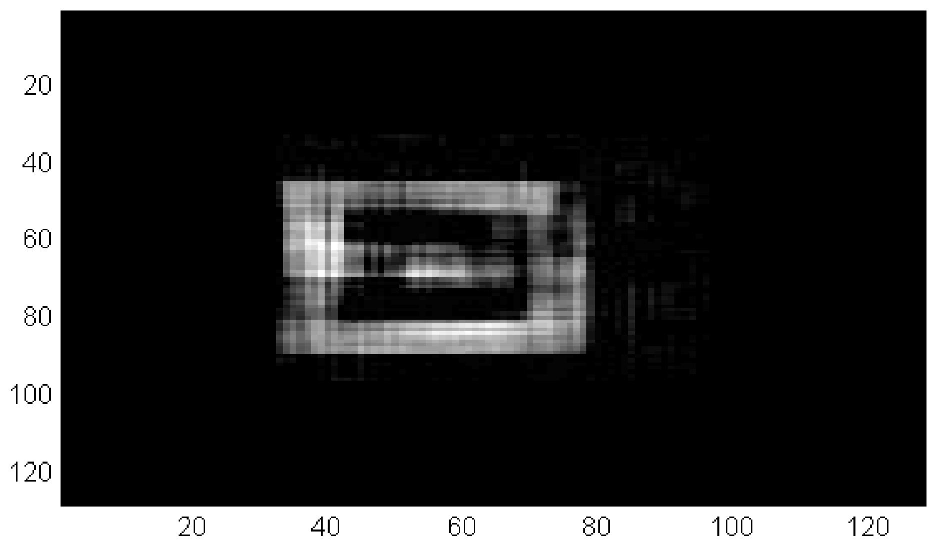
Andy Shi – HW 6

Homework 6

1. The phase recovery folder contains a routine called phaserecovery.m, an image and another utility routine cyl.m. Run phaserecovery routine in matlab and discuss what it is doing step by step.

   The phaserecovery function imports the letter g image (mapped image file) from the file letter_g.m. The information from the picture provides us with the amplitude but not the phase, so we pad the image with zeros and use the Fourier transformation. The Fourier transformation will then find the correct amplitude in the frequency domain and then recover the phase for the time domain. In the time domain, it repads the image with zeros. This then occurs until the max number of iterations if reach (3000x) for which we ends up with the following image with an error of 1.2873e-31.



Andy Shi – HW 6

2. Translate the routines phaserecovery.m and cyl.m to Python and run problem 1 in Python. Do you get the same results as in problem 1? If not, explain.

```
import numpy
import pylab
import random
import cmath

xi = numpy.zeros((128,128))

xi[41:50,41:82]=1
xi[42:86,41:50]=1
xi[78:86,49:86]=1
xi[61:78,77:86]=1
xi[61:70,59:78]=1

in0=xi
N = len(in0)/2
M = 2*N

rect = numpy.zeros((M,M))
rect[32:96,32:96] = 1
in0l = in0

in1 = abs(numpy.fft.fftshift(numpy.fft.fft2(numpy.fft.fftshift(in0l))))
Ao = in1;
randmat = numpy.random.ranf(size = (128,128))
filter = numpy.zeros((M,M))
filter[32:96,32:96] = 1

a = numpy.cos(2*(numpy.pi)*randmat)
a = a*filter

iter = 0
maxiter=3000
E = numpy.zeros((3000,))

while iter < maxiter:
   a1 = a
   A = numpy.fft.fftshift(numpy.fft.fft2(numpy.fft.fftshift(a)))
   PHASE = numpy.angle(A)
   A = Ao *exp(1j*PHASE)
   a = numpy.fft.fftshift(numpy.fft.ifft2(numpy.fft.fftshift(A)))
   P1P2 = a

   a = abs(a)*filter
   P1P2 = a
   err = abs((sum(sum(abs(a1-a)**2))/(sum(sum(abs(Ao)**2)))))
```

