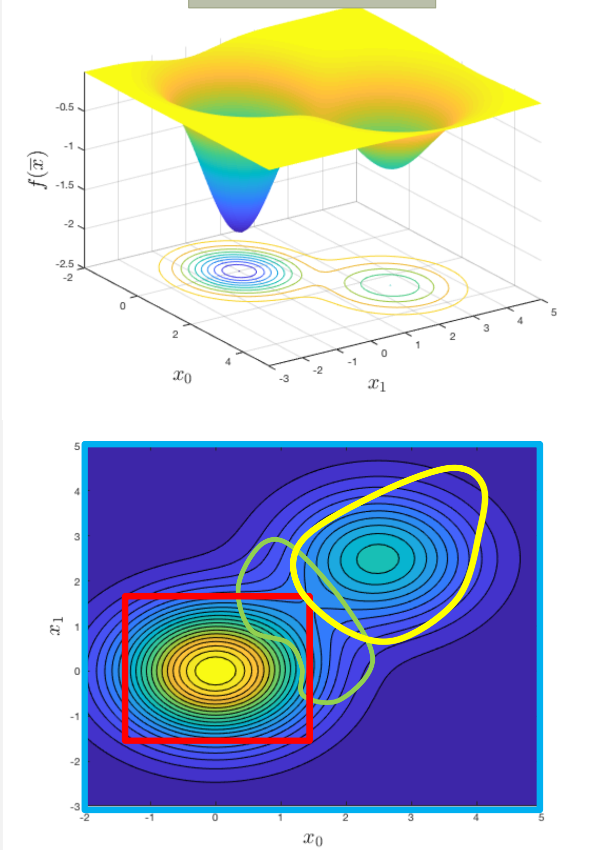
Particle swarm optimization

Selected topics from Chapters 4 and 5 of textbook

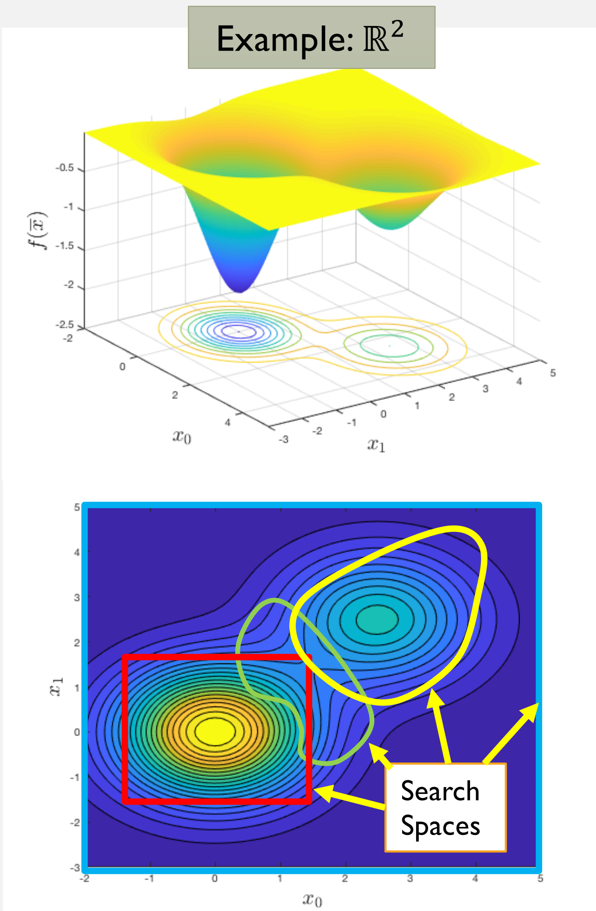
OPTIMIZATION TERMINOLOGY

- Continuous optimization problem: Find the minimum value of a function $f(\bar{x})$ in a specified domain $\bar{x} \in \mathbb{D} \subseteq \mathbb{R}^D$
  - *Maximization of $f(\bar{x})$ is equivalent to minimization of $-f(\bar{x})$

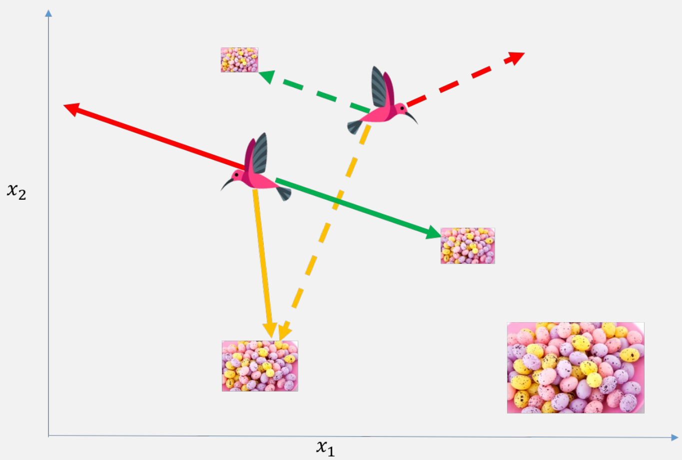
Example: $\mathbb{R}^2$



OPTIMIZATION TERMINOLOGY



- Continuous optimization problem: Find the minimum value of a function $f(\bar{x})$ in a specified domain $\bar{x} \in \mathbb{D} \subseteq \mathbb{R}^D$
  - *Maximization of $f(\bar{x})$ is equivalent to minimization of $-f(\bar{x})$
- $f(\bar{x})$ is called the fitness function
- $\mathbb{D}$ is called the search space
- Example: GLRT involves $\max_{\Theta} \lambda(\Theta)$
  - $\rightarrow \bar{x} \equiv \Theta = (\theta_1, \theta_2, \ldots \theta_D)$ and $f(\bar{x}) \equiv -\lambda(\Theta)$



PARTICLE SWARM OPTIMIZATION

- A swarm intelligence method inspired by the flocking behavior of birds
- Each organism in a swarm, such as a bird flock, is called an agent
- Each agent searches for the best value of some fitness function (e.g., food source density) but also communicates with some of its neighbors regarding what they have found
- Example: If a bird flock is looking for a good food source, each bird looks for a food source on its own but is also influenced by the flock as a whole as conveyed through the movement of its neighbors

