap(elemPointData, elemNuFissMap)
```

[41]:
```
griffCoreNuFiss = getUniValsList(coreGriffMap, 'nufiss')
griffRingNuFiss = getUniValsList(ringGriffMap, 'nufiss')
griffElemNuFiss = getUniValsList(elemGriffMap, 'nufiss')
```

[42]:
```
plt.plot(normZeroToOne(griffRingNuFiss), label = "griffin nuSigF", marker = "o")
plt.plot(normZeroToOne(ringNuFiss), label = "serpent nuSigF")
plt.ylabel("Normalized Radial NuFissRate")
plt.xlabel("Radial Bin")
plt.legend()
plt.grid()
```



[43]:
```
plt.plot(normZeroToOne(griffRingNuFiss[0:8]), label = "griffin nuSigF", marker␣
↪= "o")
plt.plot(normZeroToOne(ringNuFiss[0:8]), label = "serpent nuSigF")
plt.ylabel("Normalized Radial NuFissRate")
plt.xlabel("Radial Bin")
plt.legend()
plt.grid()
```



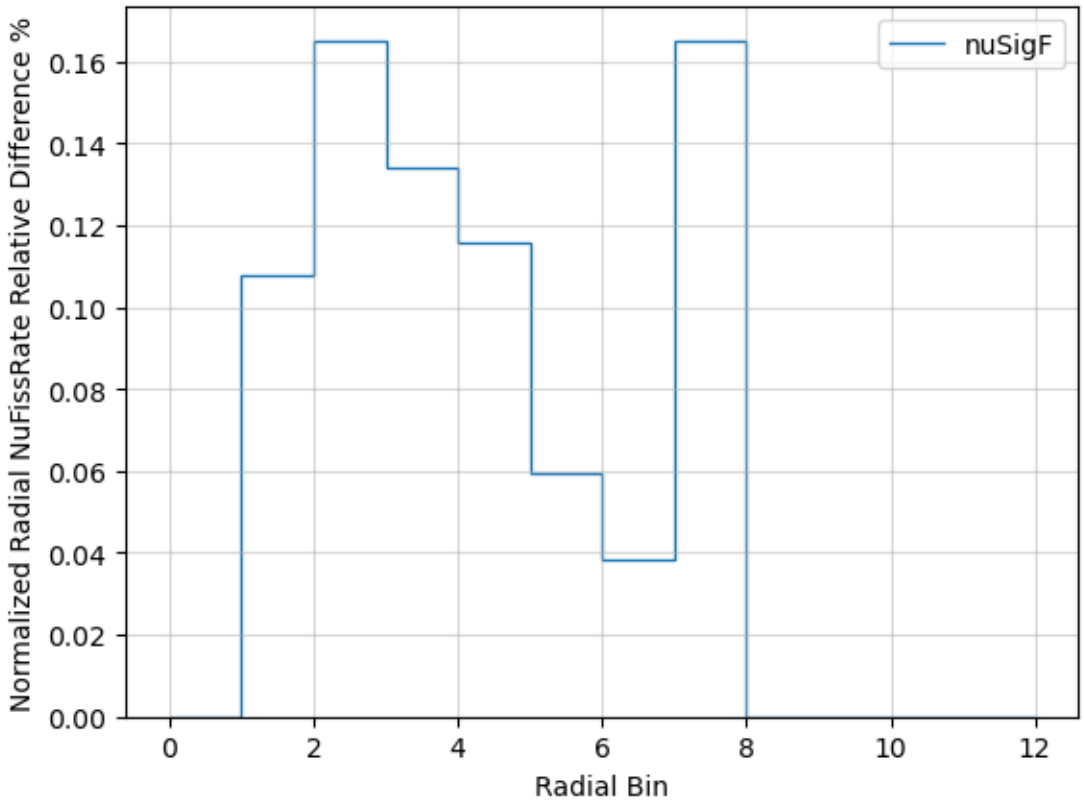
[44]:
```
ringL2NormDiffNuFiss = calcL2NormDiffPerc(normZeroToOne(ringNuFiss),␣
↪normZeroToOne(griffRingNuFiss))
print("ring resolution NuFissRate l2normDiff (%): {:.3f}".
↪format(ringL2NormDiffNuFiss))
```