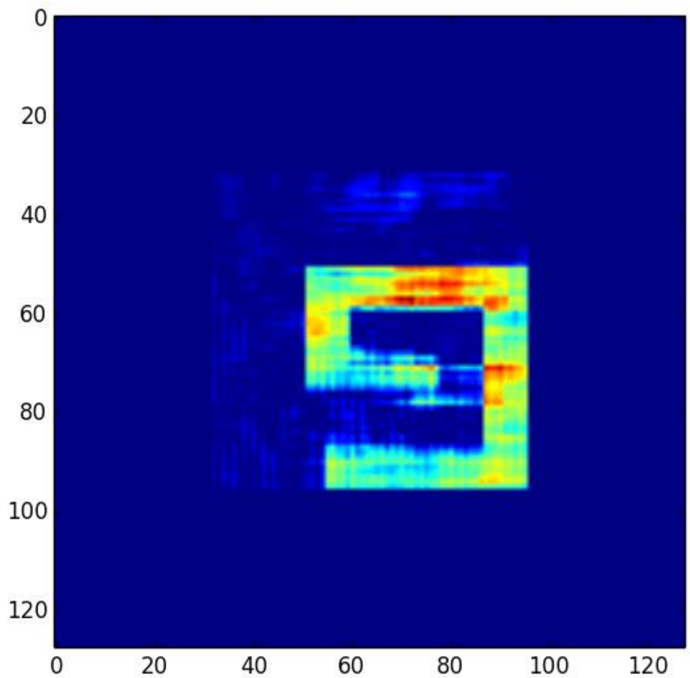
```
    print err
    E[iter,] = err
    iter = iter + 1

imshow(a)
show()
```

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

RESULTS

Final Error: 1.14234523216e-20



3. The 2-D Rosenbrock's function is given by

$$f\left(x_1, x_2\right) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

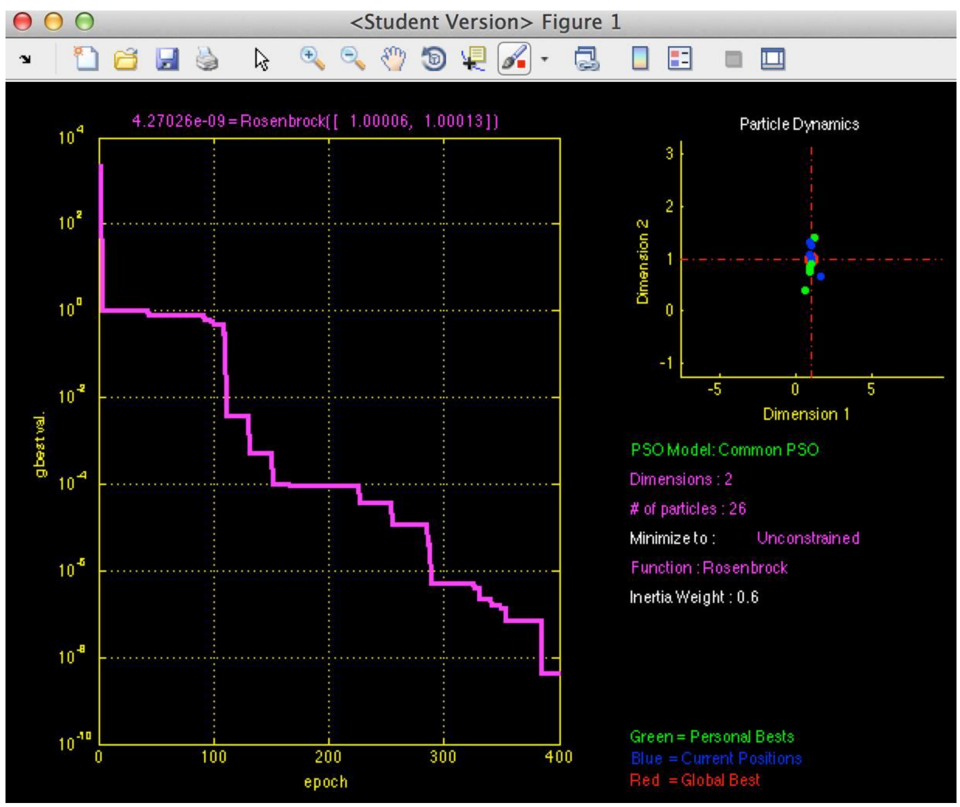
The PSOT toolbox is convenient to gain experience with the PSO algorithm.

(a) Run the routine *DemoPSOBehaviour* to gain experience with global optimization using PSO. Use the Rosenbrock function for minimization.

(b) Repeat the optimization exercise by replacing the Rosenbrock's function by (i) DeJong_f2, (ii) DeJong_f3, (iii) DeJong_f4 functions, which are given in the testfunctions folder of the PSOT toolbox.

