Lab Topic 5

PSO codes

- Clone the repository: **GitHub →SDMBIGDAT19** (Store it outside SandBox)
- The codes and slides supplement the textbook
- Lectures delivered at the BigDat19 5th International Winter school on Big Data, Cambridge University, UK (Jan, 2019)
- http://bigdat2019.irdta.eu/

We will look at the following codes in **SDMBIGDAT19/CODES**:

- **r2ss.m:** Helper function; no need to look inside
- **r2sv.m:** Helper function; no need to look inside
- **s2rs.m:** Helper function; no need to look inside
- **s2rv.m:** Helper function; no need to look inside
- **crcbchkstdsrchrng.m:** Helper function; no need to look inside
- **crcbpso.m: Main PSO code that can be applied to any fitness function**
- **crcbpsotestfunc.m:** A benchmark fitness function; Also an example for how to code fitness functions to work with crcbpso.m
- **crcbqcfitfunc.m:** The fitness function for quadratic chirp GLRT (in WGN)
- **crcbqcpso.m**: Applies PSO to the quadratic chirp fitness function
- **test_crcbpso.m:** Test function for crcbpso.m
- **test_crcbqcpso.m:** Test function for crcbqcpso.m

Exercise #1 Part 1

- Read the short user manual **CODES/CodeDoc.pdf**
- The main usage instructions are in the “help” for each function
- The **test_<funcName>.m** scripts show examples of usage for some of the functions
  - The **test_crcbpso.m** script shows how **crcbpso.m** is applied to a benchmark fitness function (defined in **crcbpsotestfunc.m**)

Exercise #1 Part 2

- Understand the concept of structures in Matlab
  - Matlab structures work in the same way as structures in C
  - X = struct('a', 5.0, 'b', 6.0);
  - disp(X.a) will show 5.0
  - disp(X.b) will show 6.0
- Structures offer a convenient way to move a large number of arguments into and out of a function
- Structures also help make your codes future-proof: New versions of codes can use new input arguments while old versions will ignore them

CRCBPSO: The PSO code

% S=CRCBPSO(Fhandle,N)

% Runs local best PSO on the fitness function with handle Fhandle. If Fname

% is the name of the function, Fhandle = @(x) <Fname>(x, FP), where FP is

% the set of parameters needed by Fname.

Example: FP would include the data vector, PSD etc.

% N is the dimensionality of the

% fitness function. The output is returned in the structure S. The field

% of S are:

% 'bestLocation : Best location found (in standardized coordinates)

% 'bestFitness': Best fitness value found

% 'totalFuncEvals': Total number of fitness function evaluations.

Test fitness function

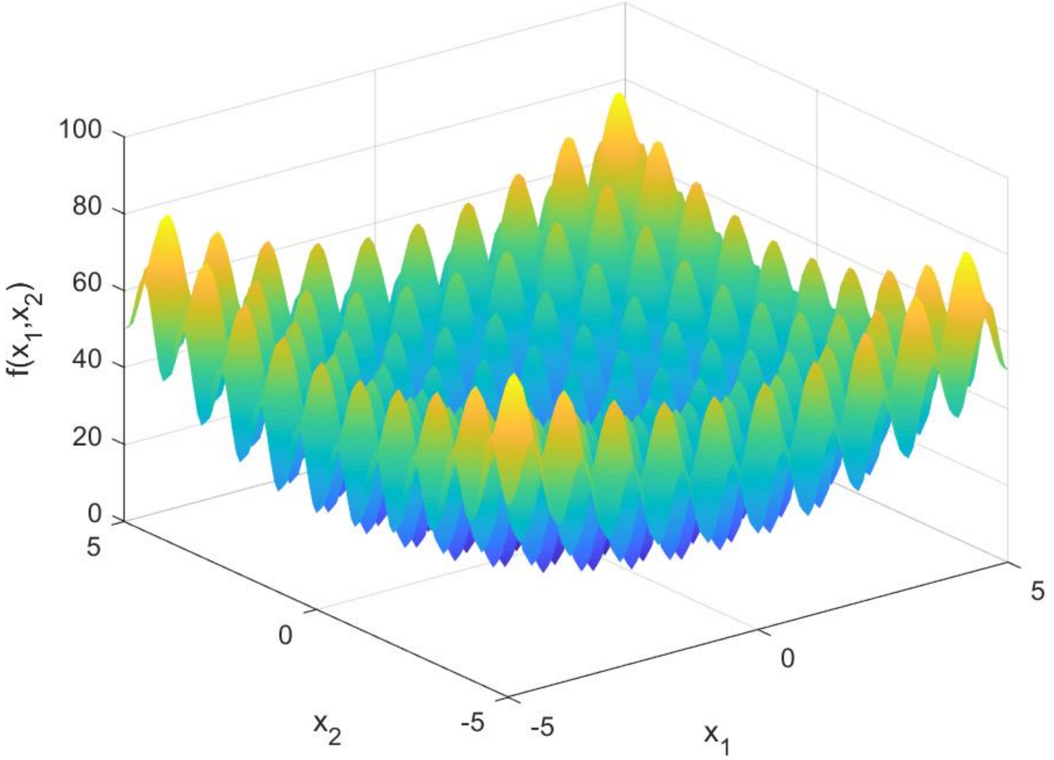
```
F = CRCBPSOTESTFUNC(X,P)
Compute the Rastrigin fitness function for each row of X.  The fitness
values are returned in F. X is standardized, that is 0<=X(i,j)<=1. P has
two arrays P.rmin and P.rmax that are used to convert X(i,j) internally to
actual coordinate values before computing fitness:
```

$$X(:,j) \; -> X(:,j) * (rmax(j) - rmin(j)) + rmin(j)$$

The main part of this function

```
for lpc = 1:nrows
    if validPts(lpc)
    % Only the body of this block should be replaced for different fitness
    % functions
        x = xVec(lpc,:);
        fitVal(lpc) = sum(x.^2-10*cos(2*pi*x)+10);
    end
end
```