```
ring resolution NuFissRate l2normDiff (%): 0.112
```

[45]:
```
coreNuFissRateRelError = calcPercentRelativeError(normZeroToOne(coreNuFiss),␣
↪normZeroToOne(griffCoreNuFiss))
coreNuFissRateRelErrorMap = dict(zip(coreUnis, coreNuFissRateRelError))

ringNuFissRateRelError = calcPercentRelativeError(normZeroToOne(ringNuFiss),␣
↪normZeroToOne(griffRingNuFiss))
ringNuFissRateRelErrorMap = dict(zip(ringUnis, ringNuFissRateRelError))

elemNuFissRateRelError = calcPercentRelativeError(normZeroToOne(elemNuFiss),␣
↪normZeroToOne(griffElemNuFiss))
elemNuFissRateRelErrorMap = dict(zip(elemUnis, elemNuFissRateRelError))
```

[46]:
```
plt.stairs(ringNuFissRateRelError , label = "nuSigF", alpha = 1)
plt.ylabel("Normalized Radial NuFissRate Relative Difference %")
```

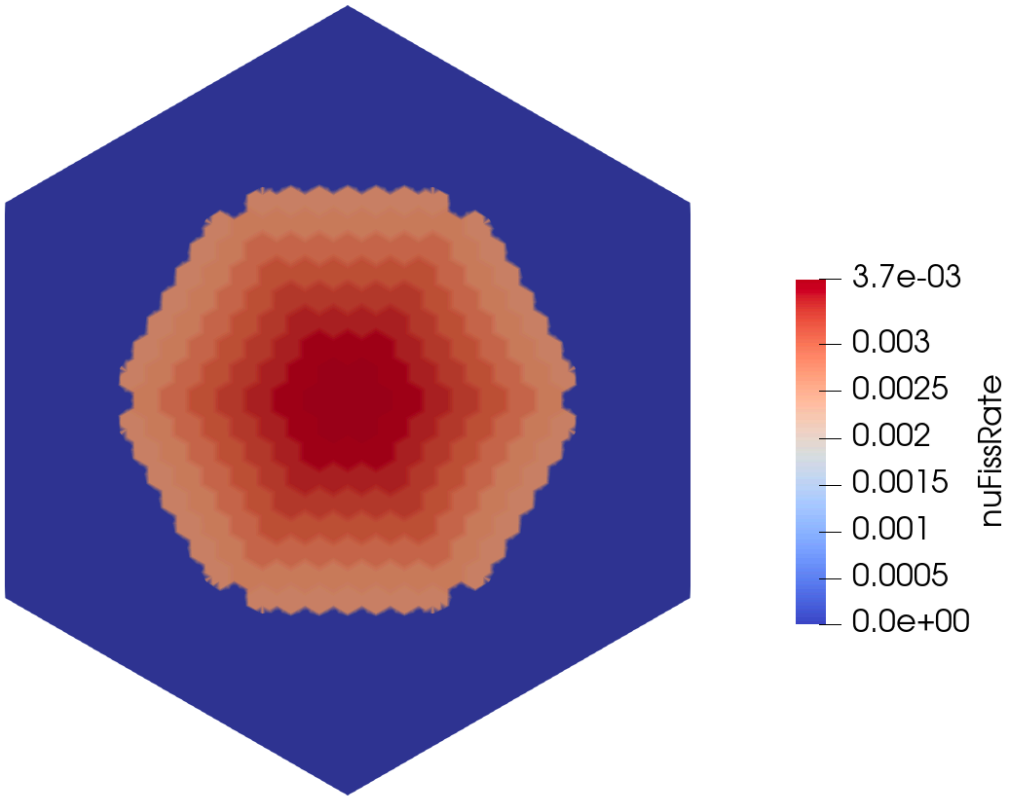
```
plt.xlabel("Radial Bin")
plt.legend()
plt.grid(alpha = 0.5)
```



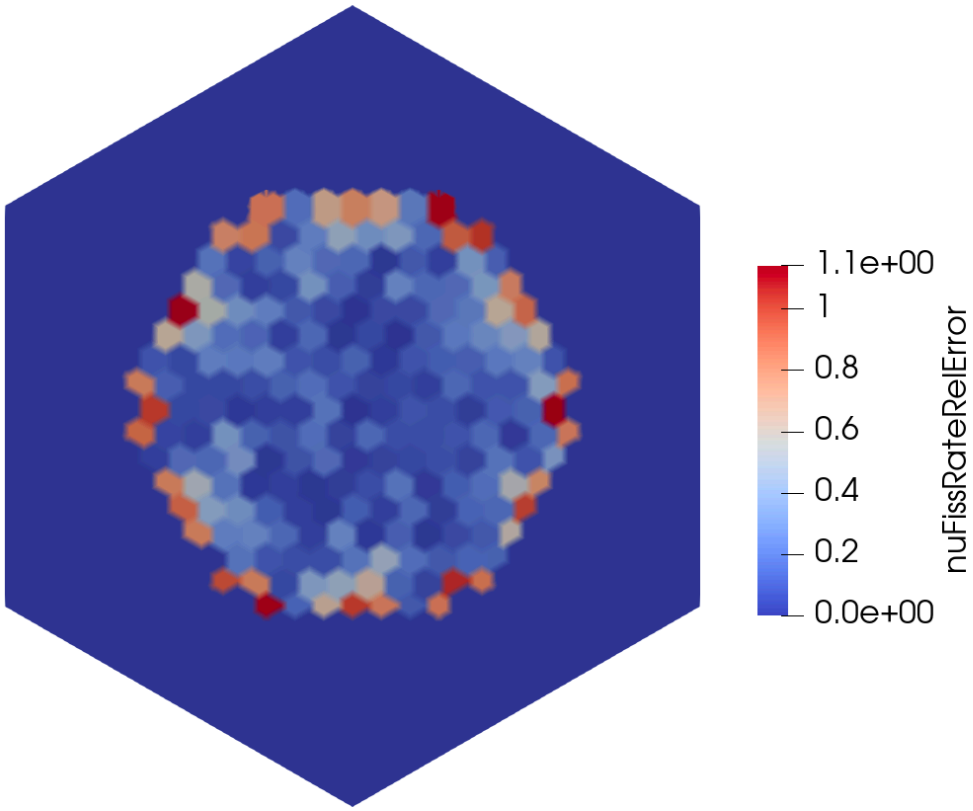
[47]:
```
def createAppendCSV(cellData, keys, params, vals, exportPath, useBlockId =␣
↪False):
    xyz = ['x', 'y', 'z']