(c) Use the continous GA program to repeat part (b). How do the results in parts (b) and (c) compare?

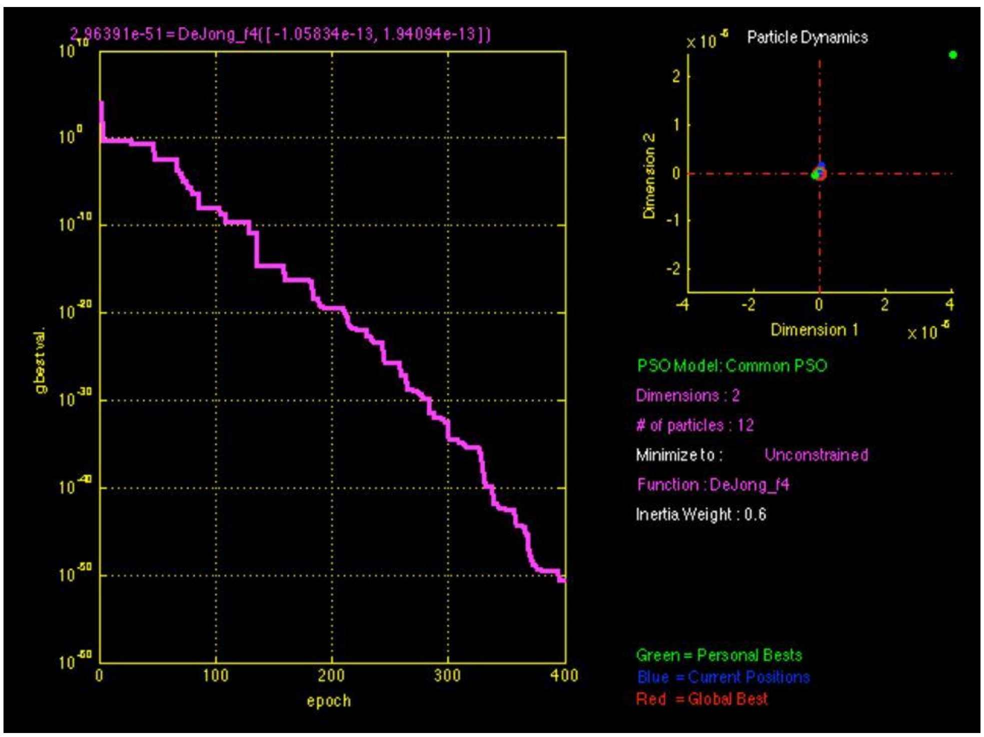
Rosenbrock
Best fit parameters:
 cost = Rosenbrock([input1, input2])
----
 input1 = 1.0001
 input2 = 1.0001
 cost = 4.2703e-09
 mean cost = 11.014



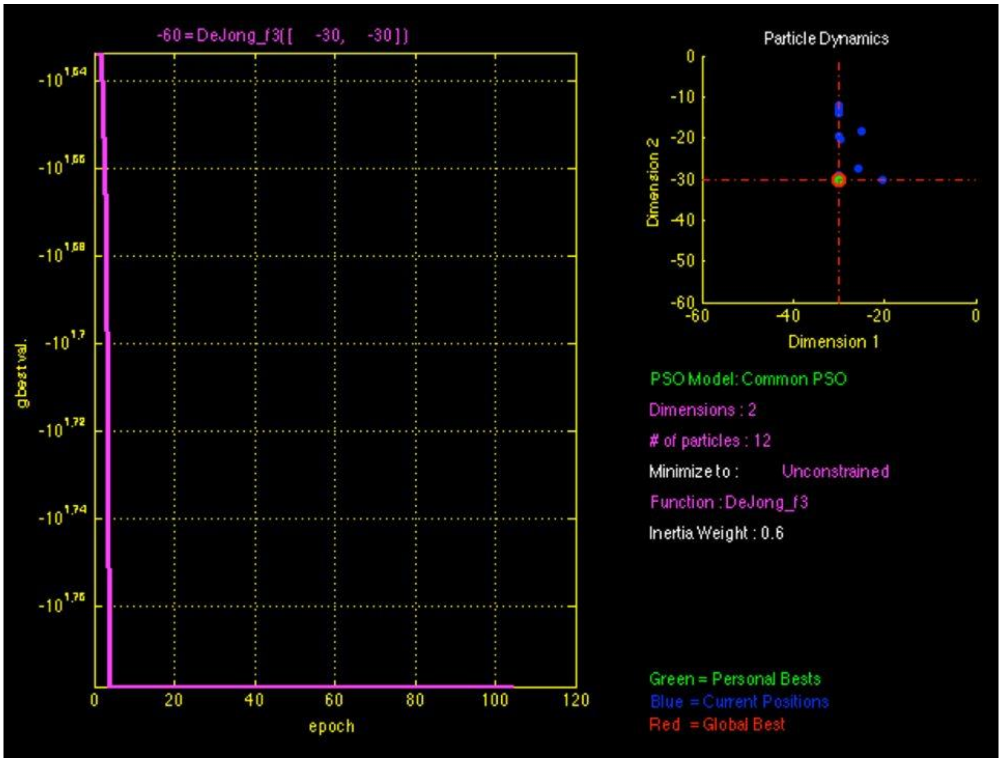
F2:
Best fit parameters:
 cost = DeJong_f2([input1, input2])
----
 input1 = 1.0001
 input2 = 1.0002
 cost = 8.992e-09
 mean cost = 1176.8246
 # of epochs = 400
Best fit parameters:
 cost = DeJong_f3([input1, input2])
----
 input1 = -30
 input2 = -30
 cost = -60
 mean cost = -59.5481
 # of epochs = 104