PARTICLE SWARM OPTIMIZATION

Initialization

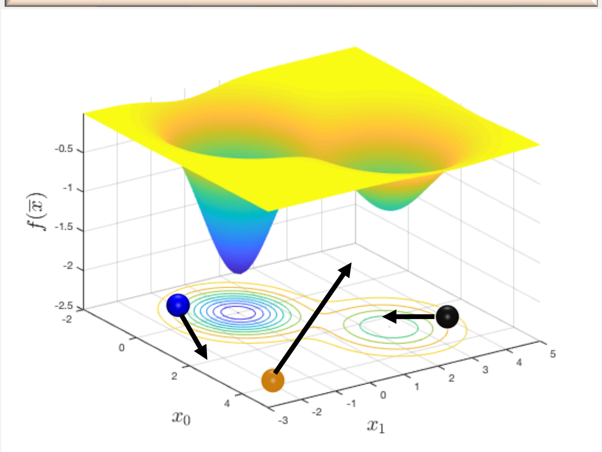
- Particle: agent location
- Particle fitness: Fitness value at location
- Particle "velocity": Displacement vector to new position

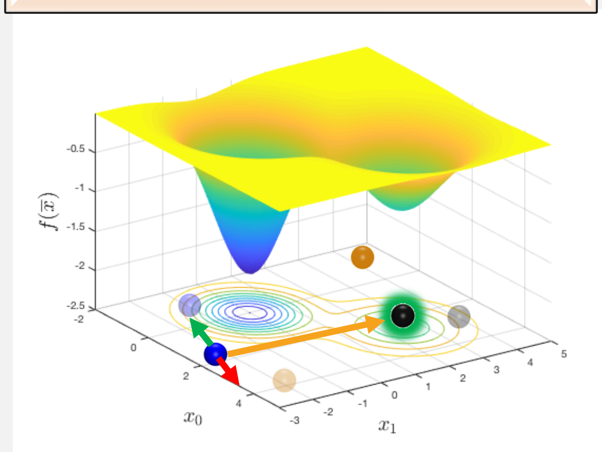
Velocity update

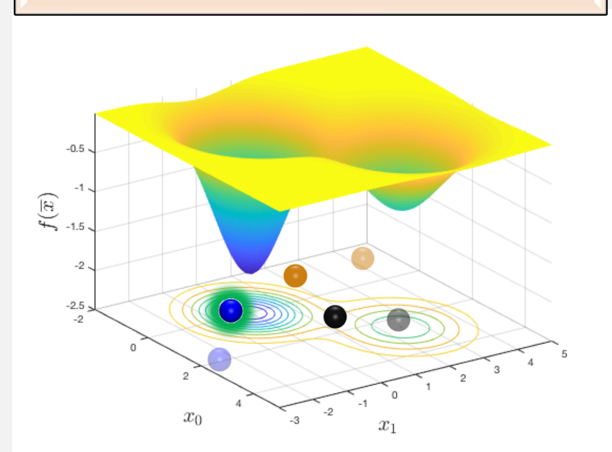
- New velocity: sum of old velocity + acceleration terms
- Acceleration strengths are random

Position update

- Particles move to new positions



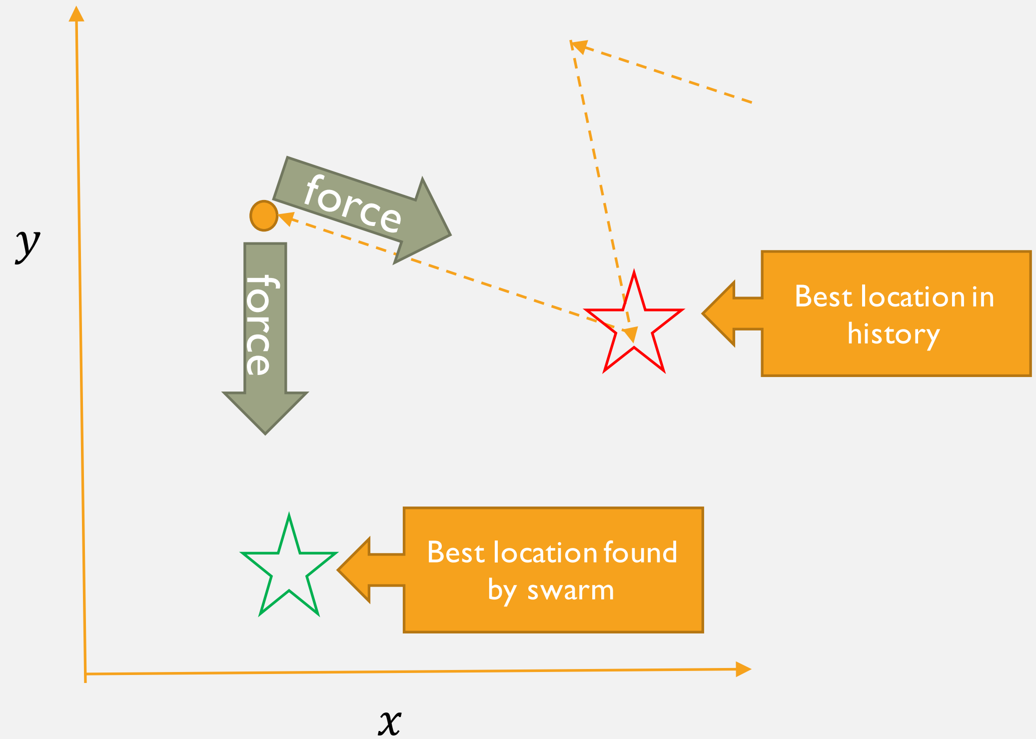




VELOCITY UPDATE

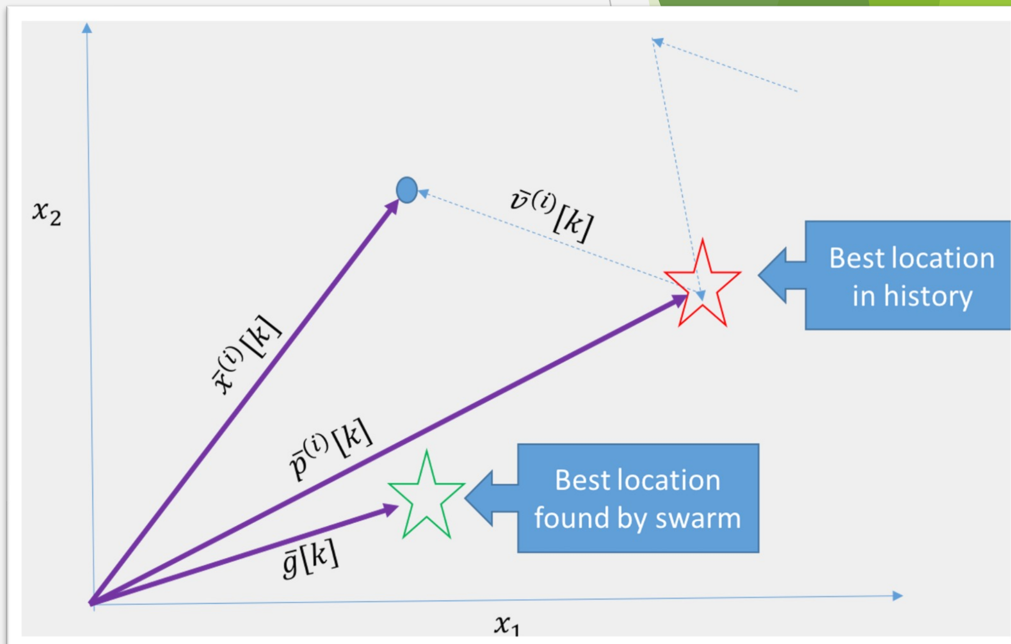
A particle explores the search space randomly but constantly feels an attractive force towards:

