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

Exercise #1 Part 3

- Understand the concept of **function handle**
- A function handle is a variable that can be used to call a function
- *z=5.0*
- $F = \underbrace{@(x,y)}_{\substack{What\ are\ the\ input\\ variables\ that\ will\ be\\ sent\ to\ F?}} \quad foo(x,z,y)$
- F is a handle to function foo
- $F(2.0, 3.0)$ is the same as $foo(2.0,\ 5.0,\ 3.0)$
- The CRCBPSO code accepts as input the handle to the fitness function to be optimized
- This allows the same PSO code to be run on any fitness function
- Type "function handle" in Matlab's "Search documentation" bar and read more about this feature

Anatomy of SDMBIGDAT19 codes

CRCBPSO: The PSO code

Inputs

- **Handle to fitness function**
  - Fitness function initialized with parameters that will not change from one call to another in PSO
  - Example: Data vector, PSD, sampling frequency
- **Number of search space dimensions** (= Number of parameters to maximize the likelihood Ratio over)
  - Example: 3 D search space for GLRT of quadratic chirp

Output

- **Structure** containing fields:
- Best fitness value found
- Location of the best fitness (<u>in standardized coordinates</u>)
- Number of fitness evaluations (remember that particle fitness is not evaluated when they are outside the search space)

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 be a structure that includes 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.

Simple fitness function

```
F = CRCBPSOTESTFUNC(X,P)
Compute the Rastrigin fitness function for each row of
X. X is standardized, that is 0<=X(i,j)<=1.
```

Example of the matrix X for a 2D space		
	Parameter 1	Parameter 2
Location 1	0.1	0.8
Location 2	0.5	0.2

```
The fitness values are returned in F.
```

Example of F for the above X	
	Fitness
Location 1	10.0
Location 2	2.0

Simple fitness function

```
F = CRCBPSOTESTFUNC(X,P)

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:
```

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

Example for 2D space: P.rmin = [-5,5]; P.rmax = [5, 10] $\Rightarrow$ range for parameter 1 is [-5,5] and for parameter 2 is [5,10]

- The fields rmin and rmax are required for any fitness function
- In your fitness function, the structure P will need more fields. Example: data, PSD, etc.

Part where the input standardized coordinates are converted to real coordinates

```
%Check for out of bound coordinates and flag them
validPts = crcbchkstdsrchrng(xVec);
%Set fitness for invalid points to infty
fitVal(~validPts)=inf;
%Convert valid points to actual locations
xVec(validPts,:) = s2rv(xVec(validPts,:),params);
```