    for param in params:
        xyz.append(param)

    appendDF = pd.DataFrame(columns=xyz)

    map = {}

    for key in keys:
        for pdx, param in enumerate(params):
            map[key] = {}

    for kdx, key in enumerate(keys):
        for pdx, param in enumerate(params):
```

```
            map[key][param] = vals[pdx][kdx]


    if not useBlockId:
        pointKeys = np.array(list(cellData['material_id'])).astype('int')
    else:
        pointKeys = []
        blocks = list(cellData['Block Name'])
        for i in range(0, len(blocks)):
            pointKeys.append(int(blocks[i].replace("block_", "")))

    appendDF['x'] = list(cellData['Points_0'])
    appendDF['y'] = list(cellData['Points_1'])
    appendDF['z'] = list(cellData['Points_2'])

    for param in params:
        pointData = []
        for i in range(0, len(pointKeys)):
            pointData.append(map[str(pointKeys[i])][param])

        appendDF[param] = pointData

    appendDF.to_csv(exportPath, index=False)

    return appendDF
```

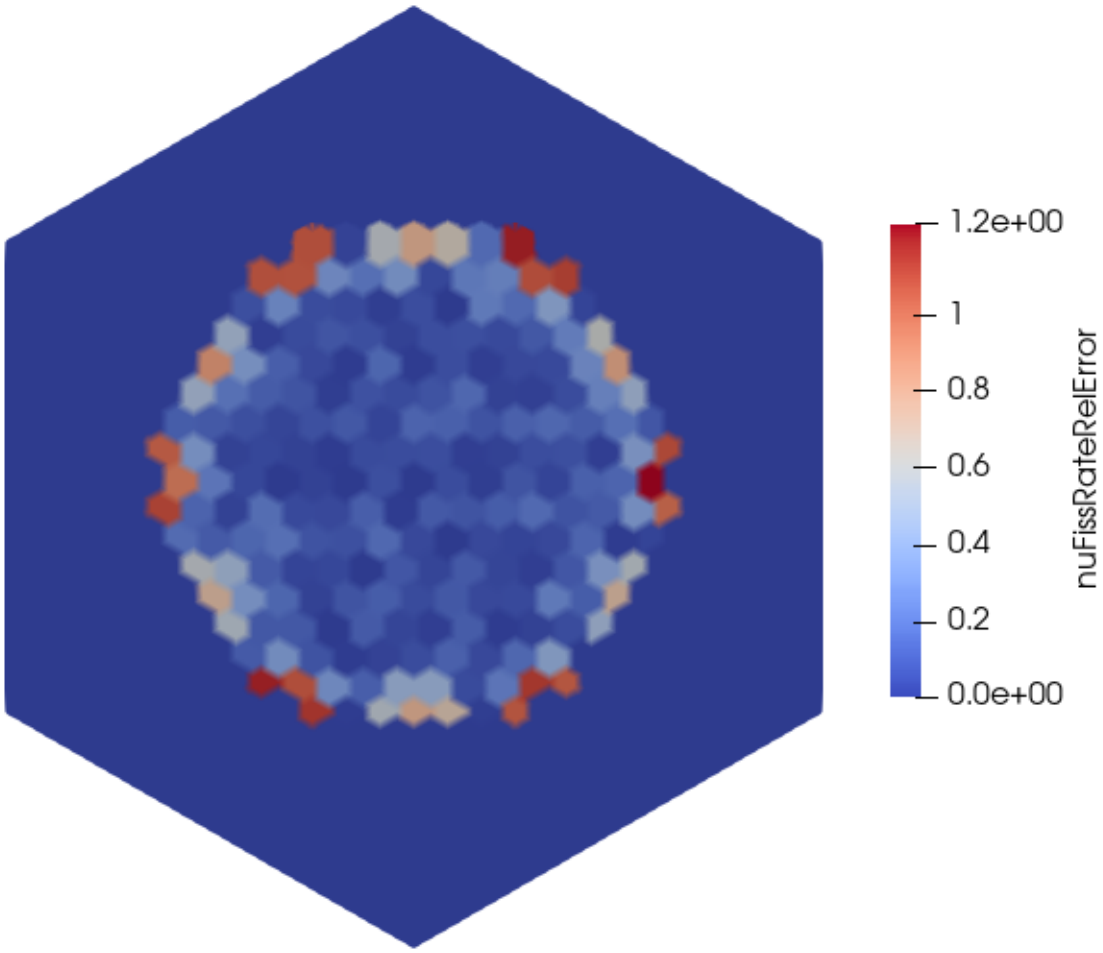
[48]:
```
coreParams = ['nuFissRate', 'nuFissRateRelError']
coreVals = [getUniValsList(coreGriffMap, 'nufiss'), coreNuFissRateRelError]
corePath = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_coreres_cellPointdataAppend.csv'

ringParams = ['nuFissRate', 'nuFissRateRelError']
ringVals = [getUniValsList(ringGriffMap, 'nufiss'), ringNuFissRateRelError]
ringPath = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_ringres_cellPointdataAppend.csv'

