sweep_analysis

July 10, 2018

1 Experiments to determine model behavior across the parameter space

By using our random forcing and perturbations, we run our model hundreds of times for each combination of parameters and compute statistics on how often blocking occurs, and how large blocking events are. We can use this to make predictions for climate change.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
```

1.0.1 What effect does the C(x) background wavenumber (normally n=2) have?

```
In [2]: wavenum = np.load("wavenum.npy",encoding='latin1').item()
```

```
In [3]: wavenums = wavenum["var"]
```

```
In [4]: avgnblocks = []
        avgblocksz = []
        stdnblocks = []
        stdblocksz = []
        avgnblockspe = []
        avgblockszpe = []
        stdnblockspe = []
        stdblockszpe = []
        wnevents = []
        wstdevents = []
```

```
In [5]: len(wavenum["data"])
```

```
Out[5]: 10
```

```
In [6]: for run in wavenum["data"]:
            avgnblocks.append(run["avg_nblocks"])
            avgblocksz.append(run["avg_blocksize"])
            stdnblocks.append(run["std_nblocks"])
            stdblocksz.append(run["std_blocksize"])
            avgnblockspe.append(run["avg_nblocks_perevent"])
            avgblockszpe.append(run["avg_blocksize_perevent"])
```

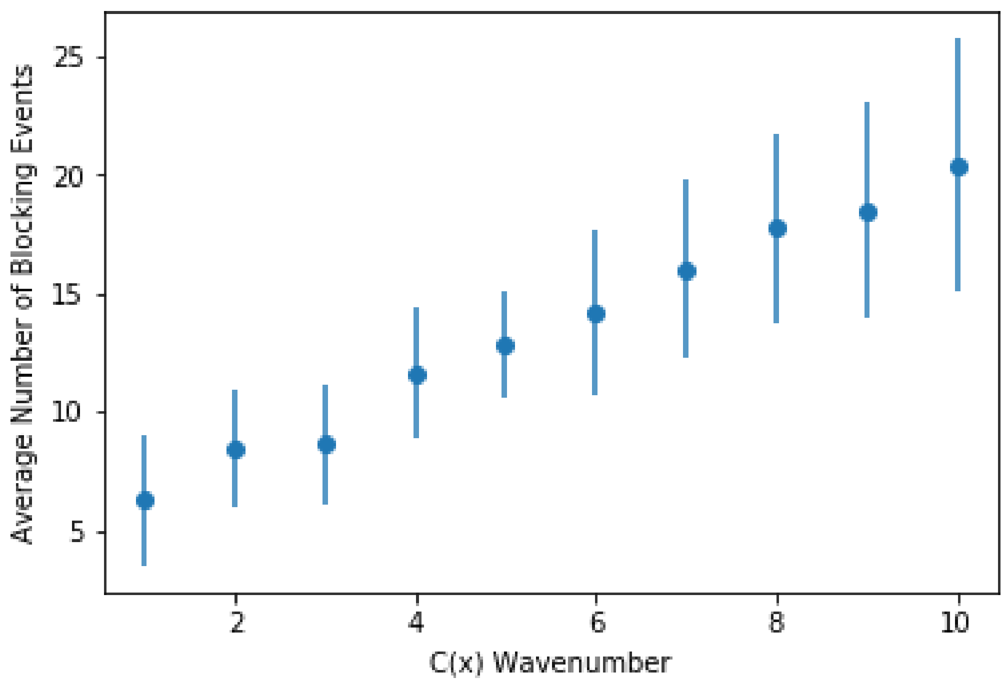
```
        stdnblockspe.append(run["std_nblocks_perevent"])
        stdblockszpe.append(run["std_blocksize_perevent"])
        eventsn = np.zeros(10)
        for n in range(0,10):
            eventsn[n] = len(run["forcing_coords"][n][0])
        wnevents.append(np.mean(eventsn))
        wstdevents.append(np.std(eventsn))
```

```
In [7]: len(avgnblocks)
```

```
Out[7]: 10
```

```
In [8]: plt.errorbar(wavenums,avgnblocks,yerr=stdnblocks,fmt='o')
        plt.xlabel("C(x) Wavenumber")
        plt.ylabel("Average Number of Blocking Events")
```

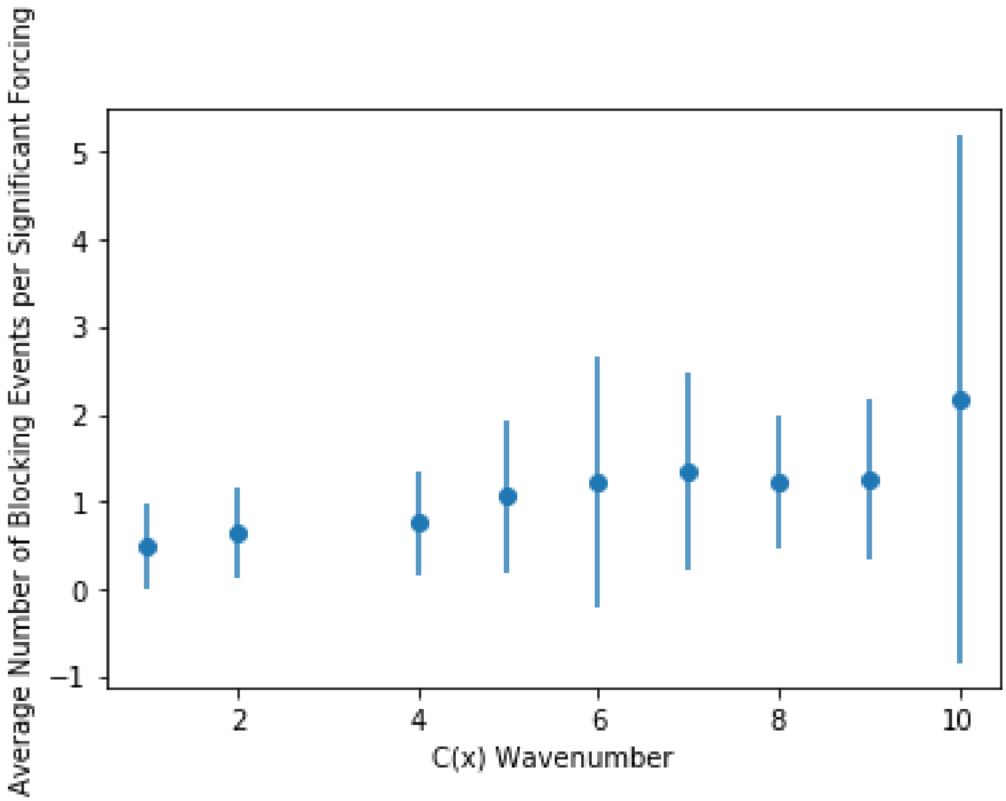
```
Out[8]: Text(0,0.5,'Average Number of Blocking Events')
```



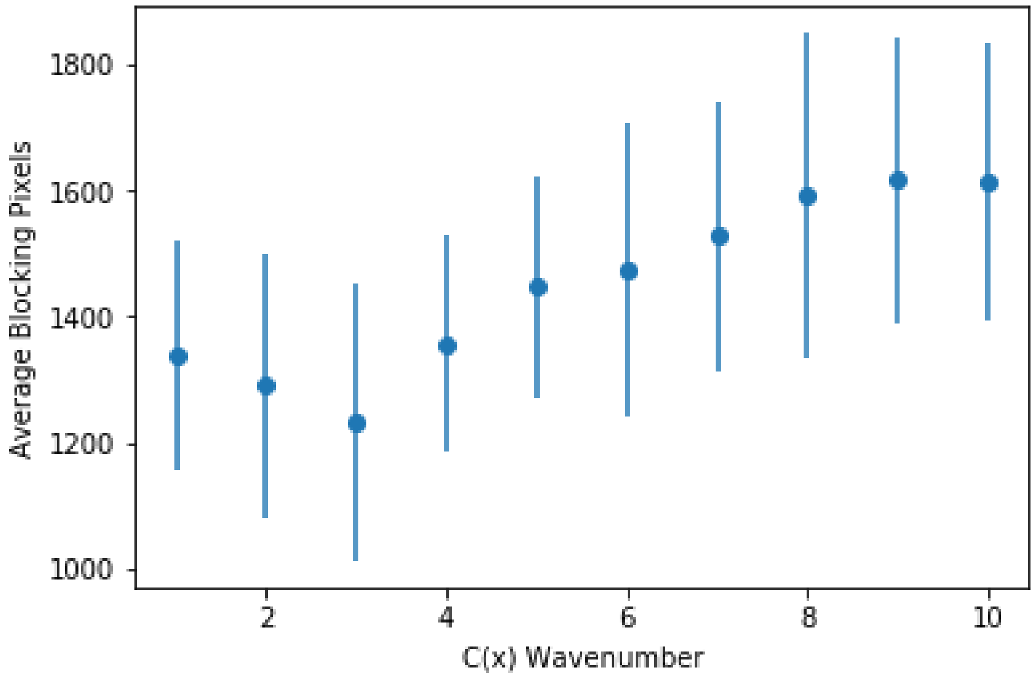
Number of blocking events increase.

```
In [9]: plt.errorbar(wavenums,avgnblockspe,yerr=stdnblockspe,fmt='o')
        plt.xlabel("C(x) Wavenumber")
        plt.ylabel("Average Number of Blocking Events per Significant Forcing")
```

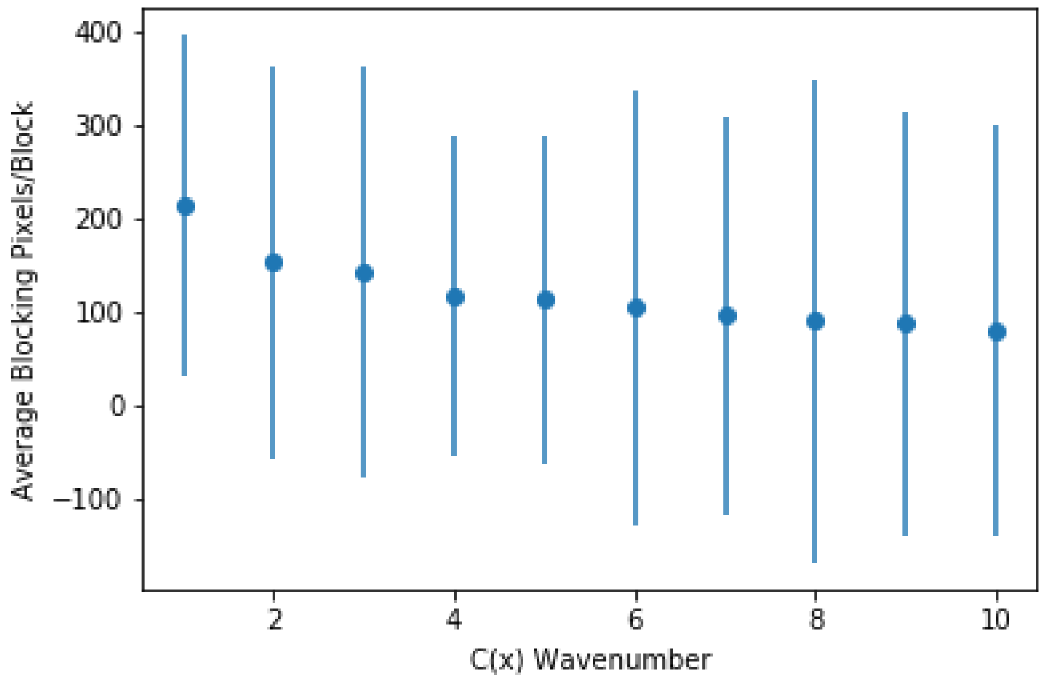
```
Out[9]: Text(0,0.5,'Average Number of Blocking Events per Significant Forcing')
```




```
In [10]: plt.errorbar(wavenums,avgblocksz,yerr=stdblocksz,fmt='o')
         plt.xlabel("C(x) Wavenumber")
         plt.ylabel("Average Blocking Pixels")

Out[10]: Text(0,0.5,'Average Blocking Pixels')
```

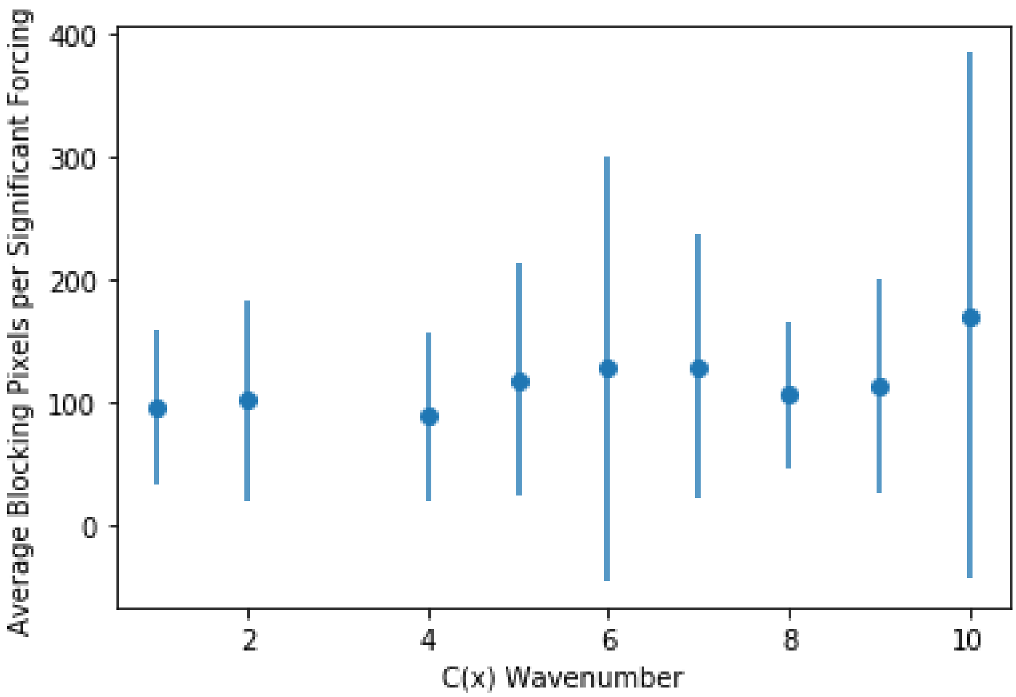



```
In [11]: plt.errorbar(wavenums,np.array(avgblocksz)/np.array(avgnblocks),
                      yerr=np.sqrt(np.array(stdnblocks)**2+np.array(stdblocksz)**2),
                      fmt='o')
         plt.xlabel("C(x) Wavenumber")
         plt.ylabel("Average Blocking Pixels/Block")

Out[11]: Text(0,0.5,'Average Blocking Pixels/Block')
```



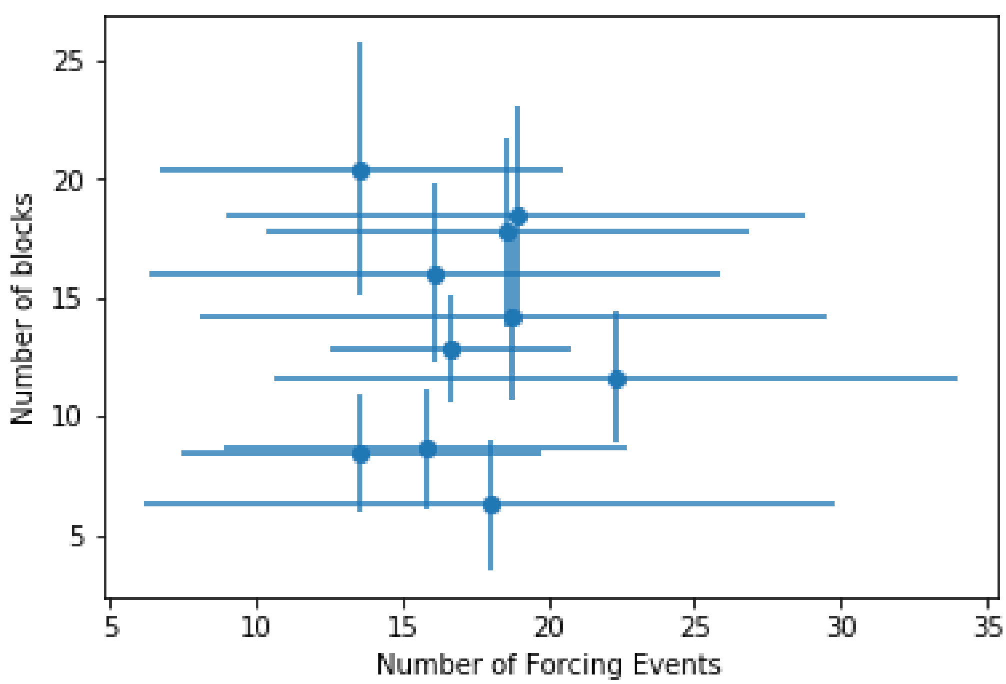
The size of the blocks might get smaller, but we lack statistical strength here.