1. Personal best: best location it has found so far and …
2. Global best: the best location found by the swarm so far



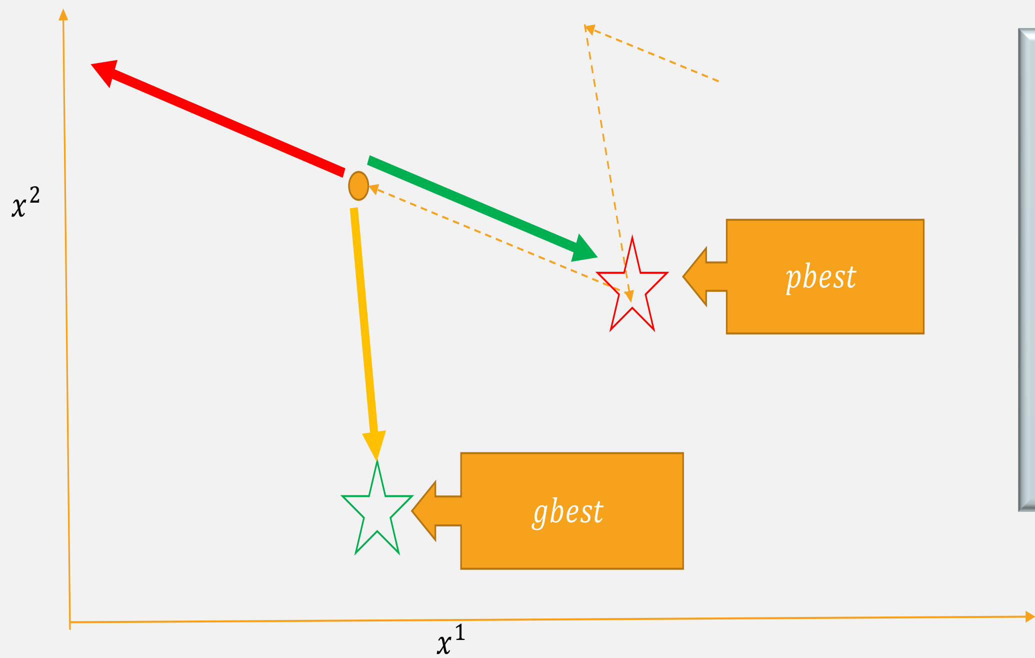
PSO terminology

Term	Definition
Particles	Locations in D-dimensional search space
$\bar{x}^{(i)}[k]$	• Position of i^{th} particle in k^{th} iteration • $\bar{x}^{(i)}[k] = (x_0^{(i)}[k], x_1^{(i)}[k], \dots, x_{D-1}^{(i)}[k])$
$\bar{v}^{(i)}[k]$	• Velocity of i^{th} particle in k^{th} iteration • $\bar{v}^{(i)}[k] = (v_0^{(i)}[k], v_1^{(i)}[k], \dots, v_{D-1}^{(i)}[k])$
$pbest$ $(\bar{p}^{(i)}[k])$	Personal best: Best location found by the i^{th} particle in its history over iterations 1 through k
$gbest$ $(\bar{g}[k])$	Global best: Best location found by any particle over iterations 1 through k (i.e., best location found by the swarm in its history)
v_{max}	Maximum velocity "Velocity Clamping": $v_j^{(i)}[k] \in [-v_{max}, v_{max}]$



VELOCITY UPDATE

$$v_j^{(i)}[k+1] = w\, v_j^{(i)}[k] + c_1 r_{1,j}(p_j^{(i)}[k] - x_j^{(i)}[k]) + c_2 r_{2,j}(g_j\,[k] - x_j^{(i)}[k])$$



$r_{m,j}$: random variable with uniform PDF in $[0,1]$

c_1, c_2 :"**acceleration constants**"

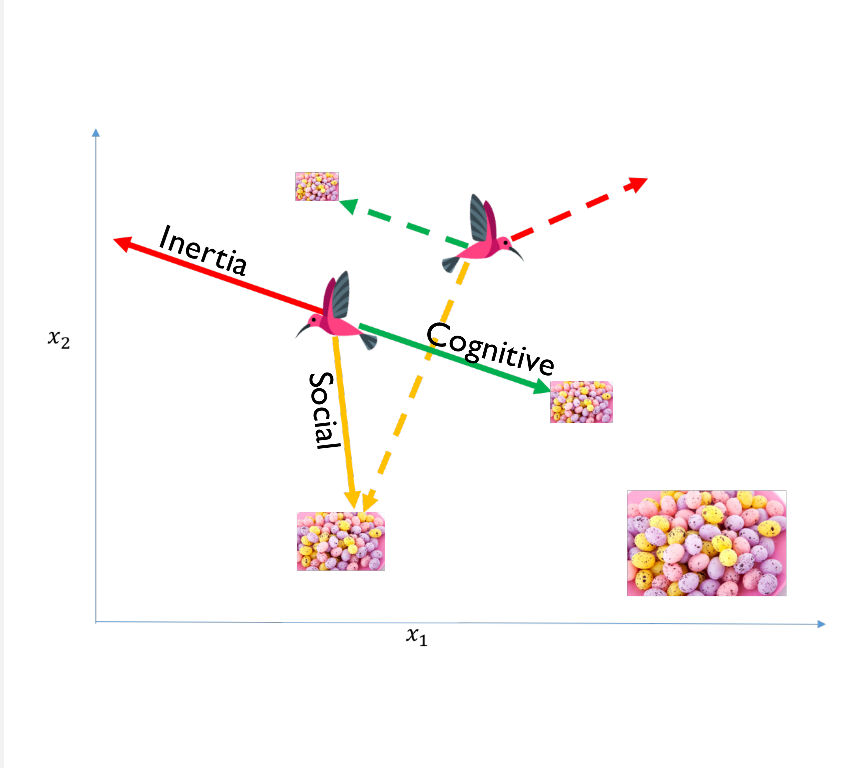
w : "inertia" $\rightarrow w\, v_j^{(i)}[k]$:"**Inertia Term**"