Andy Shi – HW 6

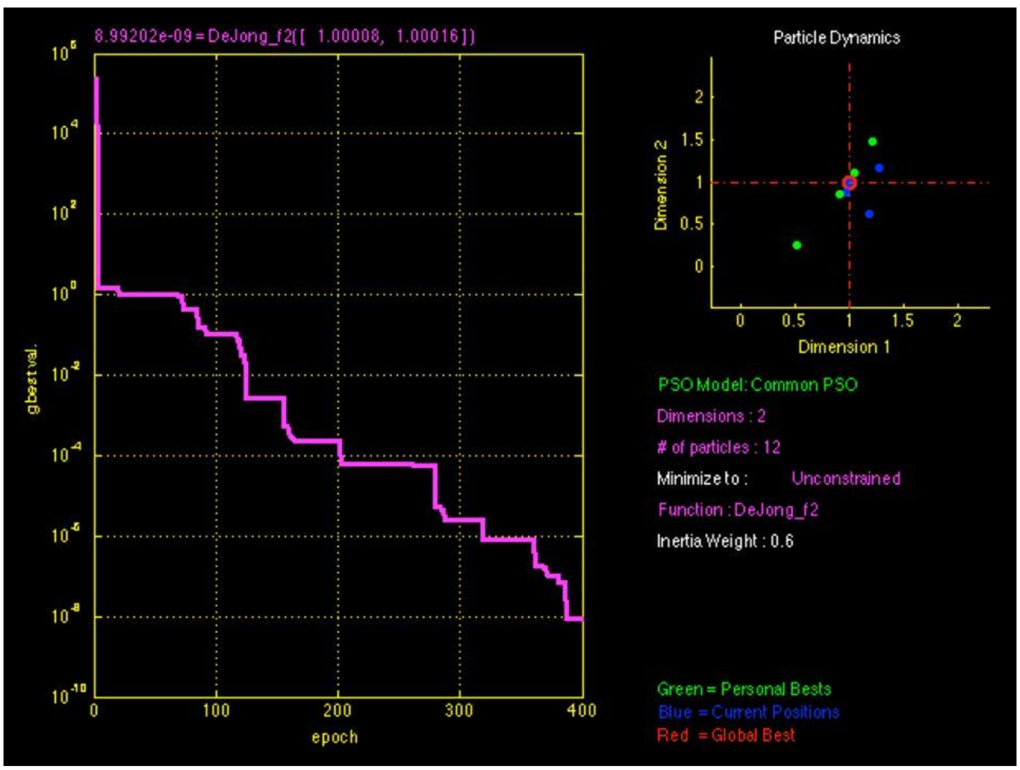
Best fit parameters:
cost = DeJong_f4([input1, input2])
----
input1 = -1.0583e-13
input2 = 1.9409e-13
cost = 2.9639e-51
mean cost = 51.2045
# of epochs = 400