```
In [12]: plt.errorbar(wavenums,avgblockszpe,yerr=stdblockszpe,fmt='o')
         plt.xlabel("C(x) Wavenumber")
         plt.ylabel("Average Blocking Pixels per Significant Forcing")

Out[12]: Text(0,0.5,'Average Blocking Pixels per Significant Forcing')
```



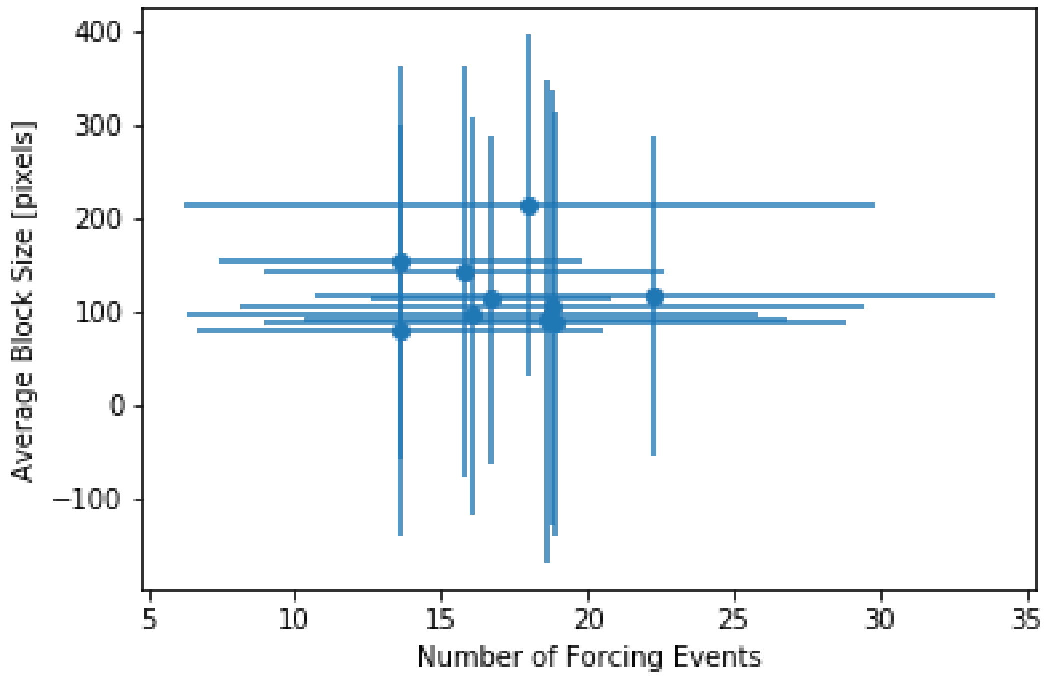

```
In [13]: plt.errorbar(wnevents,avgnblocks,xerr=wstdevents,yerr=stdnblocks,fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Number of blocks")
```

```
Out[13]: Text(0,0.5,'Number of blocks')
```



```
In [14]: plt.errorbar(wnevents,np.array(avgblocksz)/np.array(avgnblocks),xerr=wstdevents,
                      yerr=np.sqrt(np.array(stdnblocks)**2+np.array(stdblocksz)**2),fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Average Block Size [pixels]")
```

```
Out[14]: Text(0,0.5,'Average Block Size [pixels]')
```



1.0.2 What effect does varying the forcing strength have?

```
In [15]: fpeak = np.load("fpeak.npy",encoding='latin1').item()
```

```
In [16]: fpeaks = fpeak["var"]
```

```
In [17]: favgnblocks = []
         favgblocksz = []
         fstdnblocks = []
         fstdblocksz = []
         favgnblockspe = []
         favgblockszpe = []
         fstdnblockspe = []
         fstdblockszpe = []
         nevents = []
         stdevents = []
```

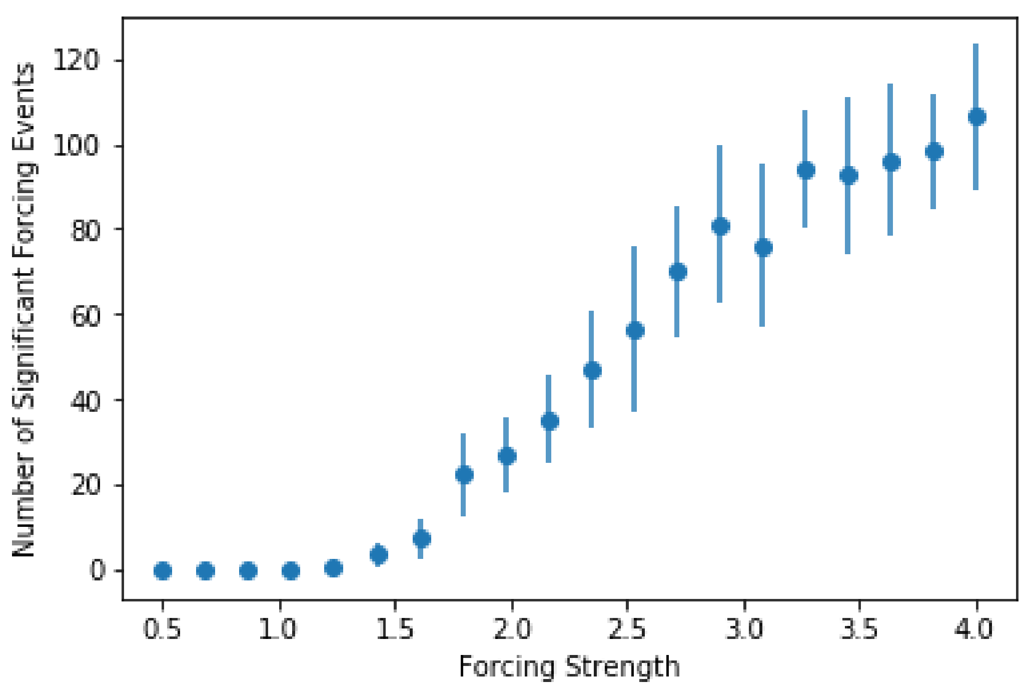
```
In [18]: len(fpeak["data"])
```

```
Out[18]: 20
```

```
In [19]: for run in fpeak["data"]:
             favgnblocks.append(run["avg_nblocks"])
             favgblocksz.append(run["avg_blocksize"])
             fstdnblocks.append(run["std_nblocks"])
             fstdblocksz.append(run["std_blocksize"])
             favgnblockspe.append(run["avg_nblocks_perevent"])
             favgblockszpe.append(run["avg_blocksize_perevent"])
             fstdnblockspe.append(run["std_nblocks_perevent"])
             fstdblockszpe.append(run["std_blocksize_perevent"])
             eventsn = np.zeros(20)
             for n in range(0,20):
                 eventsn[n] = len(run["forcing_coords"][n][0])
             nevents.append(np.mean(eventsn))
             stdevents.append(np.std(eventsn))
```

```
In [20]: plt.errorbar(fpeaks,nevents,yerr=stdevents,fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Number of Significant Forcing Events")
```

```
Out[20]: Text(0,0.5,'Number of Significant Forcing Events')
```

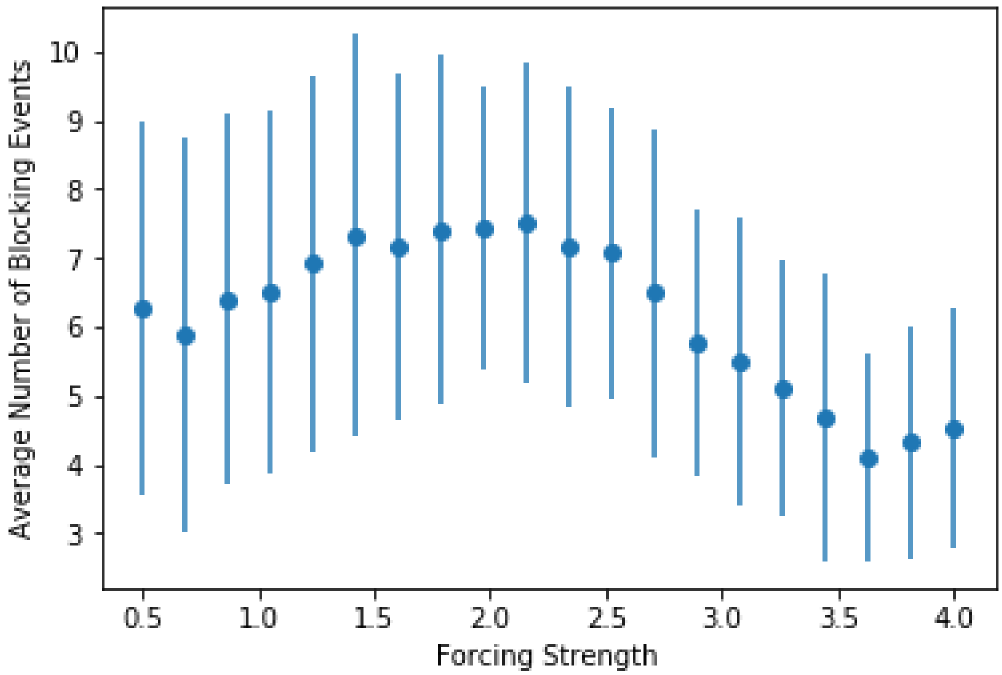



```
In [21]: len(favgnblocks)
```

Out[21]: 20

```
In [22]: plt.errorbar(fpeaks,favgnblocks,yerr=fstdnblocks,fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Average Number of Blocking Events")
```

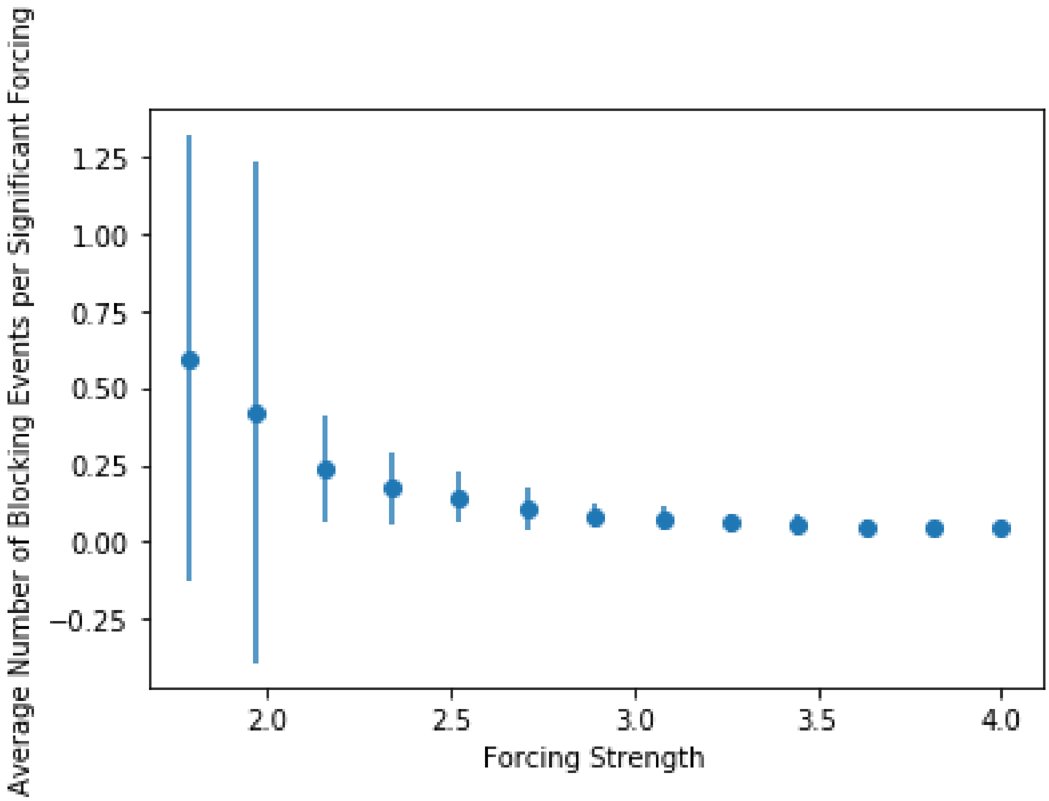
Out[22]: Text(0,0.5,'Average Number of Blocking Events')



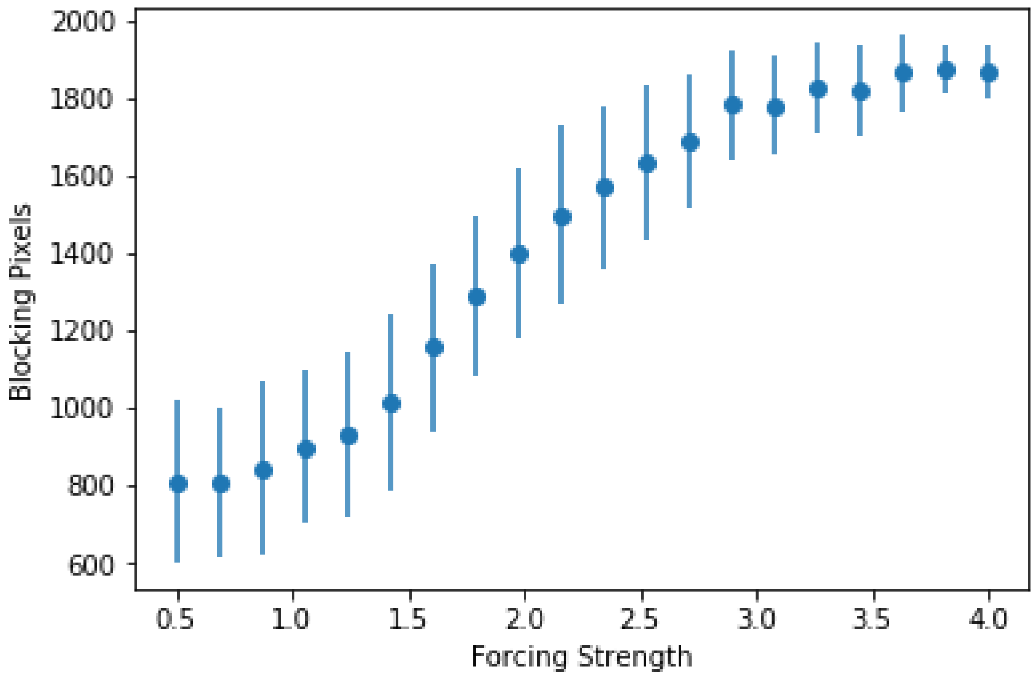
The way to understand this turnover is that blocks merge into larger blocks.

```
In [23]: plt.errorbar(fpeaks,favgnblockspe,yerr=fstdnblockspe,fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Average Number of Blocking Events per Significant Forcing")
```

Out[23]: Text(0,0.5,'Average Number of Blocking Events per Significant Forcing')

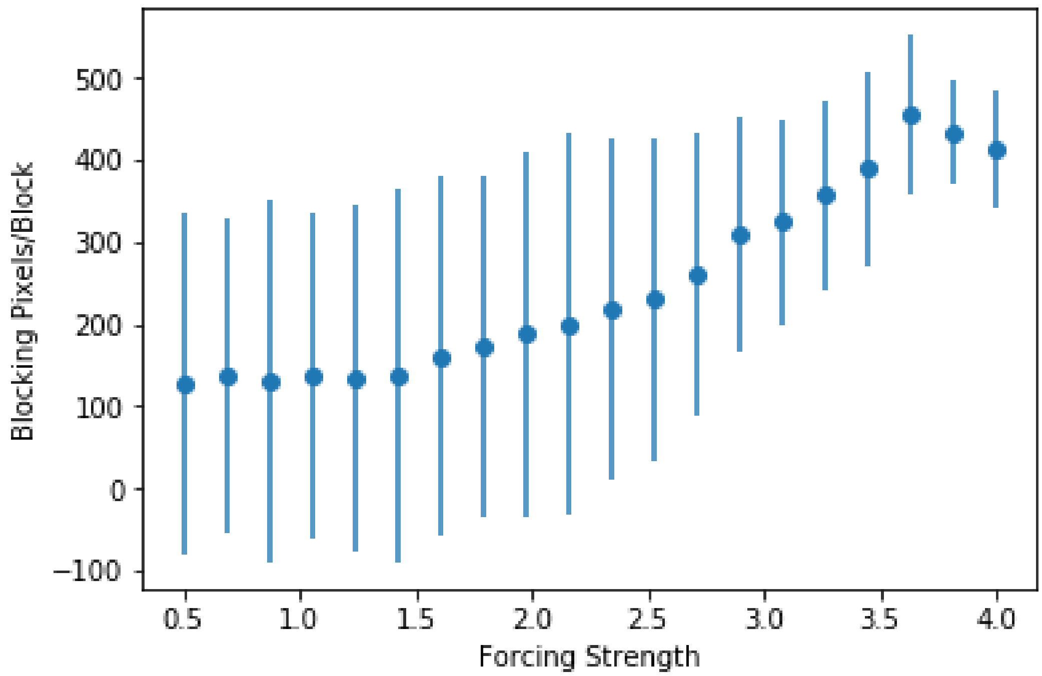



```
In [24]: plt.errorbar(fpeaks,favgblocksz,yerr=fstdblocksz,fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Blocking Pixels")

Out[24]: Text(0,0.5,'Blocking Pixels')
```

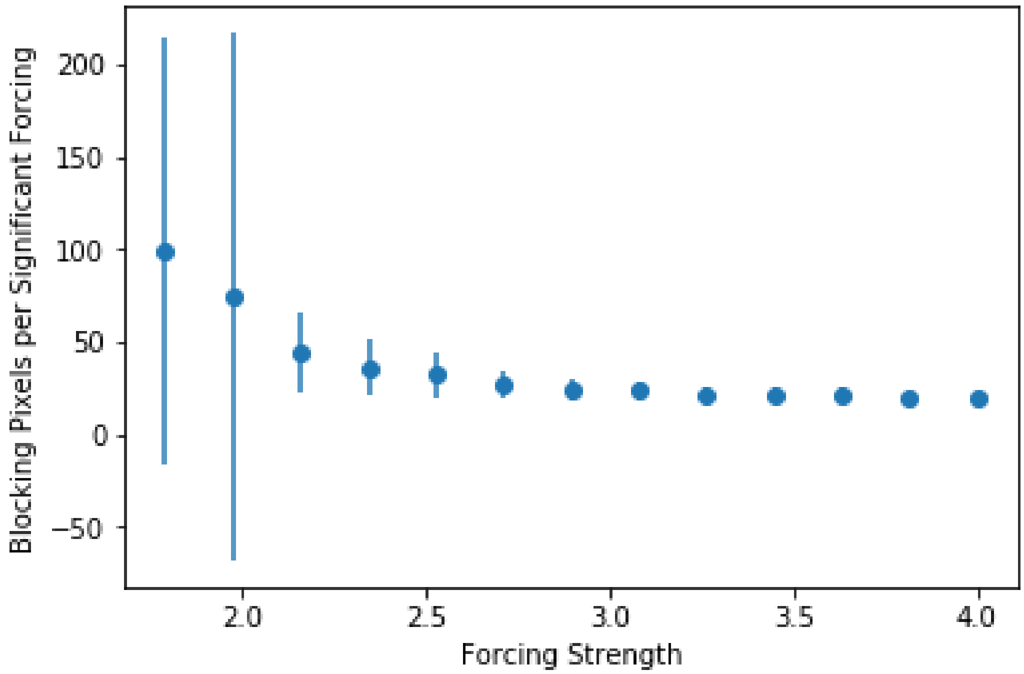


Here we see a clear convergence toward perpetual blocking.