$c_1 r_{1,j}(p_i^j[k] - x_i^j[k])$:"**Cognitive term**"

$c_2 r_{2,j}(g[k] - x_i^j[k])$:"**Social term**"

INTERPRETATION

- Inertia term: promotes exploration
  - On its own, this term will keep the particle moving without caring about its fitness value → exploration
  - $w < 1$ to avoid "particle explosion": Particles can leave the search space and never return
  - Common choice: Linear decay of w
- Social and cognitive terms: promote exploitation (caring about improving fitness)
  - Randomization in these terms promotes exploration



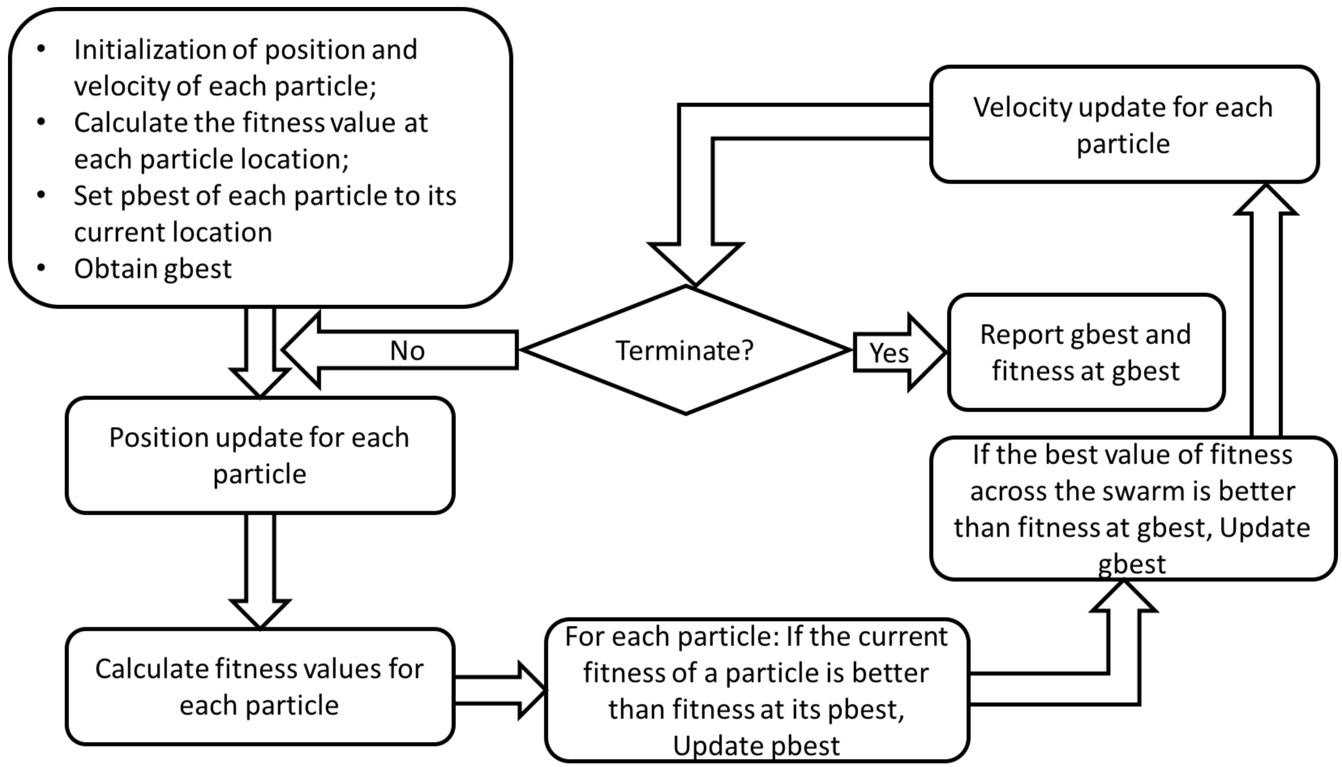
PSO DYNAMICAL EQUATIONS

Velocity update

$$v_j^{(i)}[k+1] = w\, v_j^{(i)}[k] + c_1 r_{1,j}(p_j^{(i)}[k] - x_j^{(i)}[k]) + c_2 r_{2,j}(g_j\,[k] - x_j^{(i)}[k])$$

Position update

$$x_j^{(i)}[k+1] = x_j^{(i)}[k] + v_j^{(i)}[k+1]$$

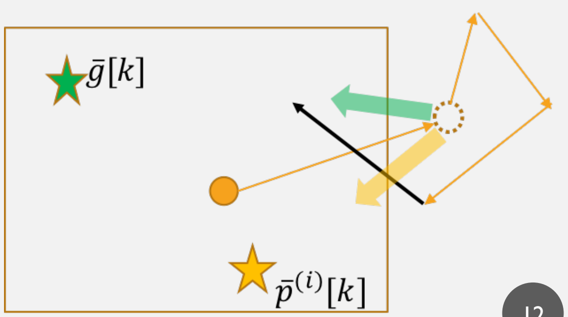

Initialization of position and velocity of each particle;
Calculate the fitness value at each particle location;
Set pbest of each particle to its current location
Obtain gbest
Velocity update for each particle
No
Terminate?
Yes
Report gbest and fitness at gbest
Position update for each particle
If the best value of fitness across the swarm is better than fitness at gbest, Update gbest
Calculate fitness values for each particle
For each particle: If the current fitness of a particle is better than fitness at its pbest, Update pbest