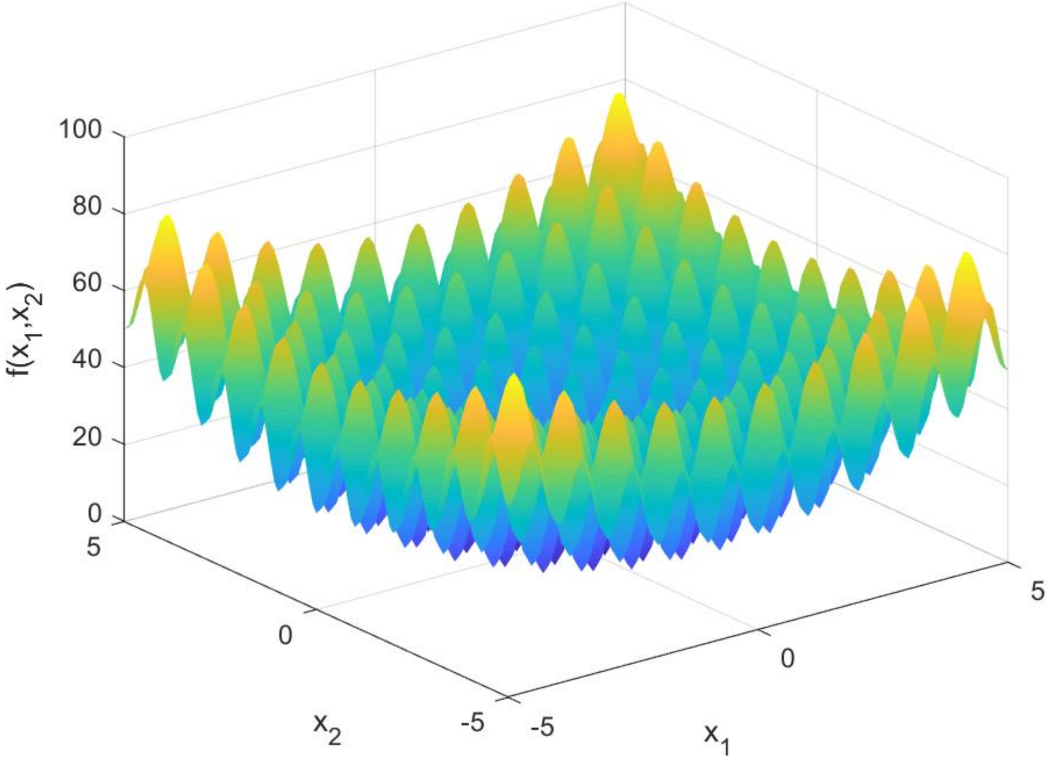
Simple fitness function

```
F = CRCBPSOTESTFUNC(X,P)


                        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 **test_crcbpso.m**: it runs **crcbpso** on the standard benchmark fitness function **crcbpsotestfunc.m**
- Experiment:
  - Change number of dimensions
  - Change search range
- Optional challenge: Learn how to use the **surf** function in Matlab and make the plot shown (see Matlab documentation for surf and worked out examples)

Optional Exercise

- 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	$(-600, 600)^D$
$f_8 = \frac{\pi}{D}\left\{10\sin^2\left(\pi y_i\right) + \sum_{i=1}^{D-1}\left(y_i - 1\right)^2\left\{1 + 10\sin^2\left(\pi y_{i+1}\right)\right\} + \left(y_D - 1\right)^2\right\}$ $+\sum_{i=1}^{D}\mu\left(x_i, 10, 100, 4\right)$ $y_i = 1 + \frac{1}{4}\left(x_i + 1\right)$ $\mu\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k\left(-x_i - a\right)^m & x_i < -a \end{cases}$	Penalized Function P8	30	$(-50, 50)^D$
$f_9 = 0.1\left\{\sin^2\left(3\pi x_i\right) + \sum_{i=1}^{D-1}\left(x_i - 1\right)^2\left\{1 + \sin^2\left(3\pi x_{i+1}\right)\right\} + \left(x_D - 1\right)^2 \times\right.$ $\left.\left\{1 + \sin^2\left(2\pi x_D\right)\right\}\right\} + \sum_{i=1}^{D}\mu\left(x_i, 5, 100, 4\right)$	Penalized Function P16	30	$(-50, 50)^D$
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump Camel-back	2	$(-5, 5)^D$
$f_{11} = \left\{1 + \left(x_1 + x_2 + 1\right)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right\} \times$ $\left\{30 + \left(2x_1 - 3x_2\right)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right\}$	Goldstein-Price	2	$(-2, 2)^D$
$f_{12} = -\sum_{i=1}^{5}\left\{\sum_{j=1}^{4}\left(x_j - a_{ij}\right)^2 + c_i\right\}^{-1}$	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	$(0, 10)^D$

Bratton, Kennedy, 2007

Main Exercise

Using PSO to get GLRT for the Quadratic Chirp in Colored Gaussian Noise

Plan

- Develop code for the fitness function
- Develop code for calling PSO on the fitness function with multiple independent runs and picking the results from the best run
- Create a data realization and use codes to obtain GLRT using PSO
- Display results

Modify code of fitness function: **crcbqcfitfunc (new name)** → Modify code of function calling PSO: **crcbqcpso (new name)** → Supply data to crcbqcpso and… magic!

Fitness function

Preliminaries

- You have already coded the fitness function required for quadratic chirp in colored Gaussian noise in Lab Topic 4
- You have to now **cast it into a form that can be called by the PSO code**
- This is best done by taking the fitness function already coded for WGN and modifying it by replacing some parts

Fitness function

F = ~~CRCBQCFITFUNC~~<New Name>(X,P)

Compute the fitness function (~~sum of squared residuals function after maximimzation over the amplitude parameter~~→ To be replaced by log-likelihood ratio for colored noise maximized over amplitude) 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). $\Rightarrow \theta_i = x_i * (b_i - a_i) + a_i$

Fitness function

F = ~~CRCBQCFITFUNC~~<New Name>(X,P)

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. Converts X to R

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