```
In [25]: plt.errorbar(fpeaks,np.array(favgblocksz)/np.array(favgnblocks),
                      yerr=np.sqrt(np.array(fstdnblocks)**2+np.array(fstdblocksz)**2),
                      fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Blocking Pixels/Block")

Out[25]: Text(0,0.5,'Blocking Pixels/Block')
```

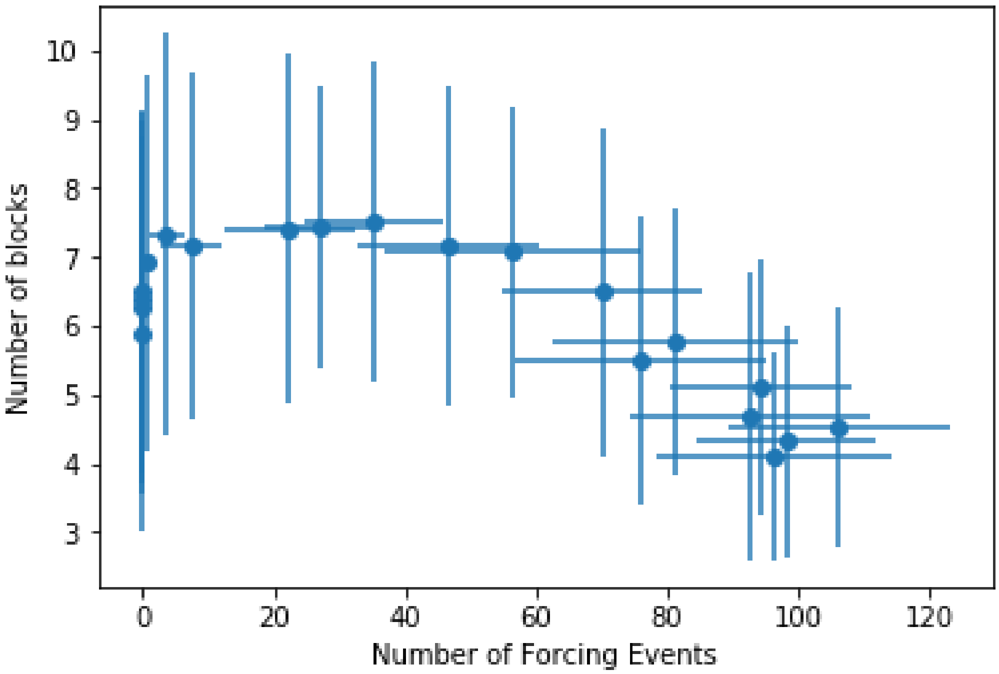



```
In [26]: plt.errorbar(fpeaks,favgblockszpe,yerr=fstdblockszpe,fmt='o')
         plt.xlabel("Forcing Strength")
         plt.ylabel("Blocking Pixels per Significant Forcing")

Out[26]: Text(0,0.5,'Blocking Pixels per Significant Forcing')
```



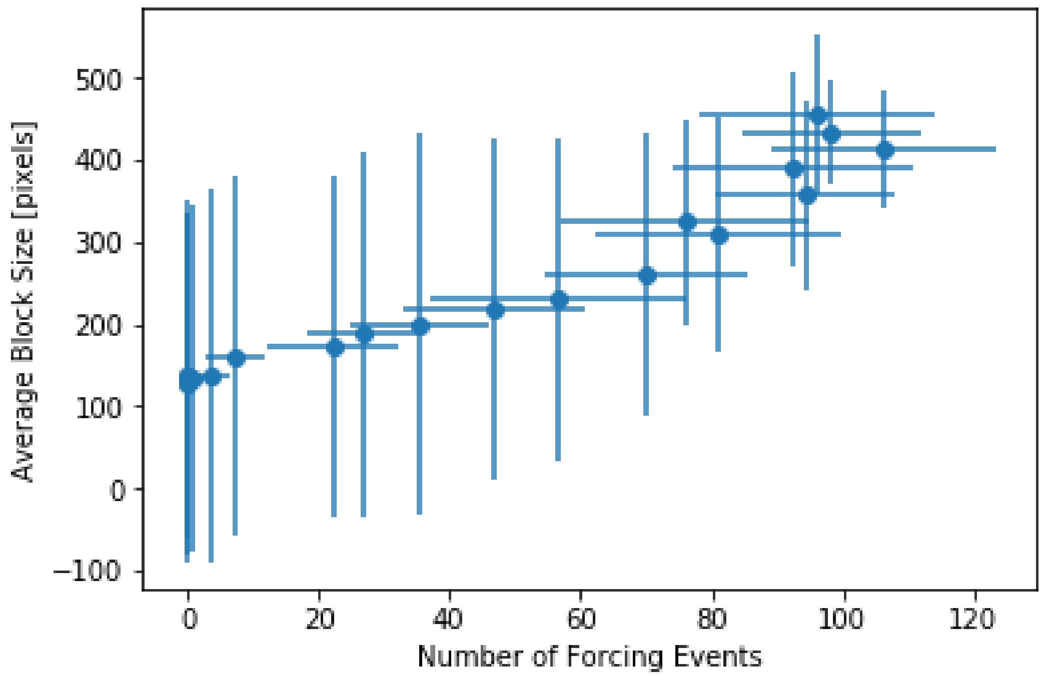
In [27]:
```
plt.errorbar(nevents,favgnblocks,xerr=stdevents,yerr=fstdnblocks,fmt='o')
plt.xlabel("Number of Forcing Events")
plt.ylabel("Number of blocks")
```

Out[27]: Text(0,0.5,'Number of blocks')



In [28]:
```
plt.errorbar(nevents,np.array(favgblocksz)/np.array(favgnblocks),
             xerr=stdevents,yerr=np.sqrt(np.array(fstdnblocks)**2+\
                                          np.array(fstdblocksz)**2),
             fmt='o')
plt.xlabel("Number of Forcing Events")
plt.ylabel("Average Block Size [pixels]")
```

Out[28]: Text(0,0.5,'Average Block Size [pixels]')



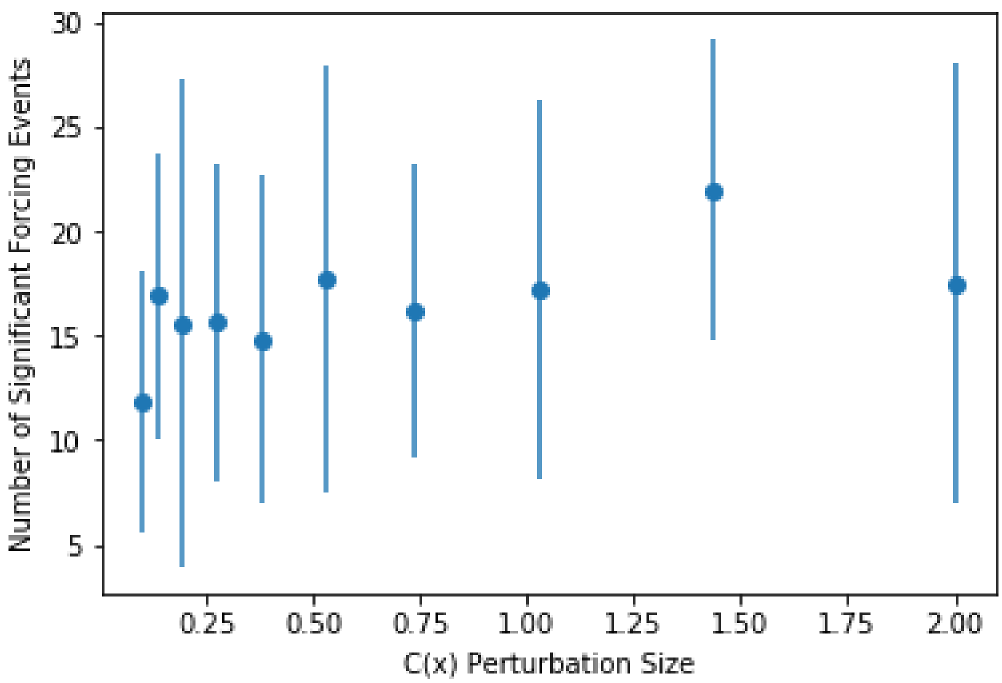
1.0.3 What effect does varying the strength of the C(x,t) perturbation have?

```
In [29]: cstrength = np.load("cstrength.npy",encoding='latin1').item()

In [30]: cstrengths = cstrength["var"]

In [31]: cavgnblocks = []
         cavgblocksz = []
         cstdnblocks = []
         cstdblocksz = []
         cavgnblockspe = []
         cavgblockszpe = []
         cstdnblockspe = []
         cstdblockszpe = []
         cnevents = []
         cstdevents = []

In [32]: len(cstrength["data"])

Out[32]: 10

In [33]: for run in cstrength["data"]:
             cavgnblocks.append(run["avg_nblocks"])
             cavgblocksz.append(run["avg_blocksize"])
             cstdnblocks.append(run["std_nblocks"])
```

```
            cstdblocksz.append(run["std_blocksize"])
            cavgnblockspe.append(run["avg_nblocks_perevent"])
            cavgblockszpe.append(run["avg_blocksize_perevent"])
            cstdnblockspe.append(run["std_nblocks_perevent"])
            cstdblockszpe.append(run["std_blocksize_perevent"])
            eventsn = np.zeros(20)
            for n in range(0,20):
                eventsn[n] = len(run["forcing_coords"][n][0])
            cnevents.append(np.mean(eventsn))
            cstdevents.append(np.std(eventsn))
```

In [34]:
```
plt.errorbar(cstrengths,cnevents,yerr=cstdevents,fmt='o')
plt.xlabel("C(x) Perturbation Size")
plt.ylabel("Number of Significant Forcing Events")
```

Out[34]: Text(0,0.5,'Number of Significant Forcing Events')

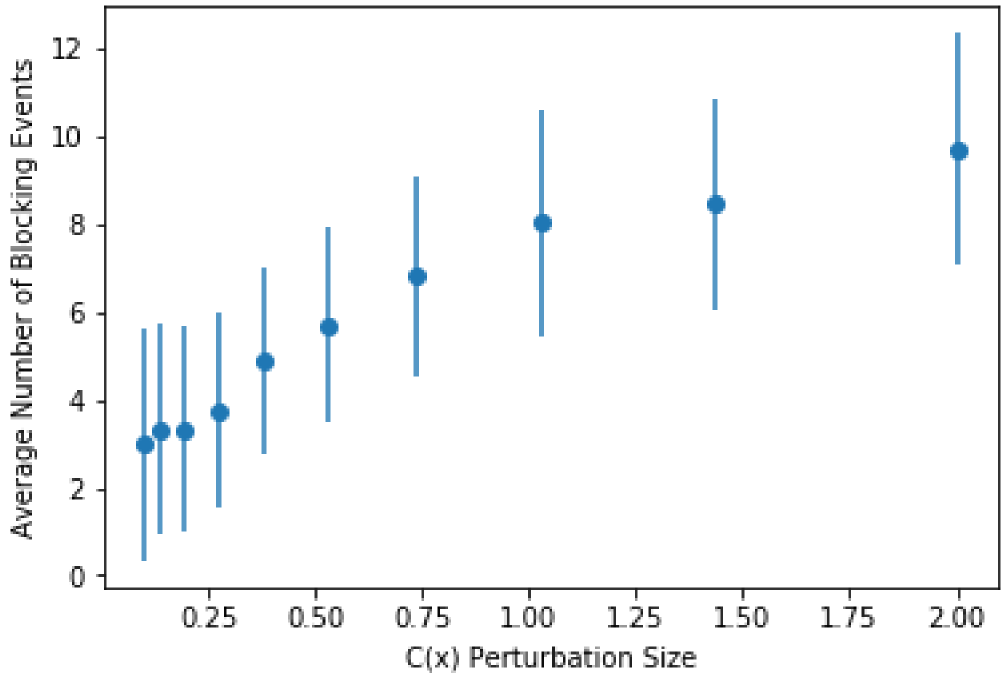


In [35]:
```
len(cavgnblocks)
```

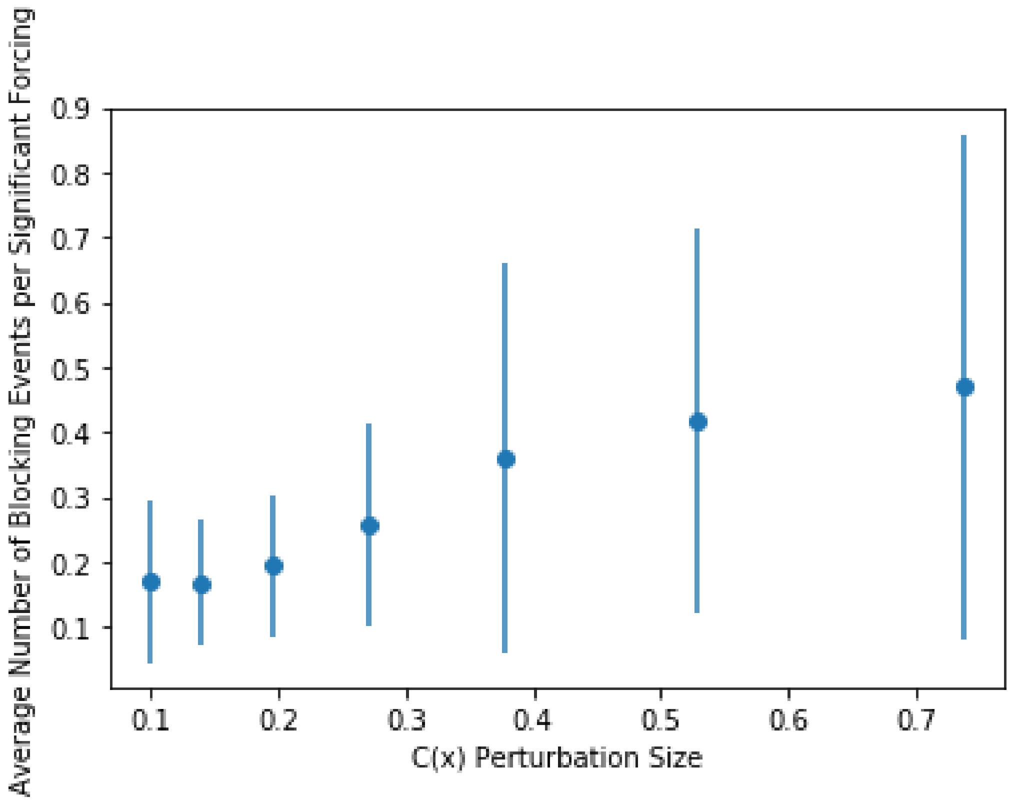
Out[35]: 10

In [36]:
```
plt.errorbar(cstrengths,cavgnblocks,yerr=cstdnblocks,fmt='o')
plt.xlabel("C(x) Perturbation Size")
plt.ylabel("Average Number of Blocking Events")
```

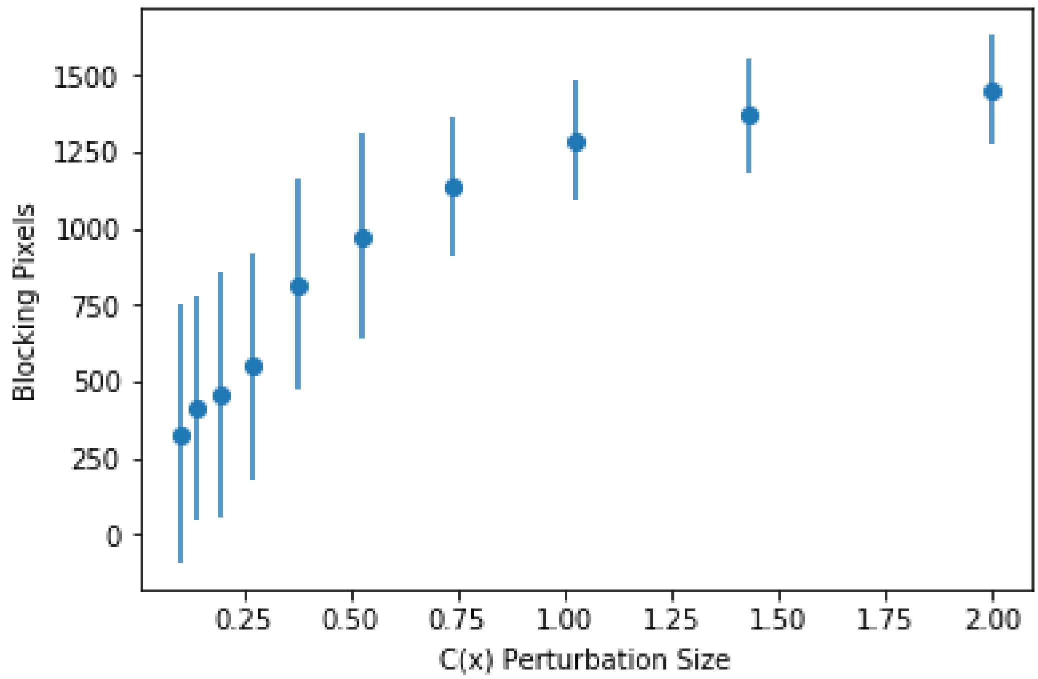
Out[36]: Text(0,0.5,'Average Number of Blocking Events')



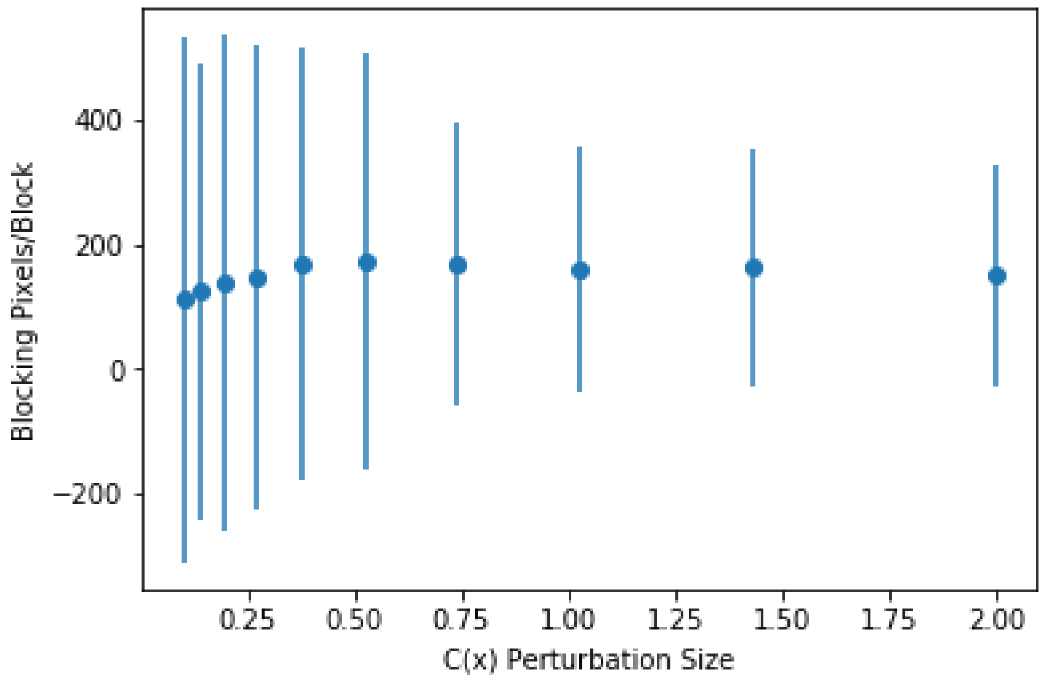