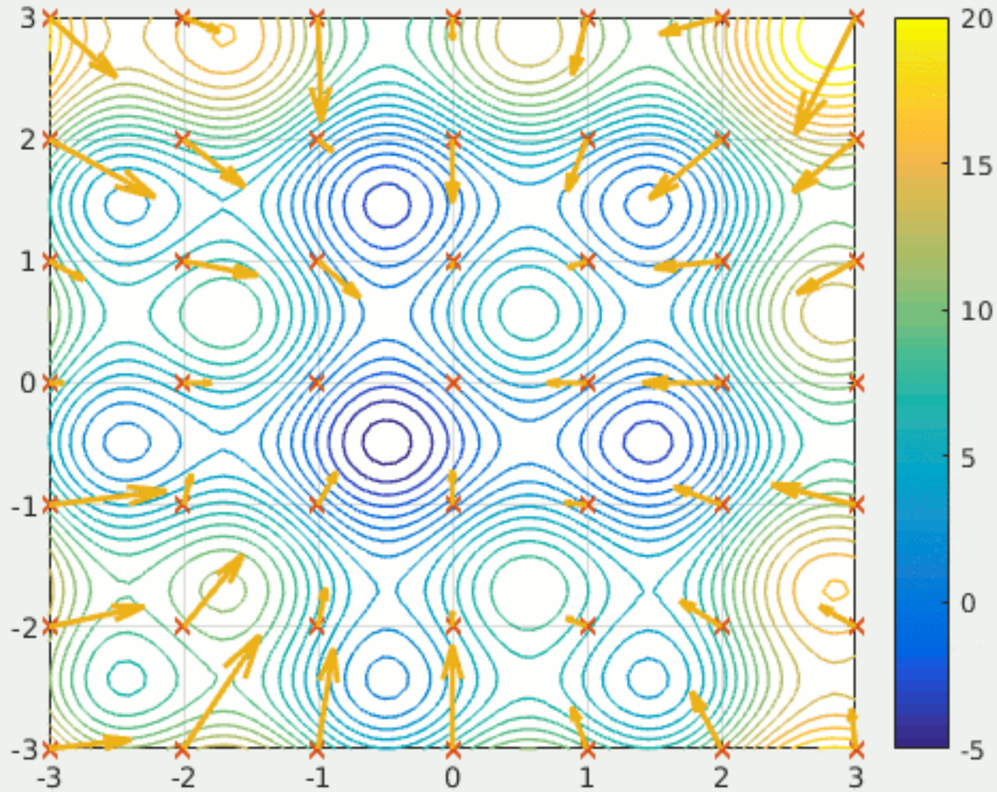
INITIALIZATION AND TERMINATION

Initialization	Initial velocity	Termination condition	Boundary condition ("let them fly")

• $x_j^{(i)}[0]$ is picked from a uniform distribution $U(0,1)$
• Search space assumed to be a hypercube
	• Boundary constrained:
	  • $v_j^{(i)}[0] \sim U(-x_j^{(i)}[0], 1 - x_j^{(i)}[0])$
	  • $\Rightarrow v_j^{(i)}[0] + x_j^{(i)}[0] \in [0,1]$ in the next iteration
	  • & Velocity clamping
		• Number of iterations	• Set fitness to $+\infty$ outside the boundary and continue to iterate the dynamical equations
			• $pbest$ and $gbest$ eventually pull the particle back

Let them fly





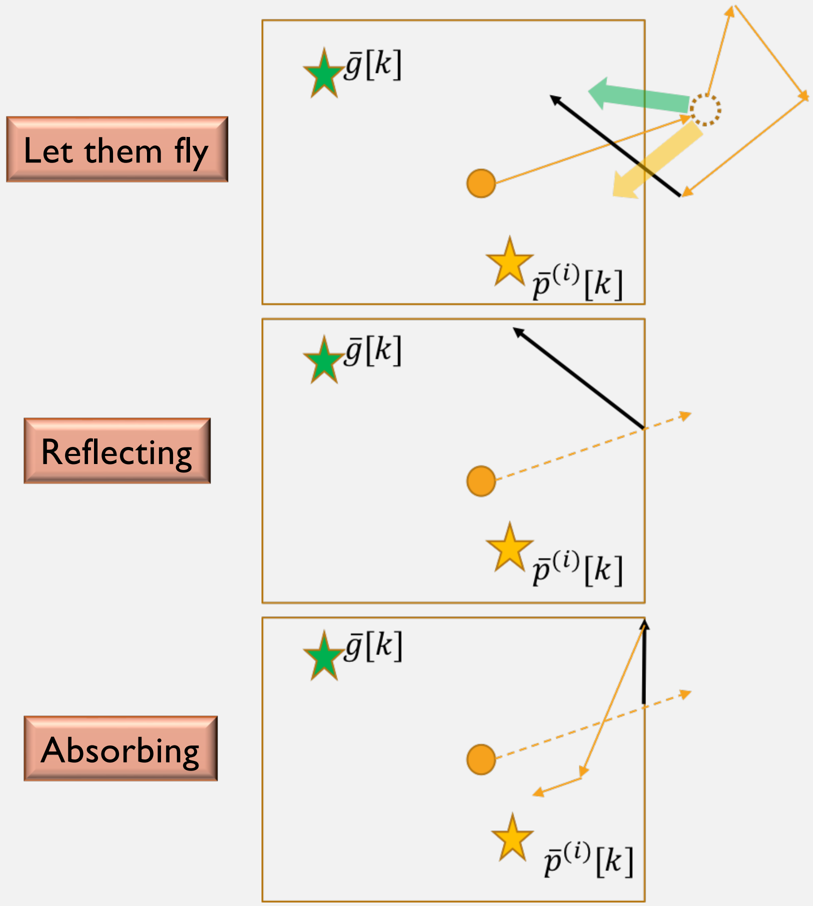
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
20
15
10
5
0
-5

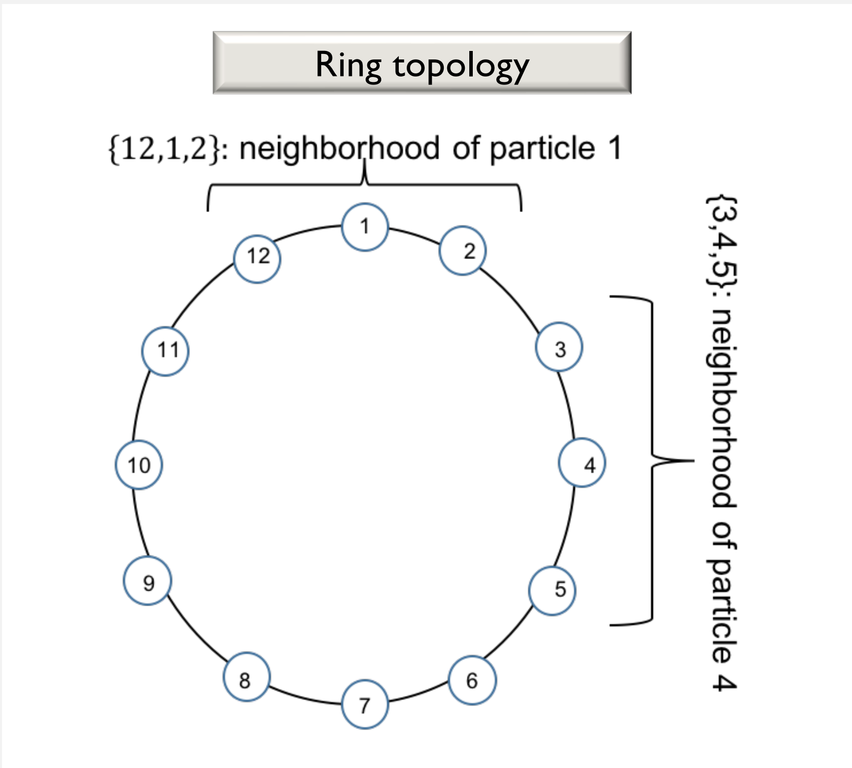
PSO VARIANTS

See textbook for more discussion

BOUNDARY CONDITIONS

- "Let them fly"
- "Reflecting walls" : Change the sign of the velocity component perpendicular to the boundary surface
- "Absorbing Walls": zero the velocity component perpendicular to the boundary surface