Andy Shi – HW 6



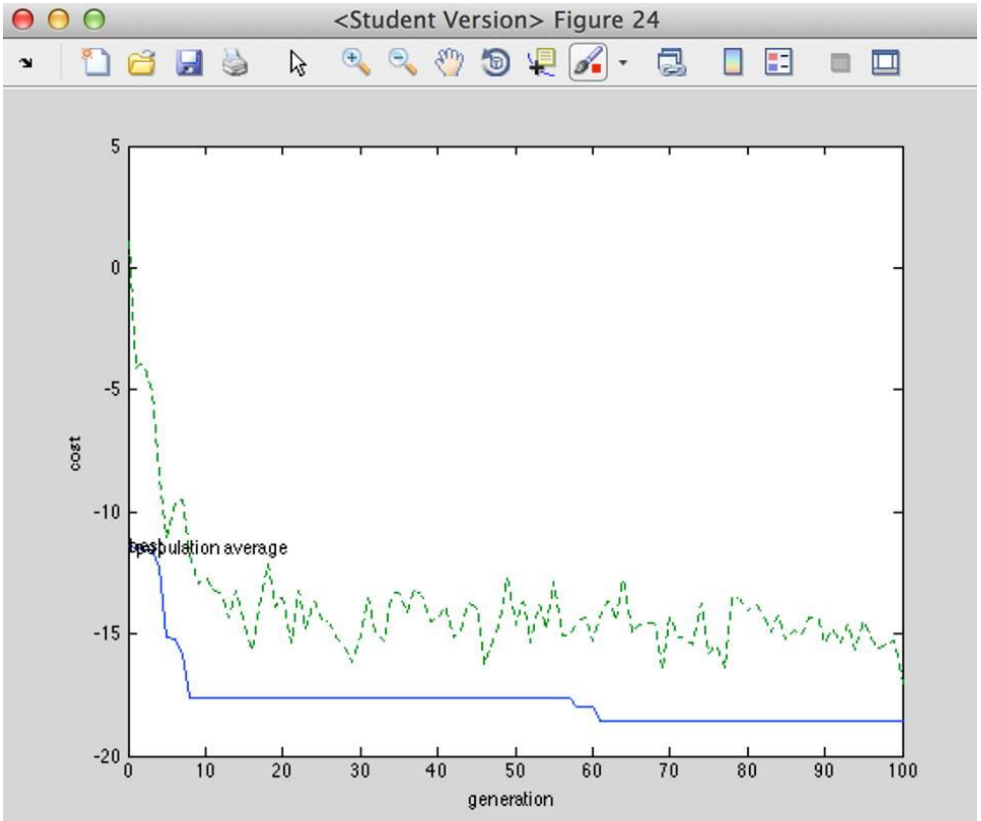
-60 = DeJong_f3([-30, -30])
gbest val.
epoch
Particle Dynamics
Dimension 1
Dimension 2
PSO Model: Common PSO
Dimensions : 2
# of particles : 12
Minimize to : Unconstrained
Function : DeJong_f3
Inertia Weight : 0.6
Green = Personal Bests
Blue = Current Positions
Red = Global Best



8.99202e-09 = DeJong_f2([1.00008, 1.00016])
gbest val.
epoch
Particle Dynamics
Dimension 1
Dimension 2
PSO Model: Common PSO
Dimensions : 2
# of particles : 12
Minimize to : Unconstrained
Function : DeJong_f2
Inertia Weight : 0.6
Green = Personal Bests
Blue = Current Positions
Red = Global Best

Andy Shi – HW 6

optimized function is testfunction
popsize = 12 mutrate = 0.2 # par = 2
#generations=100 best cost=-18.5405
best solution
9.0399   8.6953
continuous genetic algorithm



Comparing the results from B and C,
- The time it takes for the GA to settle is much shorter than the DeJong.
- The number of generation is much shorter on the GA.
- The two different DeJong has similar descend

Andy Shi – HW 6

4. Study the Python GA routines given in Marsland, Chapter 12. Run the four peaks problem by using/modifying the routines given.

Modify the ga.py function to use fourpeaks

```
from pylab import *
from numpy import *
import fourpeaks as fF
```

Next, we modified the run_ga.py file to use fourpeaks

```
import ga
ga = ga.ga(20,'fF.fourpeaks',101,100,-1,'sp',4,True)
ga.runGA()
```

Running the program generates us this image,