Exercise #2



- Run the PSO code on the standard benchmark fitness function **crcbpsotestfunc.m**
- Experiment:
  - Change number of dimensions
  - Change search range
- Challenge: Learn how to use the **surf** function in Matlab and make the plot shown (see Matlab documentation for surf and worked out examples)

Exercise #3

- Code a matlab function for any one benchmark fitness function following the example of **crcbpsotestfunc.m**
  - Different teams can pick different functions
  - Ignore the fitness functions that have red highlights (there are typographical errors in them)
  - Column "D": Dimensionality of search space
  - Column "Feasible Bounds": Range of each coordinate defining the search space, which is a hypercube
- Apply **crcbpso** to the fitness function and find the solution for the global minimum and minimizer

TABLE I

BENCHMARK FUNCTIONS

Equation	Name	D	Feasible Bounds
$f_1 = \sum_{i=1}^{D} x_i^2$	Sphere/Parabola	30	$(-100, 100)^D$
$f_2 = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	Schwefel 1.2	30	$(-100, 100)^D$
$f_3 = \sum_{i=1}^{D-1} \left\{100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right\}$	Generalized Rosenbrock	30	$(-30, 30)^D$
$f_4 = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{x_i}\right)$	Generalized Schwefel 2.6	30	$(-500, 500)^D$
$f_5 = \sum_{i=1}^{D} \left\{x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right\}$	Generalized Rastrigin	30	$(-5.12, 5.12)^D$
$f_6 = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\} - \exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right\} + 20 + e$	Ackley	30	$(-32, 32)^D$
$f_7 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank	30	$(-600, 600)^D$
$f_8 = \frac{\pi}{D}\left\{10\sin^2\left(\pi y_i\right) + \sum_{i=1}^{D-1}\left(y_i - 1\right)^2\left\{1 + 10\sin^2\left(\pi y_{i+1}\right)\right\} + \left(y_D - 1\right)^2\right\}$	Penalized Function P8	30	$(-50, 50)^D$
$+\sum_{i=1}^{D}\mu\left(x_i, 10, 100, 4\right)$			
$y_i = 1 + \frac{1}{4}\left(x_i + 1\right)$			
$\mu\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k\left(-x_i - a\right)^m & x_i < -a \end{cases}$			
$f_9 = 0.1\left\{\sin^2\left(3\pi x_i\right) + \sum_{i=1}^{D-1}\left(x_i - 1\right)^2\left\{1 + \sin^2\left(3\pi x_{i+1}\right)\right\} + \left(x_D - 1\right)^2 \times\right.$	Penalized Function P16	30	$(-50, 50)^D$
$\left.\left\{1 + \sin^2\left(2\pi x_D\right)\right\}\right\} + \sum_{i=1}^{D}\mu\left(x_i, 5, 100, 4\right)$			
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump Camel-back	2	$(-5, 5)^D$
$f_{11} = \left\{1 + \left(x_1 + x_2 + 1\right)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right\} \times$	Goldstein-Price	2	$(-2, 2)^D$
$\left\{30 + \left(2x_1 - 3x_2\right)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right\}$			
$f_{12} = -\sum_{i=1}^{5}\left\{\sum_{j=1}^{4}\left(x_j - a_{ij}\right)^2 + c_i\right\}^{-1}$	Shekel 5	4	$(0, 10)^D$
$f_{13} = -\sum_{i=1}^{7}\left\{\sum_{j=1}^{4}\left(x_j - a_{ij}\right)^2 + c_i\right\}^{-1}$	Shekel 7	4	$(0, 10)^D$

Quadratic Chirp: LR function for WGN

F = CRCBQCFITFUNC(X,P)

Compute the fitness function (~~sum of squared residuals function after maximimzation over the amplitude parameter~~$\rightarrow$ To be replaced by log-likelihood for colored noise) for data containing the quadratic chirp signal at the parameter values in X.

The fitness values are returned in F.

X is standardized, that is 0<=X(i,j)<=1.

The fields P.rmin and P.rmax are used to convert X(i,j) internally before computing the fitness: X(:,j) -> X(:,j)*(rmax(j)-rmin(j)) + rmin(j). $\Rightarrow \theta_i = x_i * (b_i - a_i) + a_i$

The fields P.dataY and P.dataX are used to transport the data and its time stamps. ~~The fields P.dataXSq and P.dataXCb contain the timestamps squared and cubed respectively.~~ You will need an extra field to supply the PSD for colored noise.

[F,R] = CRCBQCFITFUNC(X,P) returns the quadratic chirp coefficients corresponding to the rows of X in R.

[F,R,S] = CRCBQCFITFUNC(X,P) Returns the quadratic chirp signals corresponding to the rows of X in S.

```
for lpc = 1:nVecs
    if validPts(lpc)
    % Only the body of this block should be replaced for different fitness
    % functions
        x = xVec(lpc,:);
        fitVal(lpc) = ssrqc(x, params);
    end
end

%Sum of squared residuals after maximizing over amplitude parameter
function ssrVal = ssrqc(x,params)
%Generate normalized quadratic chirp
phaseVec = x(1)*params.dataX + x(2)*params.dataXSq + x(3)*params.dataXCb;
qc = sin(2*pi*phaseVec);
qc = qc/norm(qc);
```

$\Rightarrow$ norm is the one defined for WGN

```
%Compute fitness
ssrVal = -(params.dataY*qc')^2;
```

$\Rightarrow -\langle \bar{y}, \bar{q}(\Theta) \rangle^2$ for WGN