COMMUNICATION TOPOLOGY

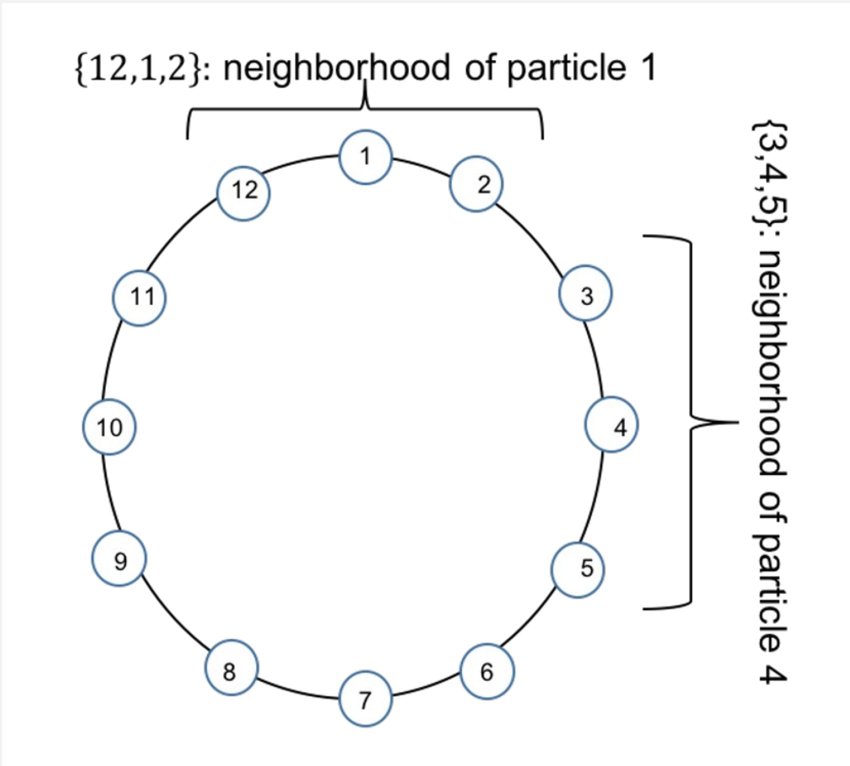
$$v_j^{(i)}[k+1] = w[k]v_j^{(i)}[k] + c_1 r_{1,j}\left(p_j^{(i)}[k] - x_j^{(i)}[k]\right) + c_2 r_{2,j}\left(g_j[k] - x_j^{(i)}[k]\right)$$

<u>Local best PSO</u>

$\bar{g}[k] \rightarrow \bar{l}^{(i)}[k]$: best location in a neighborhood of the i^{th} particle

$$lbest: \bar{l}^{(i)}[k]$$

How neighborhoods are defined sets up the topology of the swarm



$lbest$ PSO

<u>Local best PSO</u>

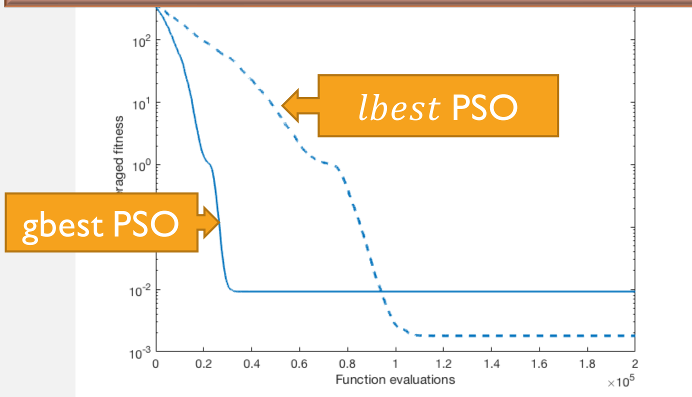
$$\bar{g}[k] \rightarrow \bar{l}^{(i)}[k]$$

- Information about global best (e.g., particle #5) shared through common particles

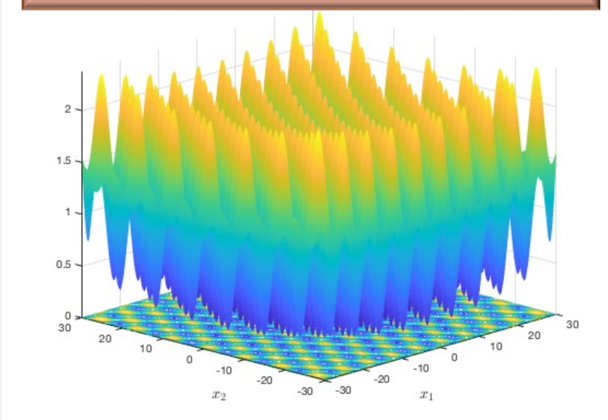
  $\ldots, (1, 2, 3), (2, 3, 4), (3, 4, 5), \ldots$

- Information about global best propagates more slowly through the swarm
- Less social attraction: extended exploration

30D Generalized Griewank; $x_i \in [-600,600]$



2D Generalized Griewank



$lbest$ PSO PERFORMANCE

- $lbest$ PSO is computationally more expensive than $gbest$ PSO but may be more successful in locating a better solution

