#Levywalk for Autowow Oct2007

```
#------------------------------------------------------------------------------------------------------------------------------------------------------------
#Here are the mathematical functions needed to implement the telescope's levy walk based sky foraging. It is written in python and runs
#on all computer platforms. Required packages include: pyephem, matplotlib, numpy and pylab
#------------------------------------------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env python
# mathematical utilities and some statistical distributions
# makelanguage, part III
# july 2007
#-----------------------------------------------------------------------
import numpy
from math import *
import matplotlib
from pylab import *
#-----------------------------------------------------------------------
#http://mathworld.wolfram.com/Pi.html
pi = 3.14159265359
#-----------------------------------------------------------------------
def sigmoid(beta, n, offset):
	val = (1 / (1 + power(math.e, (-beta * (n-offset)))))
	return(val)
#-----------------------------------------------------------------------
def sigmoid_inv(beta, n, offset):
	#inverted; from max to min
	val = (1 / (1 + power(math.e, (beta * (n-offset)))))
	return(val)
#-----------------------------------------------------------------------
def cauchyrand(a, b):
	#using the quantile function (inverse of the cdf) of the cauchy distribution
	return a + b*tan(pi*(randn()-0.5))
#-----------------------------------------------------------------------
def LevyWalk(mean, std, lowerlimit, upperlimit, location, sfactor, numpoints, doshow=1):
	#with Cauchy Distribution; returns a pair of coordinates in radians
	x = []; xx = []
	y = []; yy = []
	tx = 0; ty = 0
	randvalues = numpy.random.normal(mean, std, numpoints)
	randvalues = randvalues*2*pi #scale values to degrees (0-2pi)
	walklength = ConstrainedCauchy(lowerlimit, upperlimit, location, sfactor, numpoints)
	for i in xrange(0, numpoints):
		y.append(walklength[i]*sin(randvalues[i]))
		x.append(walklength[i]*cos(randvalues[i]))
	if(doshow):
		for i in xrange(0,numpoints):
			tx = tx+x[i]
			ty = ty+y[i]
			xx.append(tx)
			yy.append(ty)
		figure(figsize=(8,8))
		axes([0.1, 0.1, 0.8, 0.8])
		plot(xx,yy, 'bD')
		plot(xx,yy, 'r-')
		text = "levy walk - pruned cauchy distribution (n=" + str(numpoints) + " ,scale factor=" + str(sfactor) + ")"
		title(text)
		grid(True)
		show()
	return(x,y)
#-----------------------------------------------------------------------
def CauchyRandomVariables(location, sfactor, numpoints):
	#generate a series of cauchy random points at a desired location and scale factor (>0)
	cauchyvals=[]
	for i in arange(0, numpoints):
		cauchyvals.append(cauchyrand(location,sfactor))
	return(cauchyvals)
#-----------------------------------------------------------------------
```

```
def ConstrainedCauchy(lowerlimit, upperlimit, location, sfactor, numpoints):
	#dont shorten paths, only elongate selectively, within the interval lower-upper
	ar =[];a=[];b=[];c=[];d=[];e=[];nvals = 0
	while(nvals < numpoints):
		a = CauchyRandomVariables(location, sfactor, numpoints)
		ar = array(a) #convert to array
		b = ar.compress((ar < -lowerlimit).flat)
		b = b.compress((b > -upperlimit).flat)
		c = ar.compress((ar > lowerlimit).flat)
		c = c.compress((c < upperlimit).flat)
		d = concatenate((b,c), axis=0)
		for i in xrange(0, len(d)):
			e.append(d[i])
		nvals = len(e)
		a=[];b=[];c=[];d=[];ar=[]
	pruned_values = e[0:(numpoints)]
	return(pruned_values)
#---------------------------------------------------------------------
```