# Monte Carlo DR results for beam-width errors

# Imported packages

from numpy import \*

import sys

# Define Monte Carlo parameters

nant = 1000

sigma = 1e-1

alpha = 1e-2

rho = 9e-3

n=100

#Initialization

nbasl = nant\*(nant-1)/2

s = zeros(n, dtype=float)

alpha2 = alpha \* alpha

rho2 = rho \* rho

rho\_0 = 0.0

# Loop over Monte Carlo realization

for mc in range(n):

# alphai = random.normal(0,sigma,nant) \* alpha

alphai = random.normal(0,sigma,nant)

# Iterate over baseline

ssum = 0.0

for i in range(nant-1):

for j in range(i, nant):

argval = - ((rho - rho\_0)/(alphai[i])) \*\* 2 - ((rho - rho\_0)/(alphai[j])) \*\* 2

argval += 2 \* rho2 / alpha2

ssum += exp(argval)

s[mc] = ssum/nbasl

# print mc, s[mc]

# Print results

print "Mean intensity: ", s.mean()

drmc = 1 / s.std()

drmemo = (sqrt(nant))\*(alpha/rho)\*(alpha/rho)\*(alpha/sigma)

print "Sigma intensity: ", s.std(), ", DR= ", drmc

print "DR\_MEMO= ", drmemo, ", DR\_ratio (memo/MC) = ", drmemo / drmc

sys.stdout=open("AK\_beam\_Na%s\_sigma%s\_fract%s.txt" % (nant,sigma,rho/alpha),'w')

print "Beam-width Errors:"

print "rho =", rho, ", alpha = ", alpha, ", rho/alpha = ", rho/alpha

print "Na =", nant, ", sigma = ", sigma, ", num mc = ",n

print "Mean intensity: ", s.mean()

print "Sigma intensity: ", s.std()

print "DR\_MC = ", drmc

print "DR\_MEMO= ", drmemo, ", DR\_ratio (memo/MC) = ", drmemo / drmc