"BEST OF M RUNS" STRATEGY

probability of "success" in one run: p

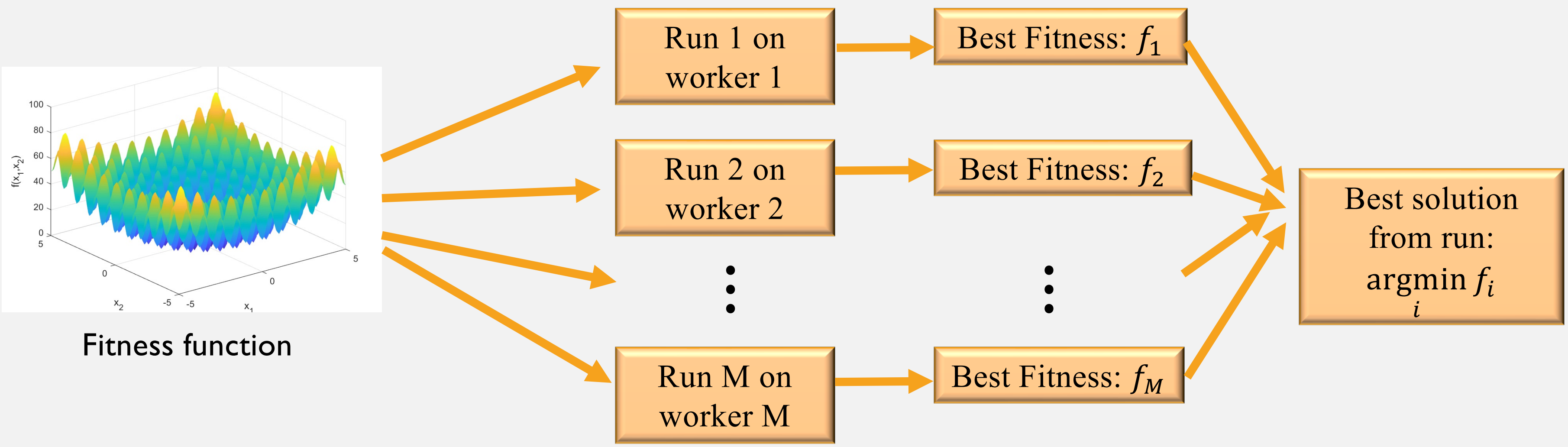
Probability of failure over M runs: $(1-p)^M$

- Example: If $p = 0.5$, failure probability over $M = 10$ runs is $\simeq 0.001$

Tuning strategy: Target a moderately high p and pick best fitness from M runs

- Moderate tuning reduces the danger of over-tuning
- Reduces the effort needed to achieve good tuning

PRALLELIZATION IN BMR STRATEGY



Homework

PSO

- There are 2 parts to this homework
- Part 1: Guides you in using PSO on a benchmark fitness function
- Part 2: Shows how to set up GLRT for a toy problem and implement it using PSO

Part 1

Benchmark fitness function

PSO codes

Clone the repository: **GitHub →SDMBIGDAT19** (Store it outside GWSC)

Lectures delivered at the BigDat19 5th International Winter school on Big Data, Cambridge University, UK (Jan, 2019)

The codes and slides supplement the textbook; Cover PSO and stochastic optimization in greater depth (including fundamental theorems)

Exercise

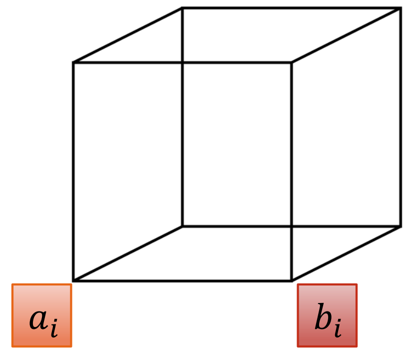
- To use the PSO code, understand the concept of structures in Matlab
    - Matlab structures work in the same way as structures in C
    - X = struct('a', 5.0, 'b', 6.0);
    - disp(X.a) will show 5.0
    - disp(X.b) will show 6.0
- Used the 'doc struct' command in Matlab to read more about structures
    - 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

- To use the PSO code, you also need to understand the concept of standardized coordinates
- Read the following slides, which explain standardized coordinates in the context of intrinsic signal parameters, $\Theta = (\theta_0, \theta_1, \dots, \theta_{M-1})$

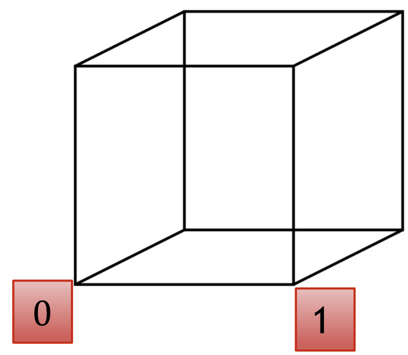
Standardized coordinates

Consider a **Hypercubical** search space: each parameter θ_i can be varied independently in some interval $\theta_i \in [a_i, b_i]$



Standardized coordinates:

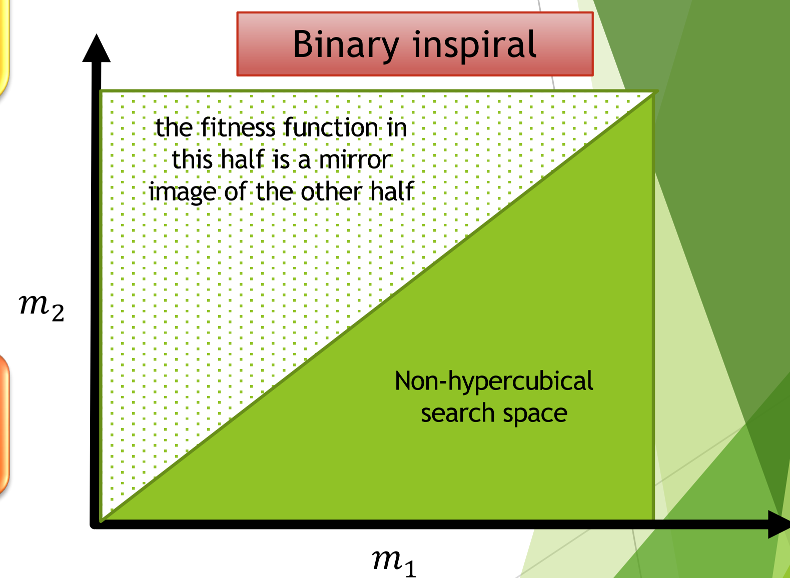
- $\theta_i \to x_i = \frac{\theta_i - a_i}{b_i - a_i}$
- $0 \le x_i \le 1$ for $\theta_i \in [a_i, b_i]$
- Any hypercubical search space can be assumed to have parameters that range between 0 and 1



Standardized coordinates

Consider a **Hypercubical** search space: each parameter θ_i can be varied independently in some interval $\theta_i \in [a_i, b_i]$

- **Caution:** Parameters may not be independent in some problems!
- Example: Binary inspiral parameter space with component masses m_1 and m_2
- The signal waveform has a symmetric



Standardized coordinates:

- $\theta_i \to x_i = \frac{\theta_i - a_i}{b_i - a_i}$
- $0 \le x_i \le 1$ for $\theta_i \in [a_i, b_i]$
- Any hypercubical search space can be assumed to have parameters that range between 0 and 1

- We will need the following codes in **SDMBIGDAT19/CODES**: Make sure they exist in the repository you have cloned