```
In [37]: plt.errorbar(cstrengths,cavgnblockspe,yerr=cstdnblockspe,fmt='o')
         plt.xlabel("C(x) Perturbation Size")
         plt.ylabel("Average Number of Blocking Events per Significant Forcing")

Out[37]: Text(0,0.5,'Average Number of Blocking Events per Significant Forcing')
```



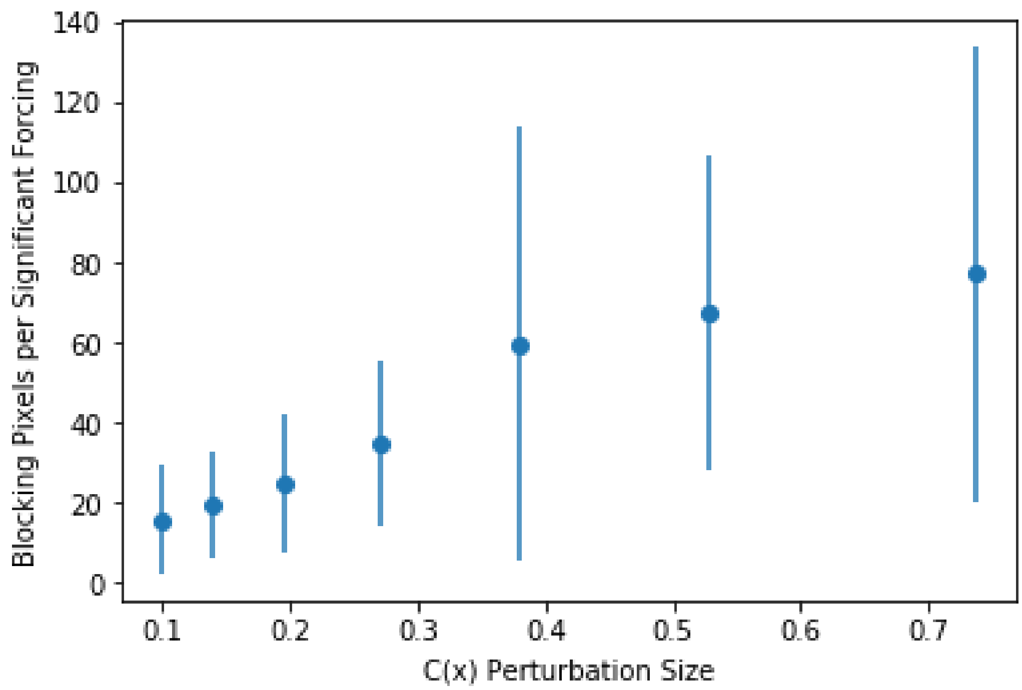

```
In [38]: plt.errorbar(cstrengths,cavgblocksz,yerr=cstdblocksz,fmt='o')
         plt.xlabel("C(x) Perturbation Size")
         plt.ylabel("Blocking Pixels")

Out[38]: Text(0,0.5,'Blocking Pixels')
```




```
In [39]: plt.errorbar(cstrengths,np.array(cavgblocksz)/np.array(cavgnblocks),
                      yerr=np.sqrt(np.array(cstdnblocks)**2+np.array(cstdblocksz)**2),
                      fmt='o')
         plt.xlabel("C(x) Perturbation Size")
         plt.ylabel("Blocking Pixels/Block")

Out[39]: Text(0,0.5,'Blocking Pixels/Block')
```

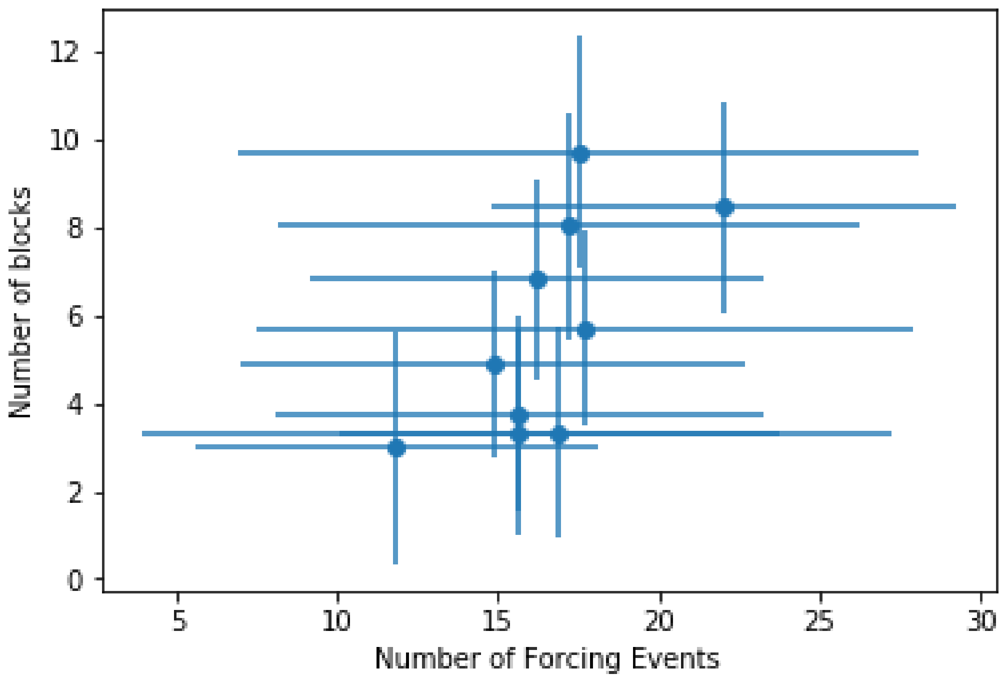



```
In [40]: plt.errorbar(cstrengths,cavgblockszpe,yerr=cstdblockszpe,fmt='o')
         plt.xlabel("C(x) Perturbation Size")
         plt.ylabel("Blocking Pixels per Significant Forcing")

Out[40]: Text(0,0.5,'Blocking Pixels per Significant Forcing')
```



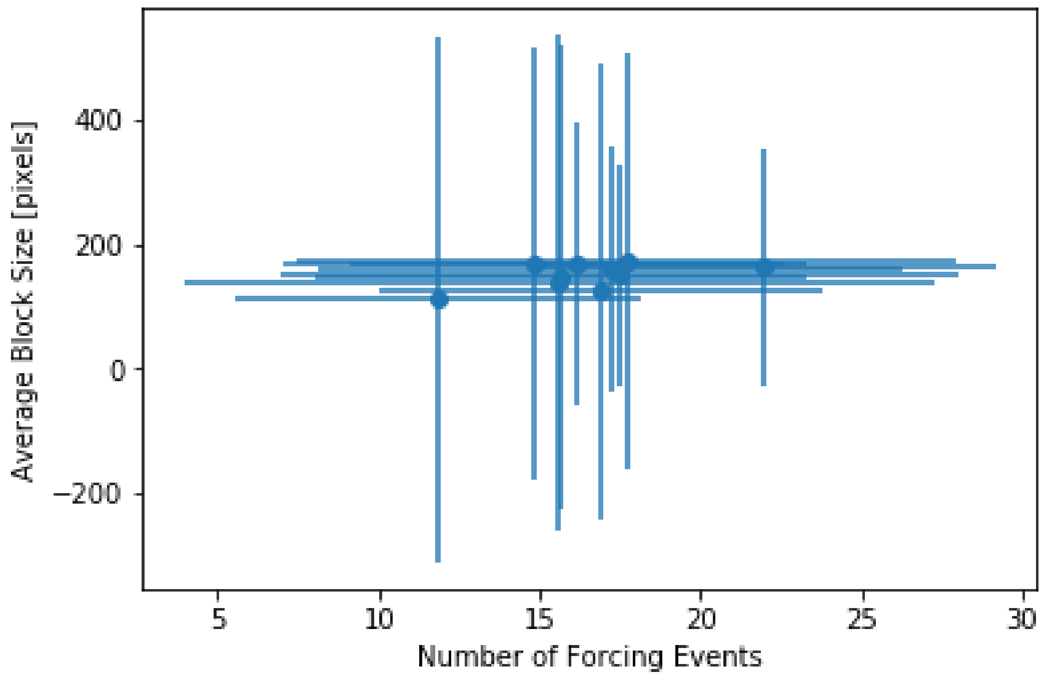
In [41]:
```
plt.errorbar(cnevents,cavgnblocks,xerr=cstdevents,yerr=cstdnblocks,fmt='o')
plt.xlabel("Number of Forcing Events")
plt.ylabel("Number of blocks")
```

Out[41]:
```
Text(0,0.5,'Number of blocks')
```



In [42]:
```
plt.errorbar(cnevents,np.array(cavgblocksz)/np.array(cavgnblocks),
             xerr=cstdevents,yerr=np.sqrt(np.array(cstdnblocks)**2+\
                                          np.array(cstdblocksz)**2),
             fmt='o')
plt.xlabel("Number of Forcing Events")
plt.ylabel("Average Block Size [pixels]")
```

Out[42]:
```
Text(0,0.5,'Average Block Size [pixels]')
```



1.0.4 Let's do quick runs where we hold the forcing constant--no statistics yet.

We'll look at varying β and also A_0.

```
In [43]: cx_beta = np.load("cx_a0y.npy",encoding='latin1').item()
```

```
In [44]: cx_a0y = np.load("cx_beta.npy",encoding='latin1').item()
```

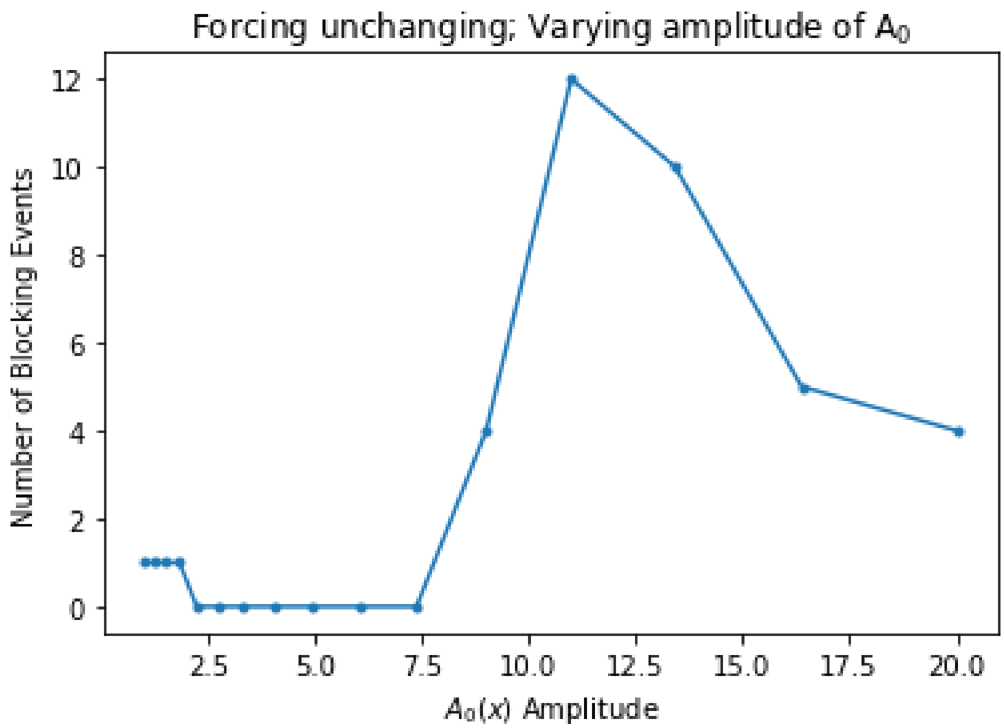
```
In [45]: for k in sorted(cx_a0y.keys()):
             print(k)
```

```
a0y
block_coords
c(x)_init
cx_peak
forcing_coords
forcing_init
forcing_peak
raw_blocksize
raw_blocksize_perevent
raw_nblocks
raw_nblocks_perevent
```

```
In [46]: plt.plot(cx_a0y["a0y"],cx_a0y["raw_nblocks"],marker='.')
         plt.xlabel("$A_0(x)$ Amplitude")
```

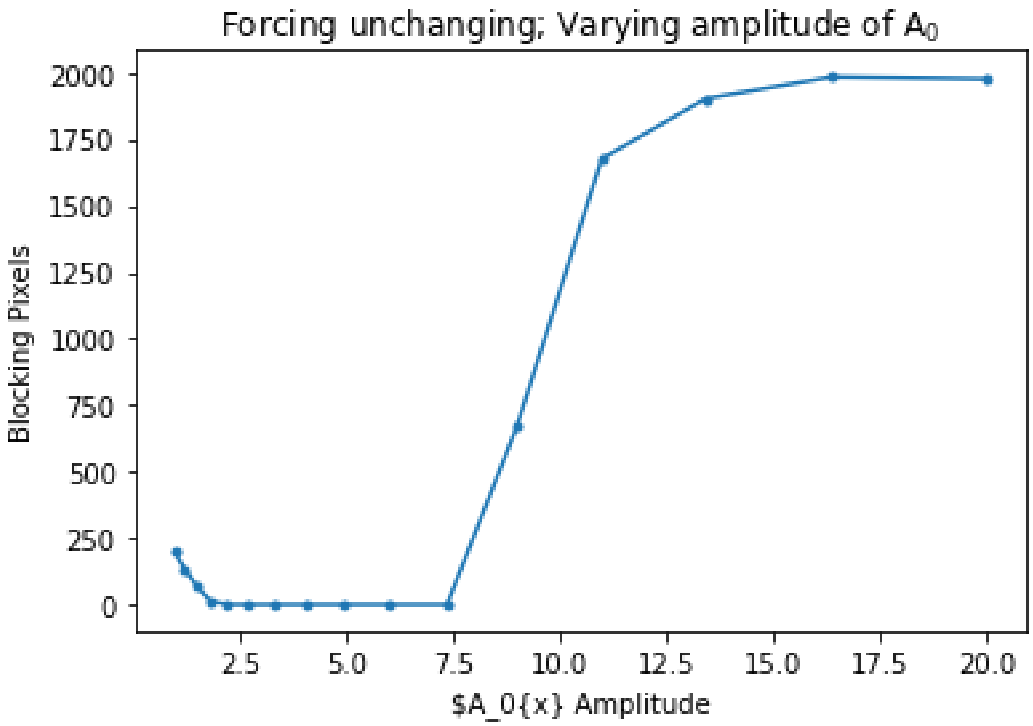
```
plt.ylabel("Number of Blocking Events")
plt.title("Forcing unchanging; Varying amplitude of A$_0$")
```

Out[46]: Text(0.5,1,'Forcing unchanging; Varying amplitude of A$_0$')

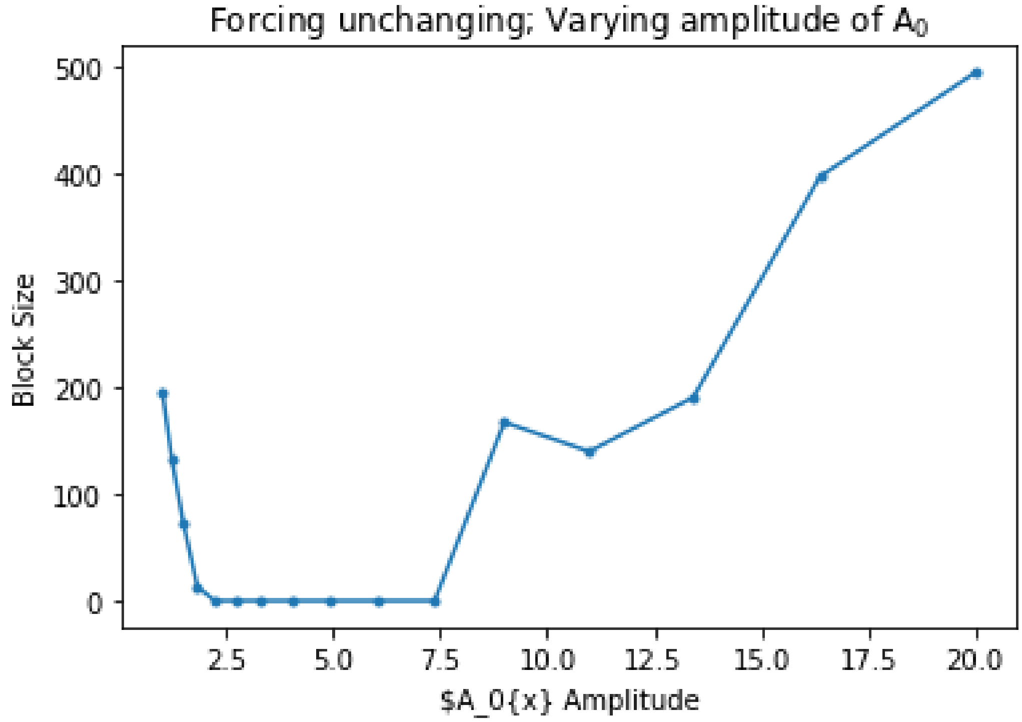


In [47]:
```
plt.plot(cx_a0y["a0y"],cx_a0y["raw_blocksize"],marker='.')
plt.xlabel("$A_0{x} Amplitude")
plt.ylabel("Blocking Pixels")
plt.title("Forcing unchanging; Varying amplitude of A$_0$")
```

Out[47]: Text(0.5,1,'Forcing unchanging; Varying amplitude of A$_0$')

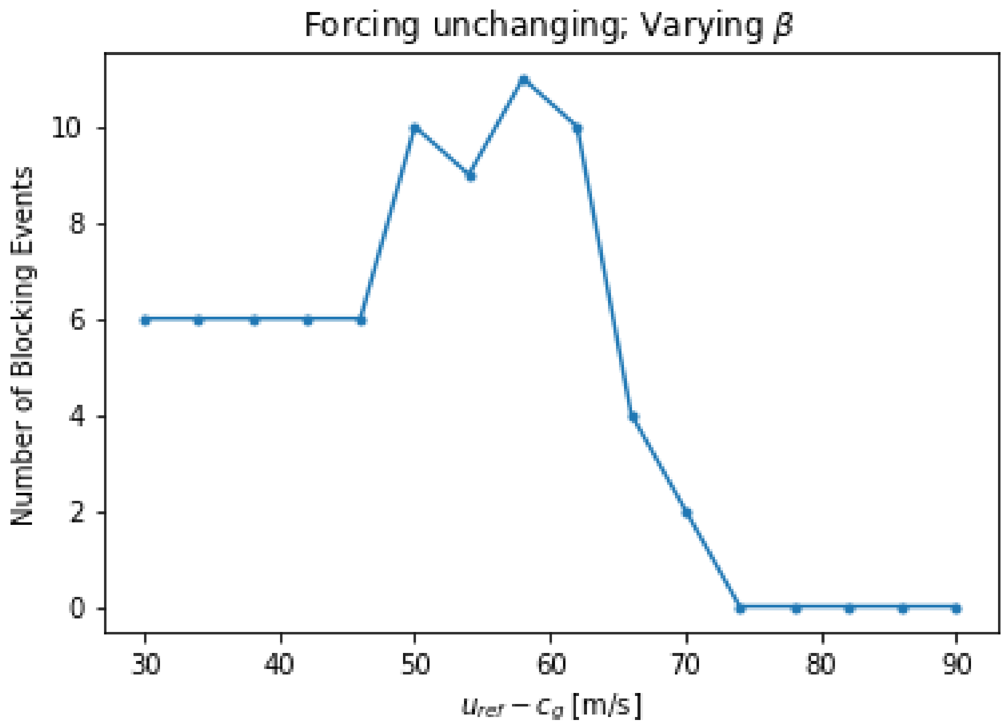



```
In [48]: avgsize = cx_a0y["raw_blocksize"]/cx_a0y["raw_nblocks"]
         plt.plot(cx_a0y["a0y"],np.nan_to_num(avgsize),marker='.')
         plt.xlabel("$A_0{x} Amplitude")
         plt.ylabel("Block Size")
         plt.title("Forcing unchanging; Varying amplitude of A$_0$")

/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.