elemParams = ['nuFissRate', 'nuFissRateRelError']
elemVals = [getUniValsList(elemGriffMap, 'nufiss'), elemNuFissRateRelError]
elemPath = '/Users/isaacnaupaaguirre/Documents/GitHub/SNAP-REACTORS/
 ↪snapReactors/reactor_models/AutomatedSerpentModels/GCU/c3_radial_sens_2d/
 ↪s82d_ac_c3_gcu_elemres_cellPointdataAppend.csv'

createAppendCSV(corePointData, coreUnis, coreParams, coreVals, corePath)
createAppendCSV(ringPointData, ringUnis, ringParams, ringVals, ringPath)
createAppendCSV(elemPointData, elemUnis, elemParams, elemVals, elemPath);
```



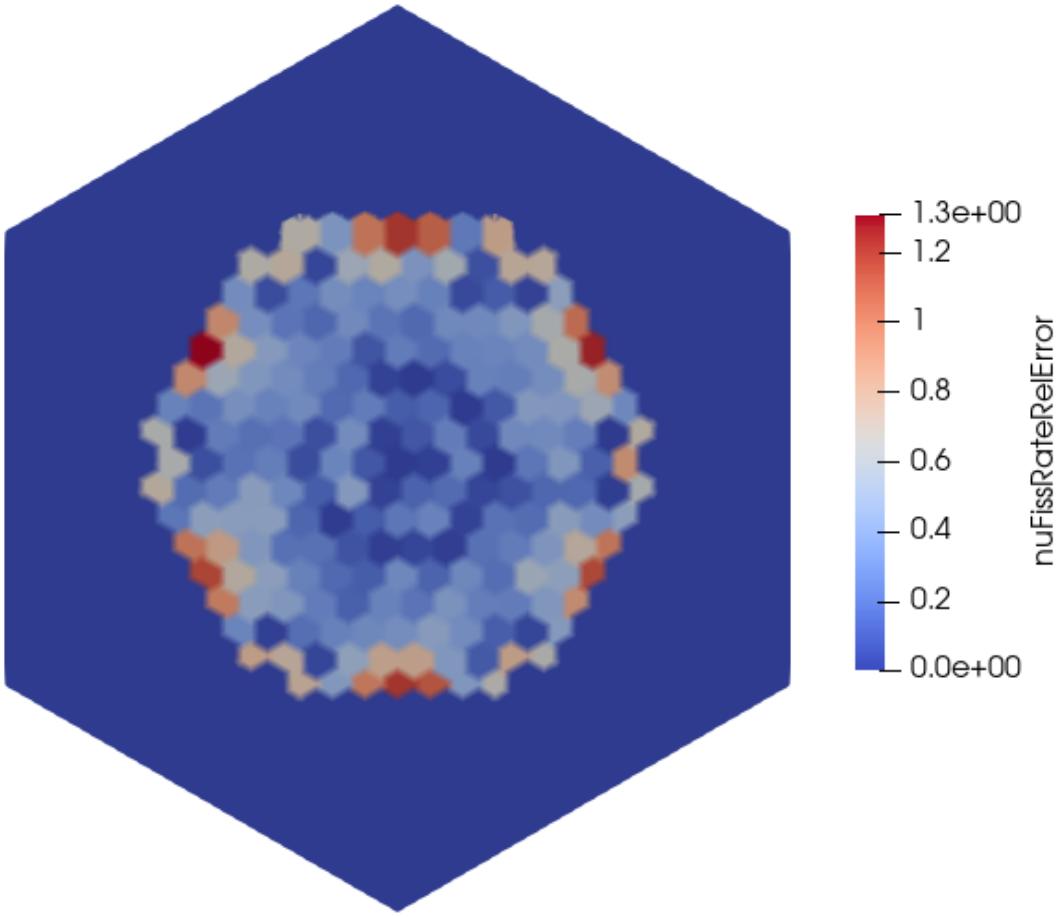
Element Based Flux Map



Element Based - Elem XS Flux Rel Error Map



Element Based - Ring XS Flux Rel Error Map



Element Based - Core XS Flux Rel Error Map