- **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

Exercise

- Read the short user manual **CODES/CodeDoc.pdf**
- The **test_crcbpso.m** script is an example that shows how to call the main function, **crcbpso** (defined in the crcbpso.m file), and apply it to a fitness function (defined in **crcbpsotestfunc.m**)
- Use Matlab's 'help' command to read the help for the **crcbpso** and **crcbpsotestfunc** functions and understand how to use these functions
- Make 2D plots of the benchmark fitness function
- Run and experiment with the user-defined parameters in test_crcbpso
- Change the dimensionality of the search space and see the effect on the performance of PSO in locating the global minimum of the benchmark fitness function

Part 2

PSO-based implementation of GLRT

PSO-based GLRT

- A data analysis problem is defined here that mimics the search for binary inspiral signals
  - The signal is a quadratic chirp with increasing frequency (but not increasing amplitude)
  - The noise is White Gaussian
- To simplify the code, the signal does not have the time-of-arrival parameter
- However, the signal has three intrinsic parameters, which makes a grid-based search computationally expensive and make PSO a natural alternative to implement the GLRT and MLE
- First, you will implement the GLRT fitness function for the signal and noise models defined above
- Next, for a given data realization, you will use PSO to obtain MLE signal parameter estimates

SIGNAL MODEL

Quadratic chirp

$$s(t;\Theta) = A\sin(2\pi\Phi(t) + \phi_0)$$

Instantaneous phase:

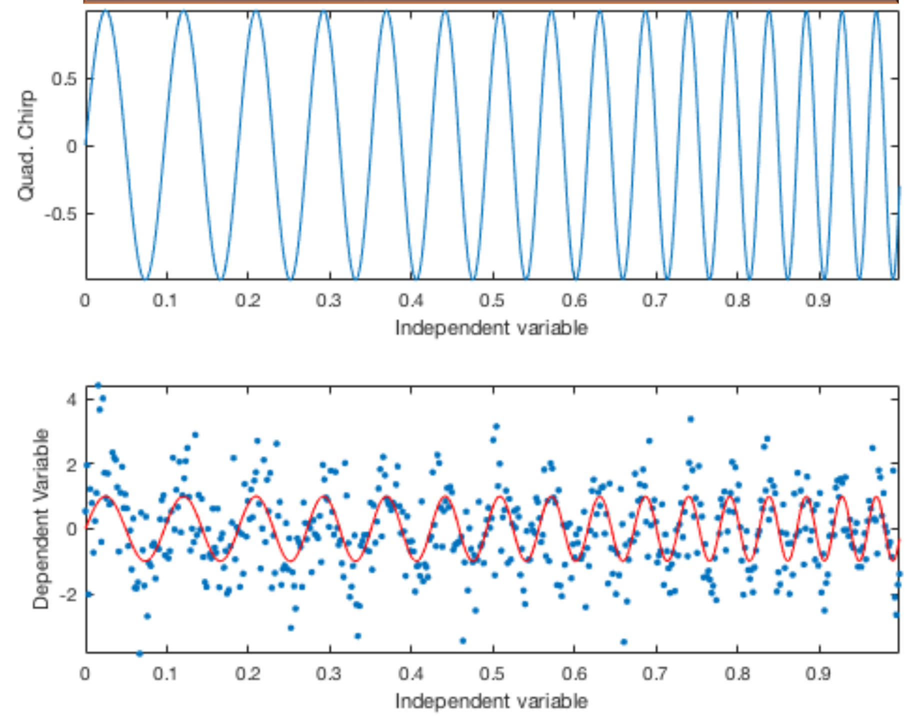
$$\Phi(t) = a_1 t + a_2 t^2 + a_3 t^3$$

Instantaneous frequency:

$$f(t) = \frac{d\Phi}{dt} = a_1 + 2a_2 t + 3a_3 t^2$$

- Extrinsic: A, ϕ_0
- Intrinsic: a_1, a_2, a_3

$f(t)$ increases with t



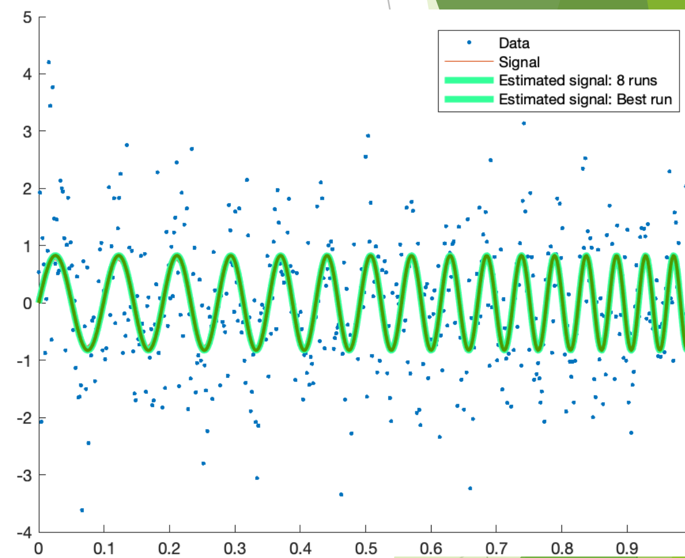
Data realization

Codes

- **crcbgenqcsig.m:** Generates a quadratic chirp signal
- **crcbqcfitfunc.m:** The fitness function for quadratic chirp in WGN
- **crcbqcpso.m**: Applies PSO to the optimization of the fitness function
- **test_crcbqcpso.m:** Test function to run crcbqcpso on a simulated data realization

Exercise

- Review the derivation of the fitness function for the case where the extrinsic parameters are amplitude and initial phase
- Read through **crcbqcfitfunc.m** to see how the fitness function is implemented
  - Note: it is implemented in terms of standardized coordinates
- Read through crcbqcpso.m to see how PSO is used and how the extrinsic parameters are estimated following the estimation of the intrinsic parameters
  - Note: Both extrinsic and intrinsic parameters are needed in order to plot the signal waveform
- Run **test_crcbqcpso.m**
- This script generates a data realization and applies MLE to estimate the signal parameters
  - It also plots the estimated waveform and compares it to the true waveform
- Experiment with the signal parameters; study the dependence of parameter estimation errors on the SNR of the signal; Use multiple data realizations to obtain parameter estimation errors (Challenge: compare with CRLB)



Challenge

- Generalize the codes for the quadratic chirp MLE and GLRT from the White Gaussian noise case to Colored Gaussian noise
- Generate data realizations with the initial LIGO PSD, add the quadratic chirp, and test the performance of PSO
  - Hint: example codes are provided in GWSC / DETEST but you will learn more by first writing your own codes