Out[48]: Text(0.5,1,'Forcing unchanging; Varying amplitude of A$_0$')
```

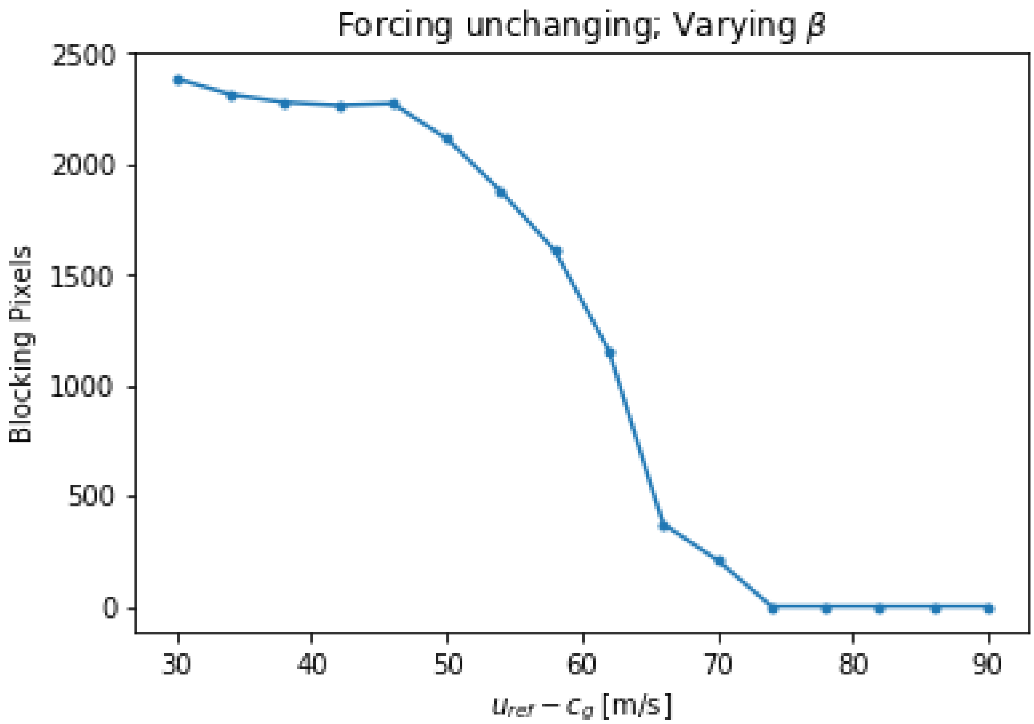


In [49]:
```
plt.plot(cx_beta["beta"],cx_beta["raw_nblocks"],marker='.')
plt.xlabel("$u_{ref}-c_g$ [m/s]")
plt.ylabel("Number of Blocking Events")
plt.title("Forcing unchanging; Varying $\\beta$")
```

Out[49]:
```
Text(0.5,1,'Forcing unchanging; Varying $\\beta$')
```



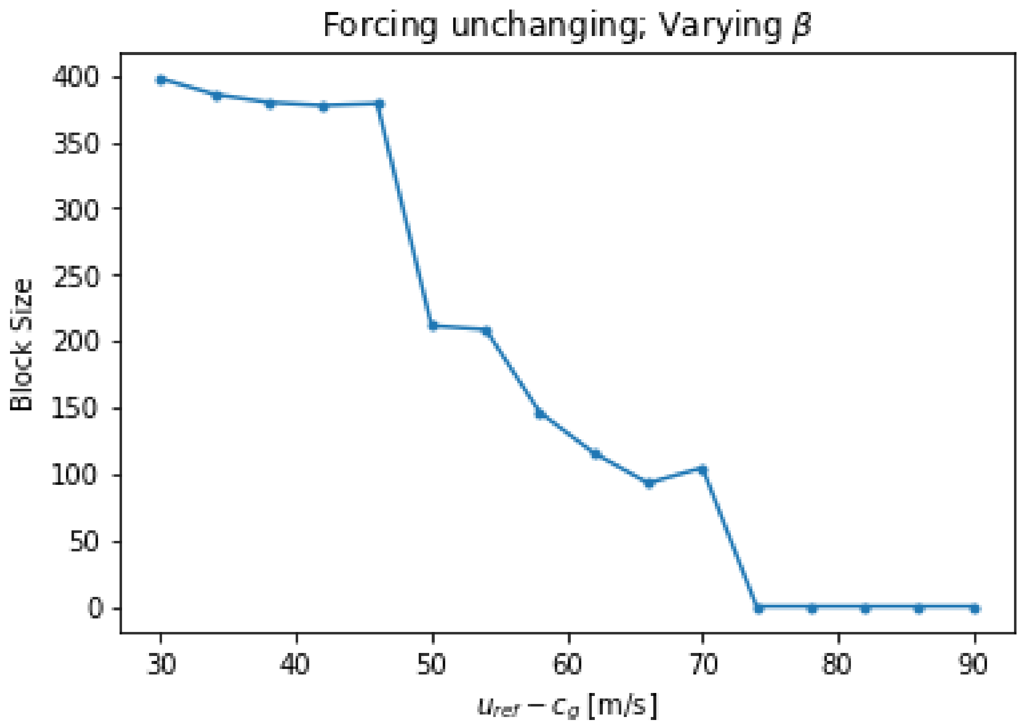

```
In [50]: plt.plot(cx_beta["beta"],cx_beta["raw_blocksize"],marker='.')
         plt.xlabel("$u_{ref}-c_g$ [m/s]")
         plt.ylabel("Blocking Pixels")
         plt.title("Forcing unchanging; Varying $\\beta$")

Out[50]: Text(0.5,1,'Forcing unchanging; Varying $\\beta$')
```




```
In [51]: avgsize = cx_beta["raw_blocksize"]/cx_beta["raw_nblocks"]
         plt.plot(cx_beta["beta"],np.nan_to_num(avgsize),marker='.')
         plt.xlabel("$u_{ref}-c_g$ [m/s]")
         plt.ylabel("Block Size")
         plt.title("Forcing unchanging; Varying $\\beta$")
```

```
/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.
```

```
Out[51]: Text(0.5,1,'Forcing unchanging; Varying $\\beta$')
```



1.0.5 What if we vary the background amplitude A_0 with statistics?

We'll do a few hundred runs for each parameter combination to compute mean and standard deviation.

```
In [52]: cxy = np.load("cxy.npy",encoding='latin1').item()
         cxb = np.load("beta.npy",encoding='latin1').item()
```

```
In [53]: ampls = cxy["var"]
```

```
In [54]: yavgnblocks = []
         yavgblocksz = []
         ystdnblocks = []
         ystdblocksz = []
         yavgnblockspe = []
         yavgblockszpe = []
         ystdnblockspe = []
         ystdblockszpe = []
         ynevents = []
         ystdevents = []
```

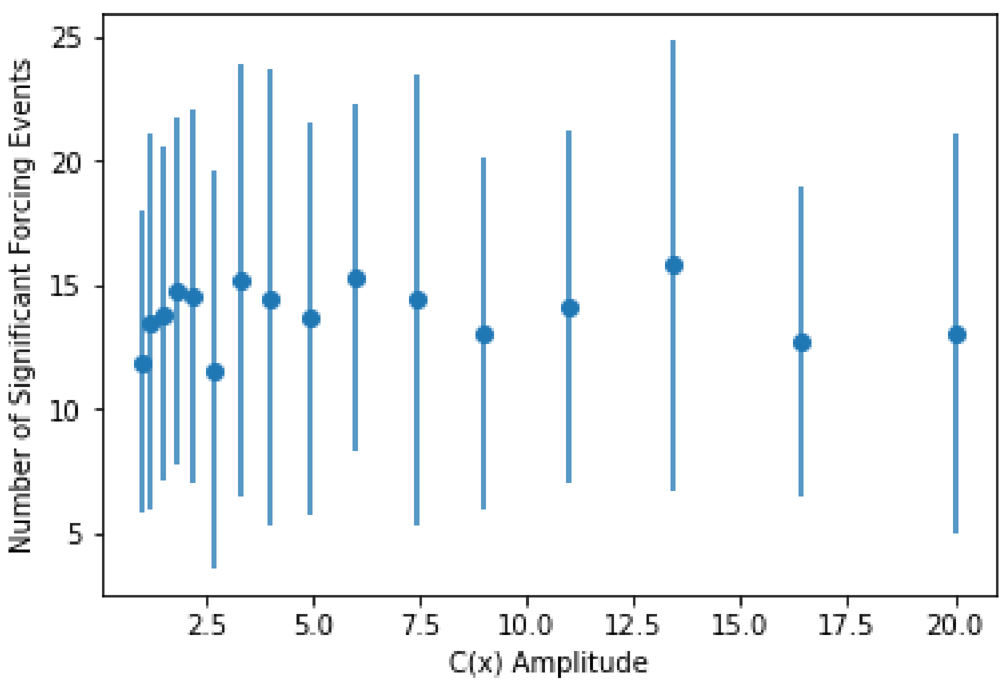
```
In [55]: len(cxy["data"])
```

```
Out[55]: 16
```

```
In [56]: for run in cxy["data"]:
             yavgnblocks.append(run["avg_nblocks"])
             yavgblocksz.append(run["avg_blocksize"])
             ystdnblocks.append(run["std_nblocks"])
             ystdblocksz.append(run["std_blocksize"])
             yavgnblockspe.append(run["avg_nblocks_perevent"])
             yavgblockszpe.append(run["avg_blocksize_perevent"])
             ystdnblockspe.append(run["std_nblocks_perevent"])
             ystdblockszpe.append(run["std_blocksize_perevent"])
             eventsn = np.zeros(20)
             for n in range(0,20):
                 eventsn[n] = len(run["forcing_coords"][n][0])
             ynevents.append(np.mean(eventsn))
             ystdevents.append(np.std(eventsn))
```

```
In [57]: plt.errorbar(ampls,ynevents,yerr=ystdevents,fmt='o')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Number of Significant Forcing Events")
```

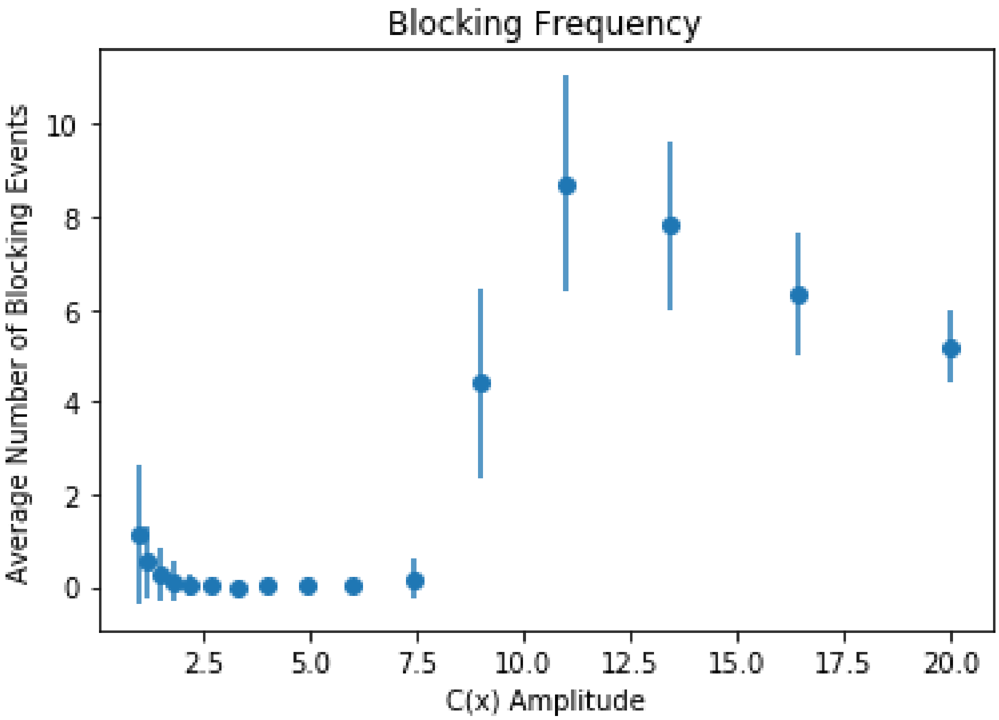
```
Out[57]: Text(0,0.5,'Number of Significant Forcing Events')
```



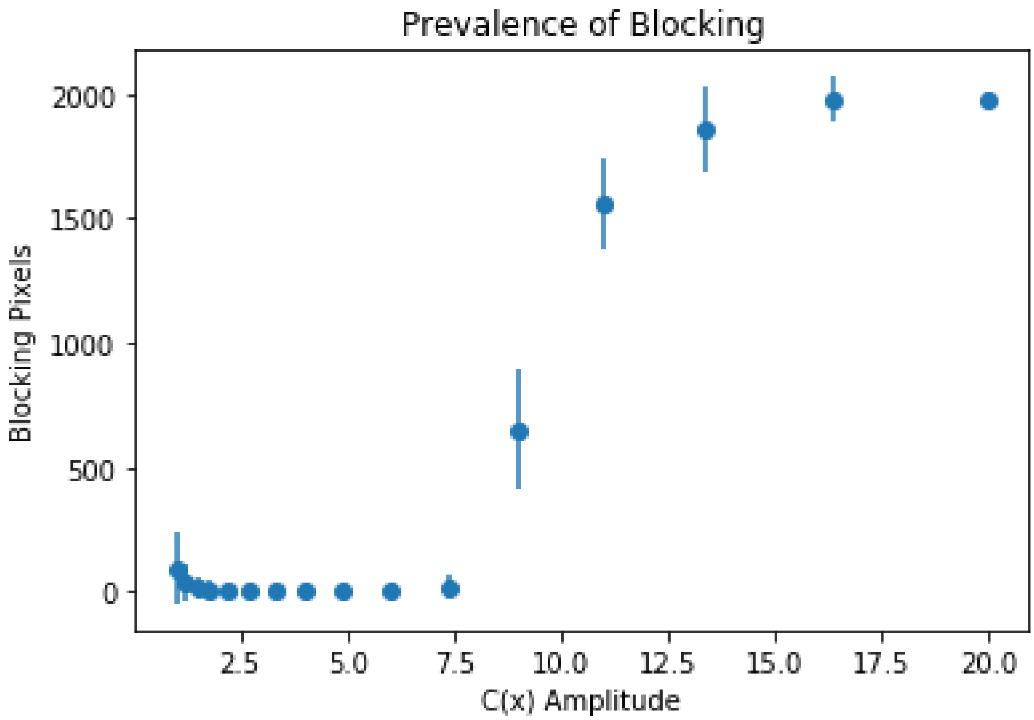