```
% Only the body of this block should be replaced for different fitness functions
fitVal(lpc) = ssrqc(x, params);


function ssrVal = ssrqc(x,params)
```
← Can be replaced with the fitness function from Lab Topic 4 Or…

```
%Generate normalized quadratic chirp
```

The signal is generated inside this function for speed (calling crcbgenqcsig.m would be slow because dataX.^2 and dataX.^3 will be recomputed in every call but they need to be computed only once)

```
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 $\rightarrow$ Replace by the norm for given PSD

```
%Compute fitness
ssrVal = -(params.dataY*qc')^2;
```
$\Rightarrow -\langle \bar{y}, \bar{q}(\Theta) \rangle^2$ for WGN $\rightarrow$ Replace by the inner product for given PSD

Calling PSO

CRCBQCPSO: Inputs

```
%O = ~~CRCBQCPPSO~~ <New Name>(I,P,N)
%I is the input struct with the fields given below.
% 'dataY': The data vector (a uniformly sampled time series).
% 'dataX': The time stamps of the data samples.
% 'dataXSq': dataX.^2
% 'dataXCb': dataX.^3
```

New fields needed for colored Gaussian noise:

```
'psd' : PSD values at positive DFT frequencies
'sampFreq': Sampling frequency
% 'rmin', 'rmax': The minimum and maximum values of the three parameters
%                 a1, a2, a3 in the candidate signal:
%                 a1*dataX+a2*dataXSq+a3*dataXCb
```

```
P is the PSO parameter
%struct. Setting P to [] will invoke default parameters (see
CRCBPSO).

% N is the number of independent PSO runs.

%The output is returned in the struct O.
```

CRCBQCPSO: Outputs

```
%The fields of O are:
% 'allRunsOutput': An N element struct array containing results
                   from each PSO run. The fields of this struct are:
%     'fitVal': The fitness value.
%     'qcCoefs': The coefficients [a1, a2, a3].
%     'estSig': The estimated signal.
%     'totalFuncEvals': The total number of fitness evaluations.
% 'bestRun': The best run.
% 'bestFitness': best fitness from the best run.
% 'bestSig' : The signal estimated in the best run.
% 'bestQcCoefs' : [a1, a2, a3] found in the best run.
```

Signal from estimated parameters

```
%Prepare output
fitVal = zeros(1,nRuns);
for lpruns = 1:nRuns
    fitVal(lpruns) = outStruct(lpruns).bestFitness;
    outResults.allRunsOutput(lpruns).fitVal = fitVal(lpruns);
    [~,qcCoefs] = fHandle(outStruct(lpruns).bestLocation);
    outResults.allRunsOutput(lpruns).qcCoefs = qcCoefs;
    estSig = crcbgenqcsig(inParams.dataX,1,qcCoefs);
    estAmp = inParams.dataY*estSig(:);
    estSig = estAmp*estSig;
    outResults.allRunsOutput(lpruns).estSig = estSig;
    outResults.allRunsOutput(lpruns).totalFuncEvals = outStruct(lpruns).totalFuncEvals;
end
```

Convert standardized coordinates to real coordinates

Replace!

When obtaining the maximum of the likelihood ratio over amplitude, we get the solution: $A = \langle \bar{y}, \bar{q}(\Theta) \rangle$ (and substituting this into the likelihood ratio gives$\langle \bar{y}, \bar{q}(\Theta) \rangle^2$)
$\therefore$ estSig ($\bar{s}(\Theta)$) $\rightarrow$ template ($\bar{q}(\Theta)$) by normalization to SNR=1$\rightarrow$ $A = \langle \bar{y}, \bar{q}(\Theta) \rangle$$\rightarrow$ estSig = $A \times \bar{q}(\Theta)$

Data realization

Data realization

- Number of samples: 512
- Sampling frequency: 512 Hz
- **Noise realization**: Toy PSD (**Note change of numbers**)

  noisePSD = @(f) (f>=50 & f<=100).*(f-50).*(100-f)/625 + 1;

  - You will need to generate the PSD at positive DFT frequencies
- **Signal**: Quadratic chirp with the following parameters
  - SNR=10
  - $a_1 = 10, a_2 = 3, a_3 = 3$
- Data: **Noise realization** + **signal**

Presentation of results

Presenting results

- Plot the data
- Plot the true signal on top
- Plot the best estimated signal on top
- (Optional) You can also try to plot the best signals from all the runs of PSO in a different color: This will show the spread in performance of PSO