```
In [58]: len(yavgnblocks)
```

```
Out[58]: 16
```

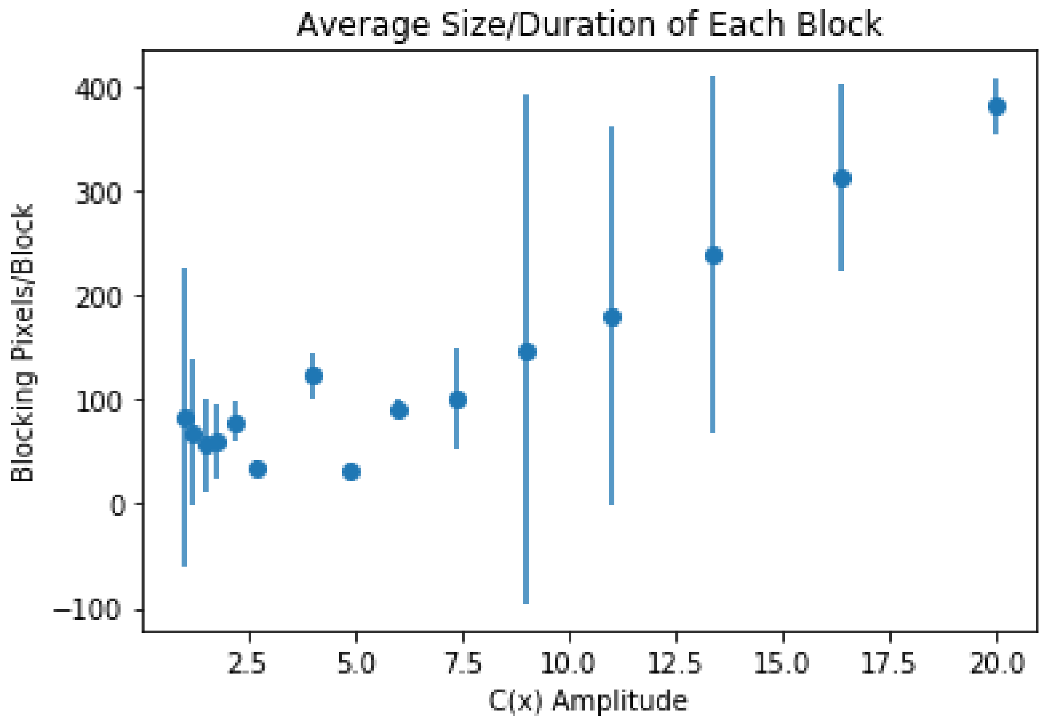
```
In [59]: plt.errorbar(ampls,yavgnblocks,yerr=ystdnblocks,fmt='o')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Average Number of Blocking Events")
         plt.title("Blocking Frequency")
         plt.savefig("cx_nblocks.png",bbox_inches='tight')
```



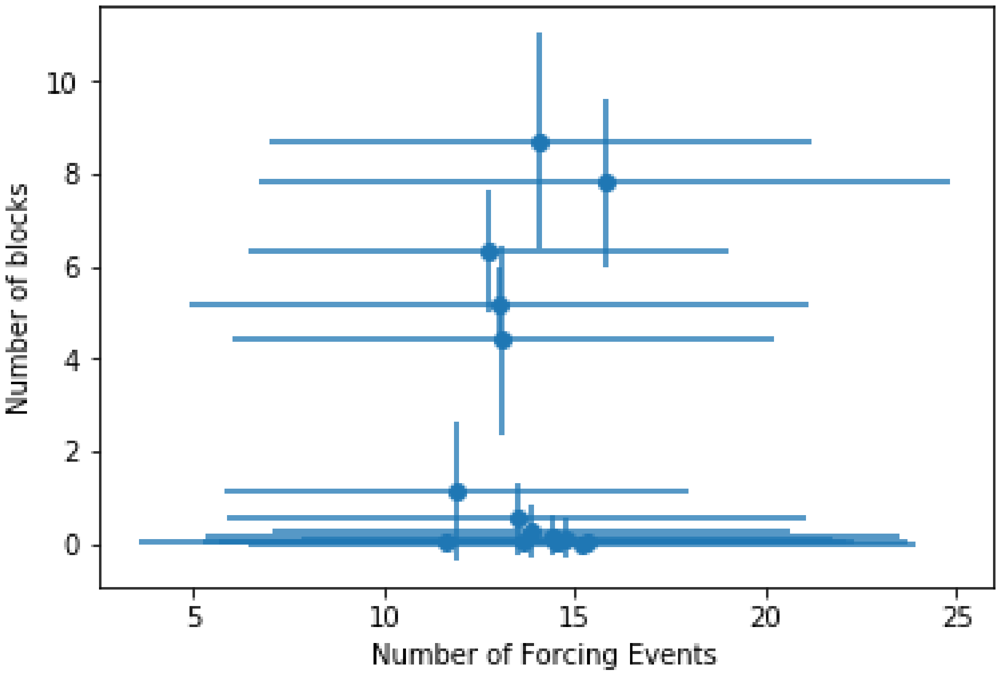

```
In [60]: plt.errorbar(ampls,yavgblocksz,yerr=ystdblocksz,fmt='o')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Blocking Pixels")
         plt.title("Prevalence of Blocking")
         plt.savefig("cx_blocksize.png",bbox_inches='tight')
```




```
In [61]: plt.errorbar(ampls,np.array(yavgblocksz)/np.array(yavgnblocks),
                      yerr=np.sqrt(np.array(ystdnblocks)**2+np.array(ystdblocksz)**2),
                      fmt='o')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Blocking Pixels/Block")
         plt.title("Average Size/Duration of Each Block")
         plt.savefig("cx_avgblocksize.png",bbox_inches='tight')

/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.
```



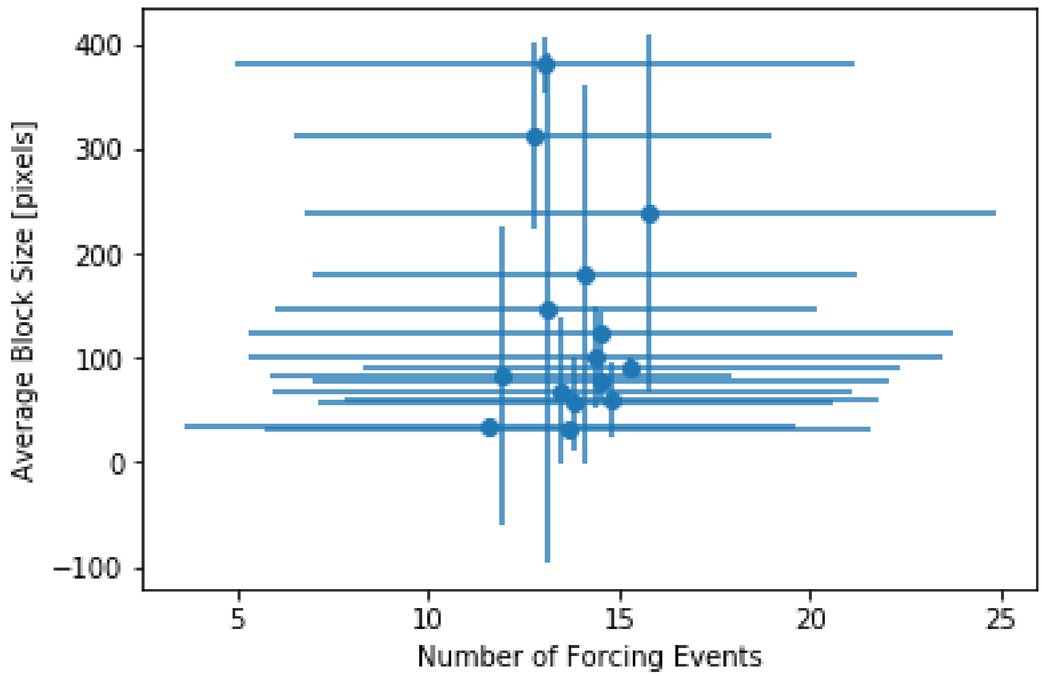

```
In [62]: plt.errorbar(ynevents,yavgnblocks,xerr=ystdevents,yerr=ystdnblocks,fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Number of blocks")

Out[62]: Text(0,0.5,'Number of blocks')
```




```
In [63]: plt.errorbar(ynevents,np.array(yavgblocksz)/np.array(yavgnblocks),
                      xerr=ystdevents,yerr=np.sqrt(np.array(ystdnblocks)**2+\
                                                   np.array(ystdblocksz)**2),
                      fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Average Block Size [pixels]")
```

```
/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.
```

```
Out[63]: Text(0,0.5,'Average Block Size [pixels]')
```



1.0.6 What if we vary the reference flow speed β?

```
In [97]: betas = cxb["var"]
```

```
In [98]: bavgnblocks = []
         bavgblocksz = []
         bstdnblocks = []
         bstdblocksz = []
         bavgnblockspe = []
         bavgblockszpe = []
         bstdnblockspe = []
         bstdblockszpe = []
         bnevents = []
         bstdevents = []
```

```
In [99]: len(cxb["data"])
```

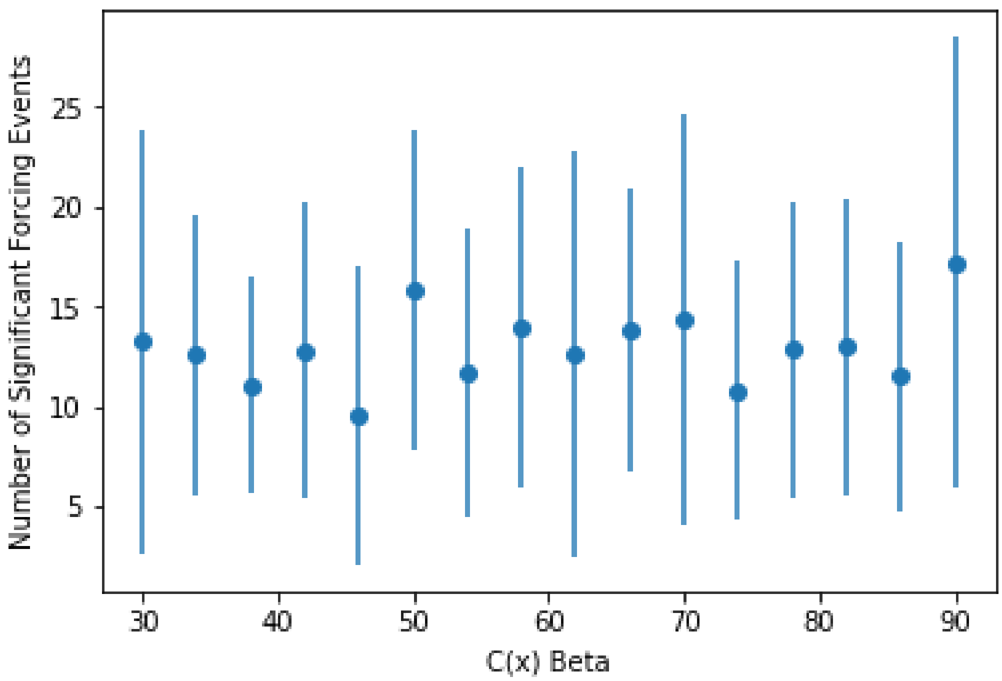
```
Out[99]: 16
```

```
In [100]: for run in cxb["data"]:
              bavgnblocks.append(run["avg_nblocks"])
              bavgblocksz.append(run["avg_blocksize"])
              bstdnblocks.append(run["std_nblocks"])
              bstdblocksz.append(run["std_blocksize"])
              bavgnblockspe.append(run["avg_nblocks_perevent"])
```

```
        bavgblockszpe.append(run["avg_blocksize_perevent"])
        bstdnblockspe.append(run["std_nblocks_perevent"])
        bstdblockszpe.append(run["std_blocksize_perevent"])
        eventsn = np.zeros(20)
        for n in range(0,20):
            eventsn[n] = len(run["forcing_coords"][n][0])
        bnevents.append(np.mean(eventsn))
        bstdevents.append(np.std(eventsn))
```

```
In [101]: plt.errorbar(betas,bnevents,yerr=bstdevents,fmt='o')
          plt.xlabel("C(x) Beta")
          plt.ylabel("Number of Significant Forcing Events")
```

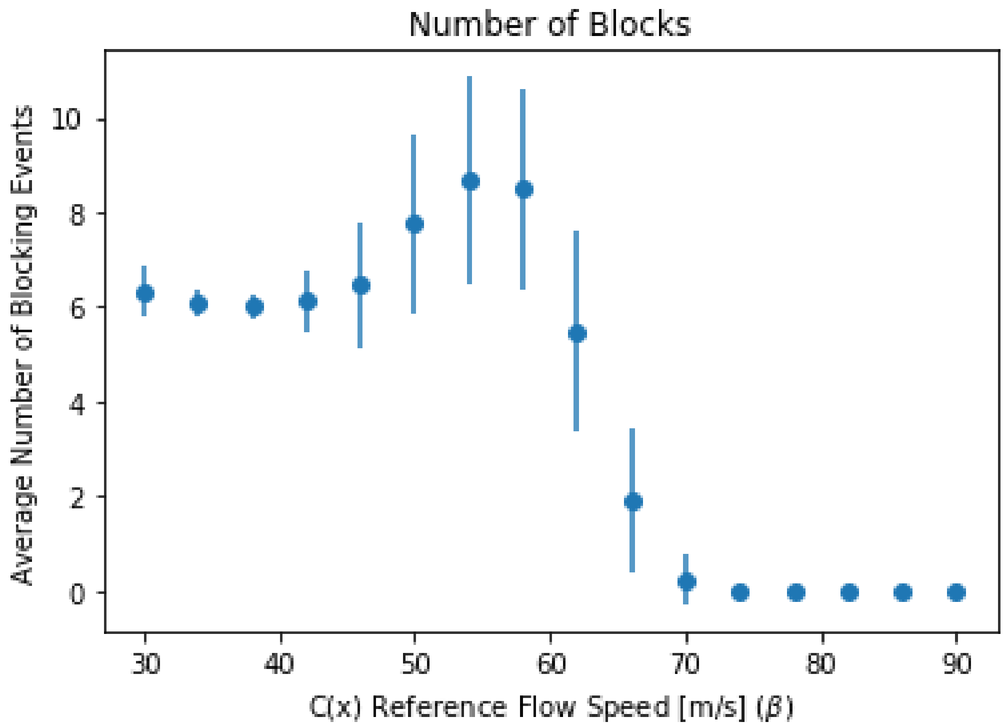
```
Out[101]: Text(0,0.5,'Number of Significant Forcing Events')
```



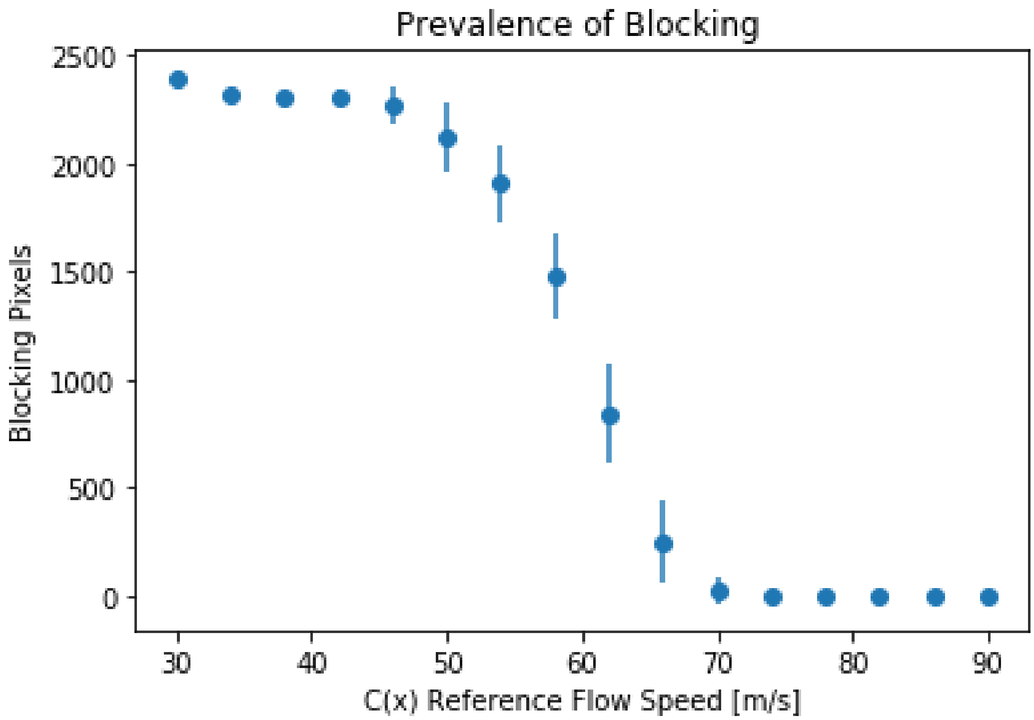

```
In [102]: len(bavgnblocks)
```

```
Out[102]: 16
```

```
In [104]: plt.errorbar(betas,bavgnblocks,yerr=bstdnblocks,fmt='o')
          plt.xlabel("C(x) Reference Flow Speed [m/s] ($\\beta$)")
          plt.ylabel("Average Number of Blocking Events")
          plt.title("Number of Blocks")
          plt.savefig("beta_nblocks.png",bbox_inches='tight')
```

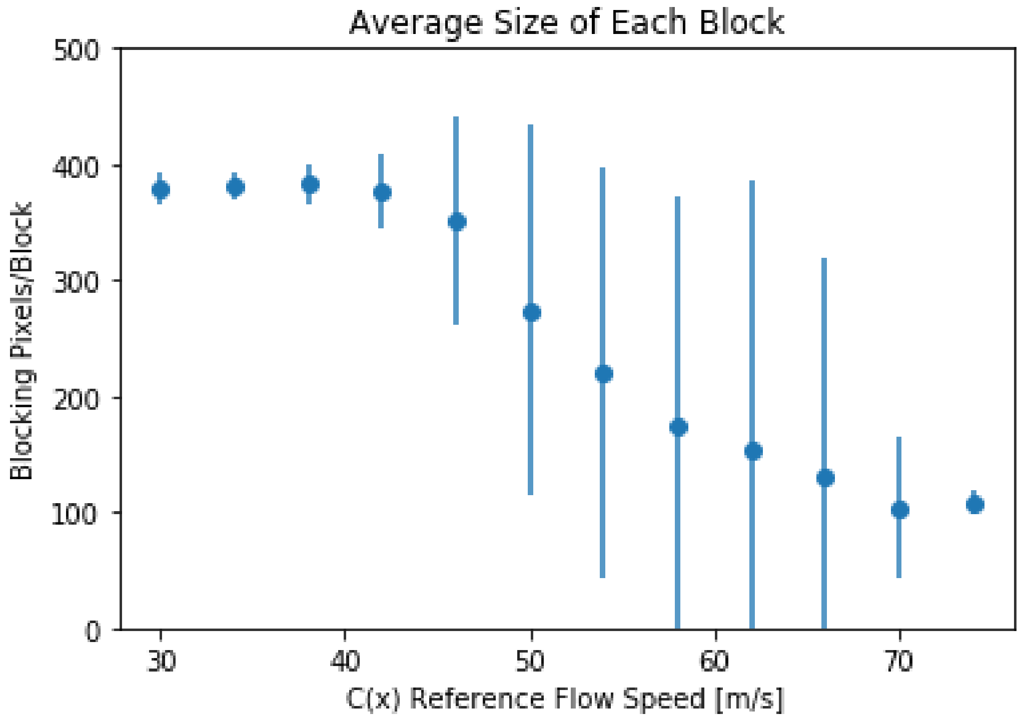



```
In [71]: plt.errorbar(betas,bavgblocksz,yerr=bstdblocksz,fmt='o')
         plt.xlabel("C(x) Reference Flow Speed [m/s]")
         plt.ylabel("Blocking Pixels")
         plt.title("Prevalence of Blocking ")
         plt.savefig("beta_blocksize.png",bbox_inches='tight')
```

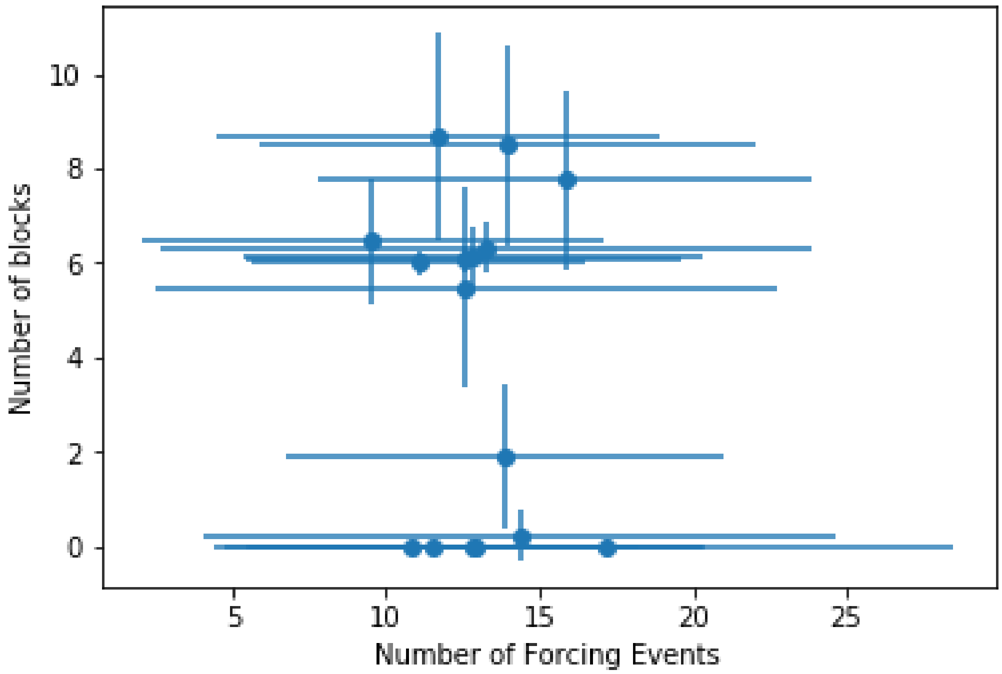



```
In [72]: plt.errorbar(betas,np.array(bavgblocksz)/np.array(bavgnblocks),
                      yerr=np.sqrt(np.array(bstdnblocks)**2+\
                                   np.array(bstdblocksz)**2),fmt='o')
         plt.xlabel("C(x) Reference Flow Speed [m/s]")
         plt.ylabel("Blocking Pixels/Block")
         plt.title("Average Size of Each Block")
         plt.ylim(0,500)
         plt.savefig("beta_avgblocksize.png",bbox_inches='tight')
```

```
/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.
```



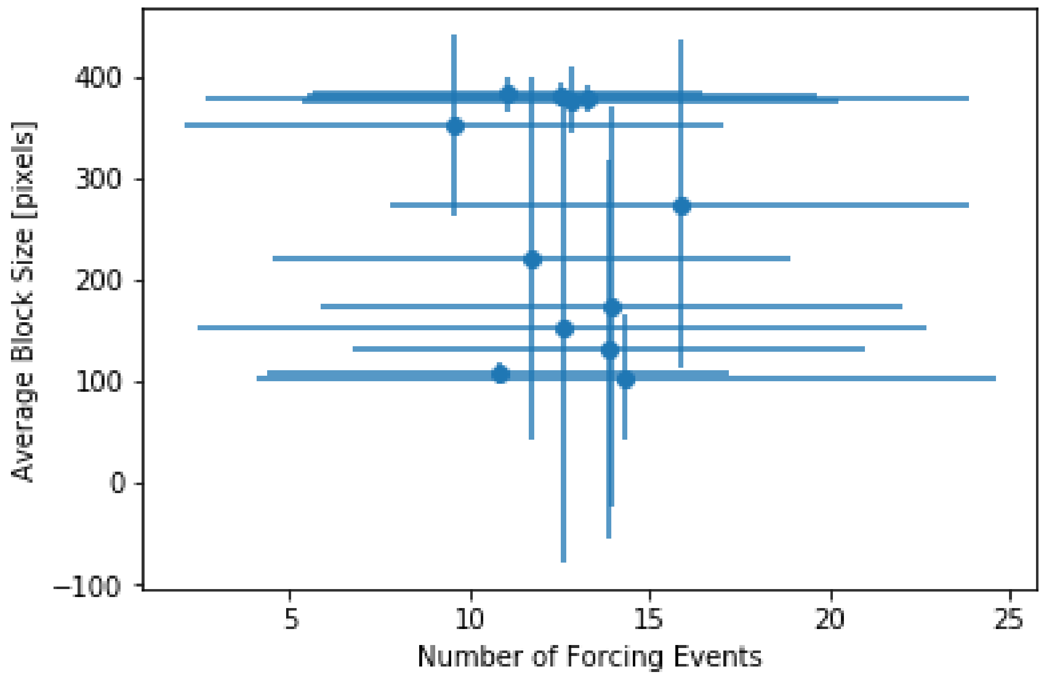

```
In [73]: plt.errorbar(bnevents,bavgnblocks,xerr=bstdevents,yerr=bstdnblocks,fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Number of blocks")

Out[73]: Text(0,0.5,'Number of blocks')
```



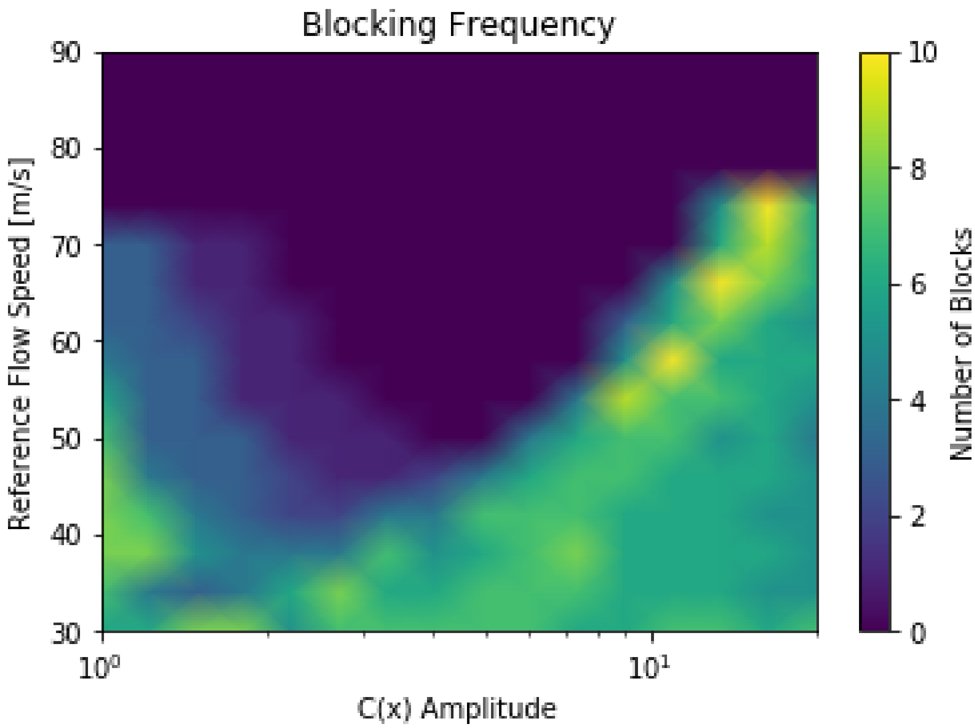

```
In [74]: plt.errorbar(bnevents,np.array(bavgblocksz)/np.array(bavgnblocks),
                      xerr=bstdevents,yerr=np.sqrt(np.array(bstdnblocks)**2+\
                                                   np.array(bstdblocksz)**2),
                      fmt='o')
         plt.xlabel("Number of Forcing Events")
         plt.ylabel("Average Block Size [pixels]")
```

```
/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid v
  """Entry point for launching an IPython kernel.
```

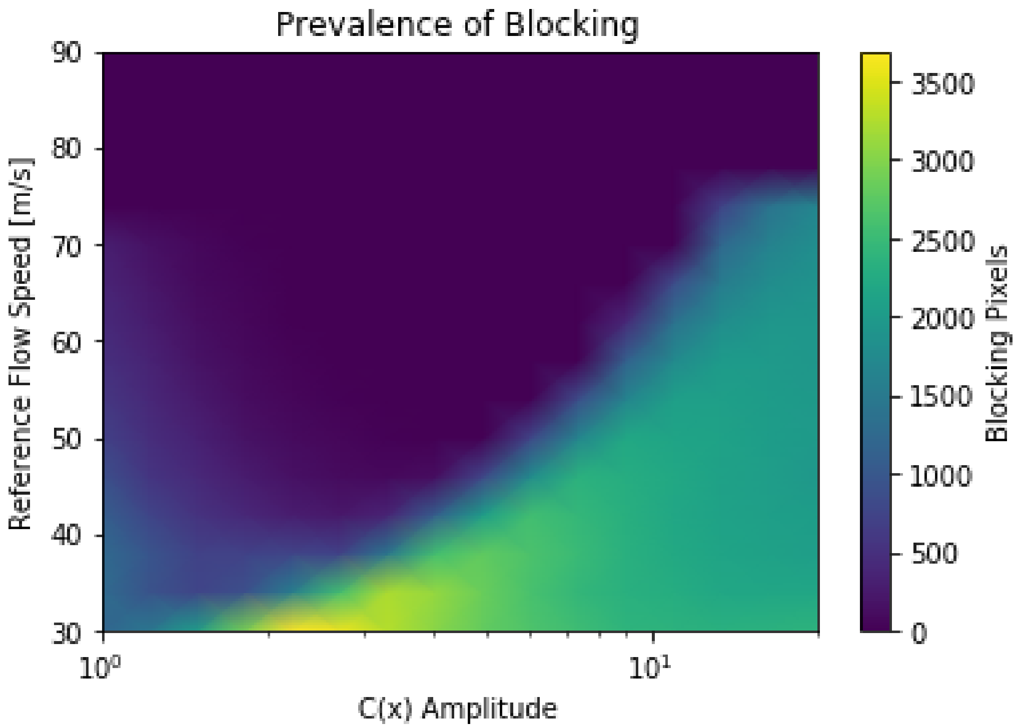
```
Out[74]: Text(0,0.5,'Average Block Size [pixels]')
```



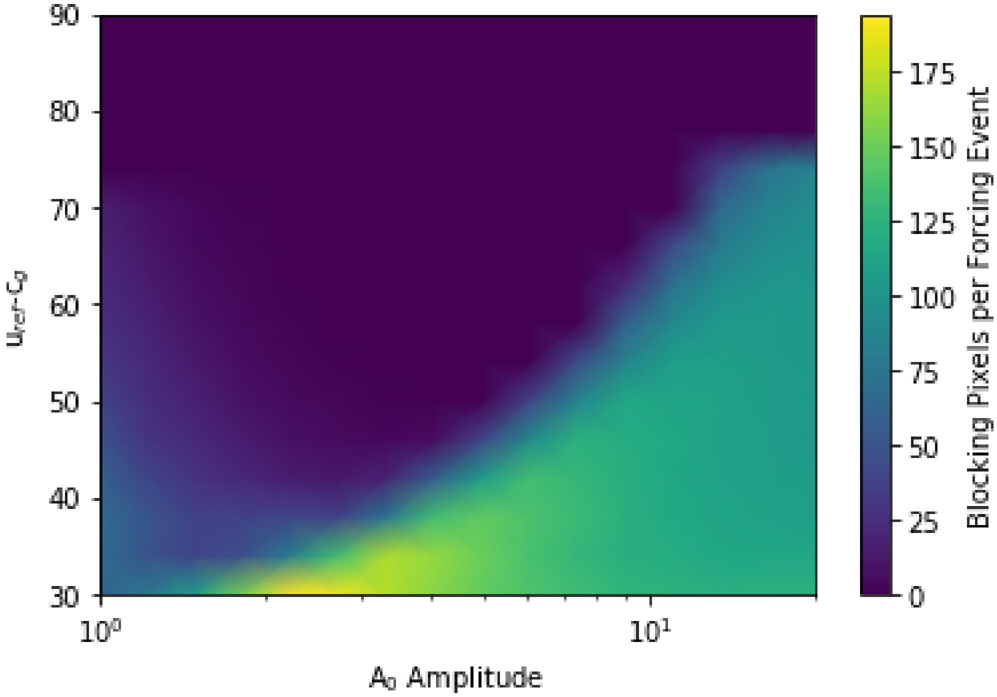

```
In [75]: cxbgrid = np.load("cx_bygrid2.npy",encoding='latin1').item()

In [76]: im=plt.pcolormesh(cxbgrid["a0y"],cxbgrid["beta"],cxbgrid["nblocks"],
                           shading='Gouraud',cmap='viridis')
         plt.xscale('log')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Reference Flow Speed [m/s]")
         plt.colorbar(im,label="Number of Blocks")
         plt.title("Blocking Frequency")
         plt.savefig("grid_nblocks.png",bbox_inches='tight')
```



In [77]:
```
im=plt.pcolormesh(cxbgrid["a0y"],cxbgrid["beta"],cxbgrid["blocksize"],
                  shading='Gouraud',cmap='viridis')
plt.xscale('log')
plt.xlabel("C(x) Amplitude")
plt.ylabel("Reference Flow Speed [m/s]")
plt.colorbar(im,label="Blocking Pixels")
plt.title("Prevalence of Blocking")
plt.savefig("grid_blocksz.png",bbox_inches='tight')
```

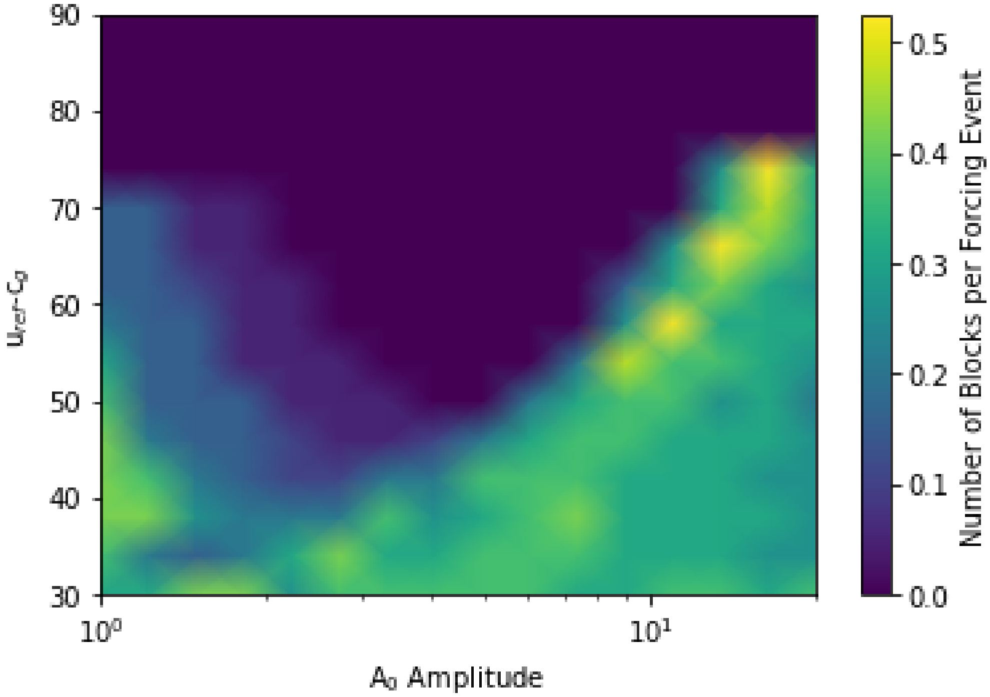



```
In [78]: im=plt.pcolormesh(cxbgrid["a0y"],cxbgrid["beta"],
                          cxbgrid["blocksize_perevent"],shading='Gouraud')
         plt.xscale('log')
         plt.xlabel("A$_0$ Amplitude")
         plt.ylabel("u$_{ref}$-c$_g$")
         plt.colorbar(im,label="Blocking Pixels per Forcing Event")

Out[78]: <matplotlib.colorbar.Colorbar at 0x7f423ae08128>
```



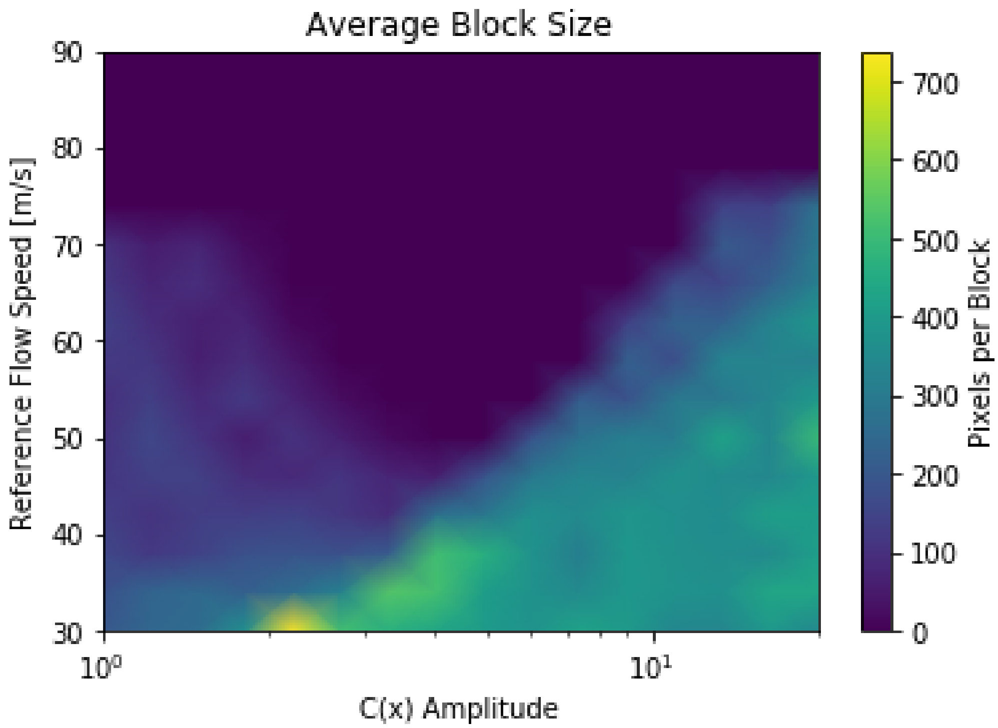
In [79]:
```python
im=plt.pcolormesh(cxbgrid["a0y"],cxbgrid["beta"],
                  cxbgrid["nblocks_perevent"],shading='Gouraud')
plt.xscale('log')
plt.xlabel("A$_0$ Amplitude")
plt.ylabel("u$_{ref}$-c$_g$")
plt.colorbar(im,label="Number of Blocks per Forcing Event")
```

Out[79]: <matplotlib.colorbar.Colorbar at 0x7f423ab59b70>




```
In [80]: im=plt.pcolormesh(cxbgrid["a0y"],cxbgrid["beta"],
                           np.nan_to_num(cxbgrid["blocksize"]/cxbgrid["nblocks"]),
                           shading='Gouraud',cmap='viridis')
         plt.xscale('log')
         plt.xlabel("C(x) Amplitude")
         plt.ylabel("Reference Flow Speed [m/s]")
         plt.colorbar(im,label="Pixels per Block")
         plt.title("Average Block Size")
         plt.savefig("grid_avgblocksize.png",bbox_inches='tight')
```

```
/home/adiv/.local/lib/python3.5/site-packages/ipykernel_launcher.py:2: RuntimeWarning: invalid v
```



1.1 Overall conclusion:

Increasing perturbation or forcing strength promotes blocking. Within the context of $C(x,t)$, increasing the amplitude of the meander and reducing the mean flow speed promote blocking. The transition to quasi-perpetual blocking is relatively sharp, which suggests a potential weakness of the model. The increase in blocking at the limit of low meander and low flow speed corresponds to forcing events being able to cause blocking events wherever they occur (rather than preferentially on background crests), thus why the size of each block is very small in that regime, as well as some false positives, and approaching the regime of spontaneous perpetual blocking without external forcing.

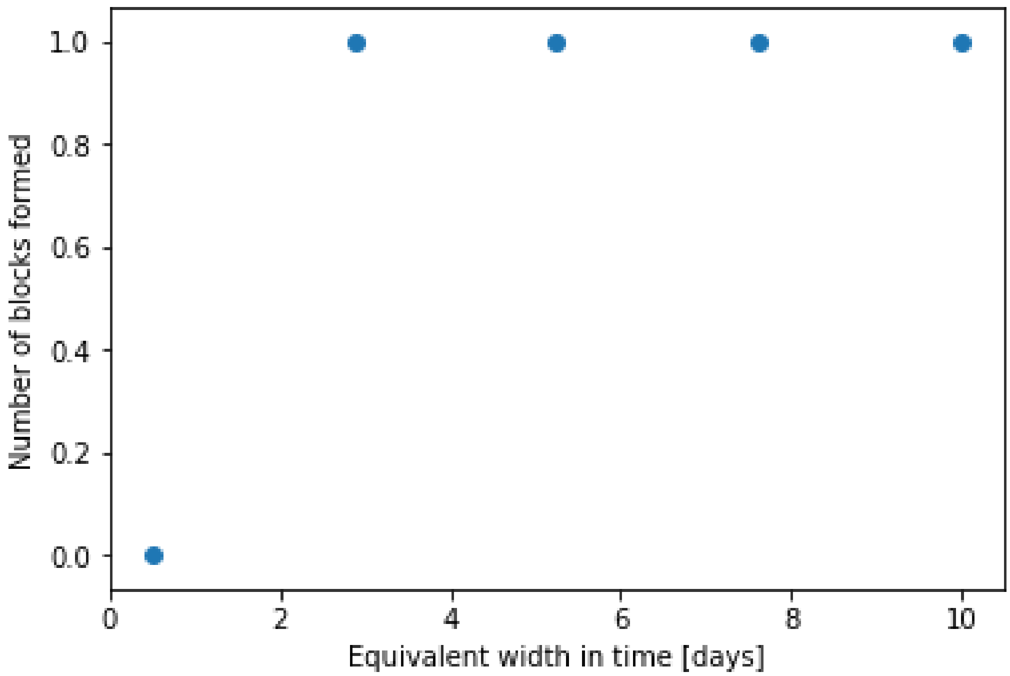
1.1.1 Does blocking depend on the integrated amount of forcing in a given event, or the intensity of the forcing?

We use a single Gaussian forcing, of the same form used in Noboru's paper. We vary the width in x or t, reducing the peak intensity accordingly to maintain the same integrated amount of forcing. If the effect depends on the integrated forcing, there should be no correlation in either number of blocks or size of block, since the integrated forcing is held constant.

```
In [81]: ftw = np.load("forcing_tw.npy",encoding='latin1').item()
         fxw = np.load("forcing_xw.npy",encoding='latin1').item()
         fpk = np.load("forcing_pk.npy",encoding='latin1').item()
```

In [88]:
```
plt.scatter(ftw["tw"],ftw["nblocks"])
plt.xlabel("Equivalent width in time [days]")
plt.ylabel("Number of blocks formed")
```

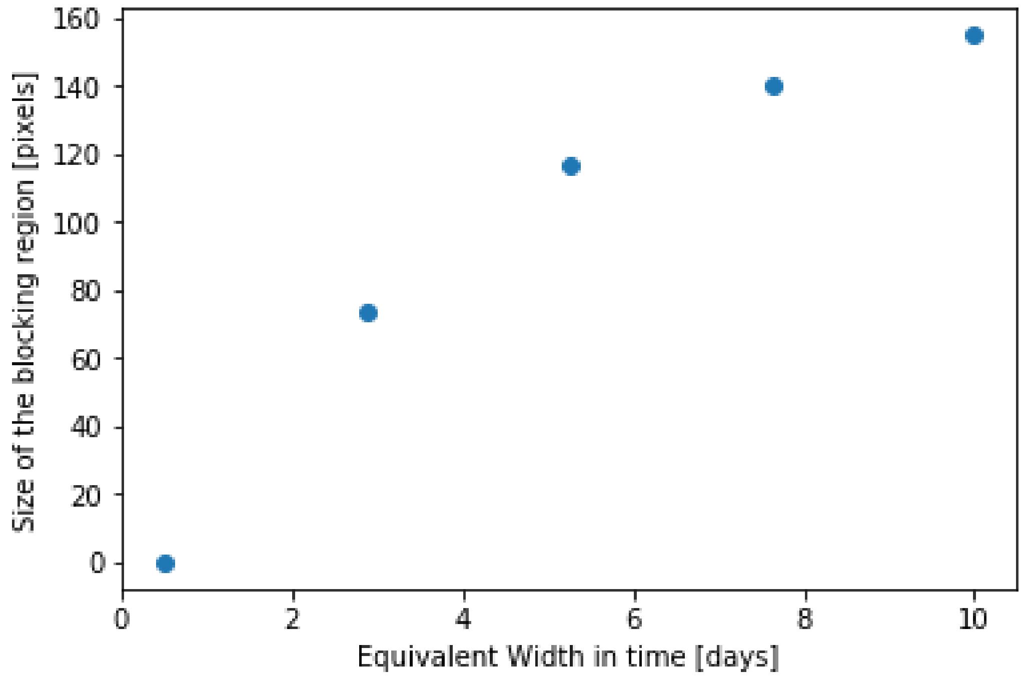
Out[88]:
```
Text(0,0.5,'Number of blocks formed')
```



The integrated forcing in time at least seems to matter.

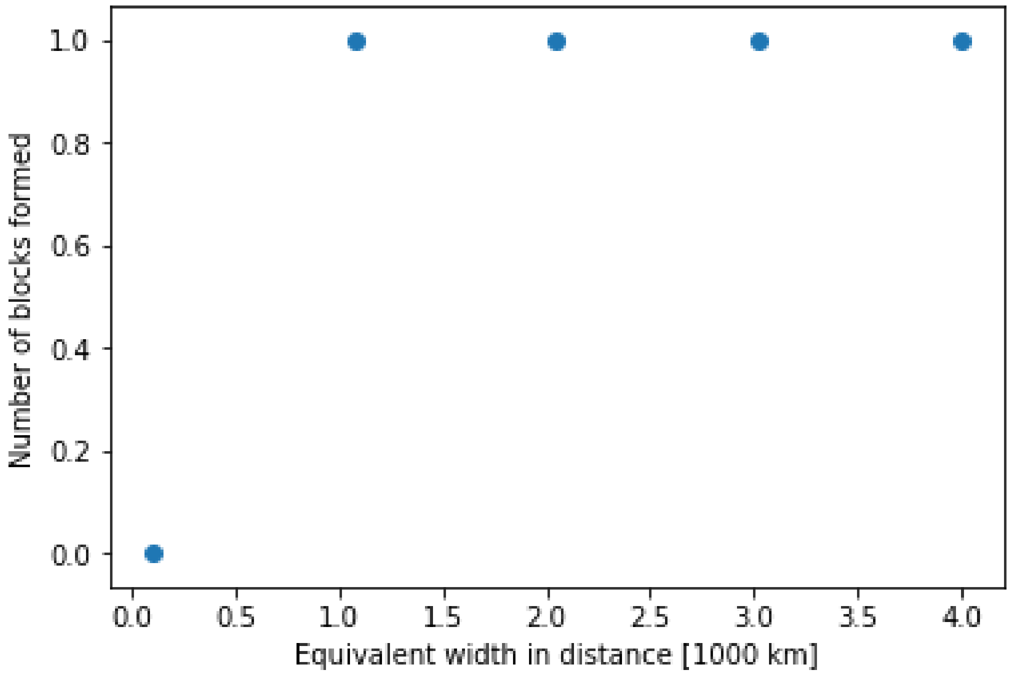
In [89]:
```
plt.scatter(ftw["tw"],ftw["blocksize"])
plt.xlabel("Equivalent Width in time [days]")
plt.ylabel("Size of the blocking region [pixels]")
```

Out[89]:
```
Text(0,0.5,'Size of the blocking region [pixels]')
```

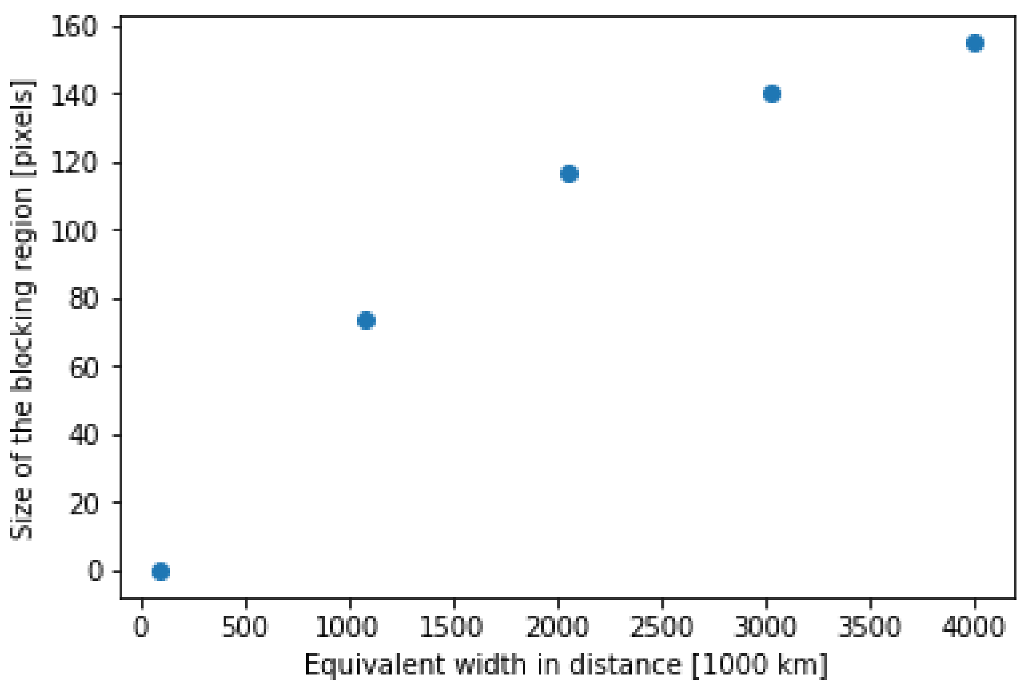


However, longer but less intense events create larger blocks. So the equivalent width at least matters as well, not just the overall integrated forcing.

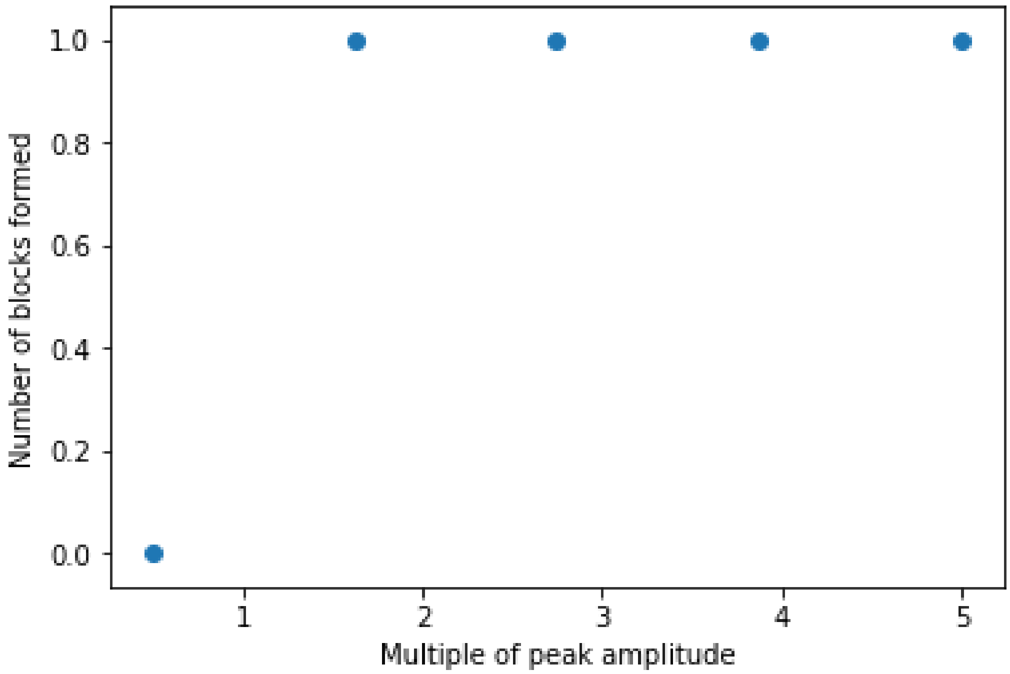
```
In [90]: plt.scatter(fxw["xw"]/1000.0e3,fxw["nblocks"])
         plt.xlabel("Equivalent width in distance [1000 km]")
         plt.ylabel("Number of blocks formed")

Out[90]: Text(0,0.5,'Number of blocks formed')
```



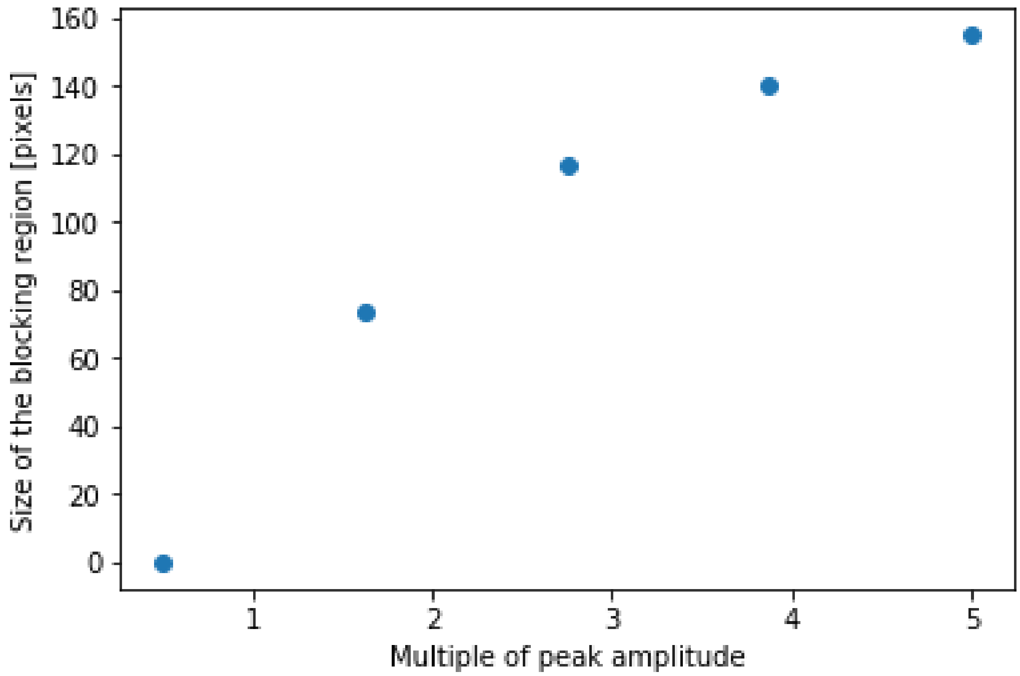

```
In [91]: plt.scatter(fxw["xw"]/1.0e3,fxw["blocksize"])
         plt.xlabel("Equivalent width in distance [1000 km]")
         plt.ylabel("Size of the blocking region [pixels]")

Out[91]: Text(0,0.5,'Size of the blocking region [pixels]')
```



```
In [92]: plt.scatter(fpk["peak"],fpk["nblocks"])
         plt.xlabel("Multiple of peak amplitude")
         plt.ylabel("Number of blocks formed")

Out[92]: Text(0,0.5,'Number of blocks formed')
```




```
In [95]: plt.scatter(fpk["peak"],fpk["blocksize"])
         plt.xlabel("Multiple of peak amplitude")
         plt.ylabel("Size of the blocking region [pixels]")

Out[95]: Text(0,0.5,'Size of the blocking region [pixels]')
```



The conclusion we must therefore draw is that **both** the integrated forcing and the intensity of the forcing matter.