Florida International University
Optimization in Water Resources Engineering
Spring 2020

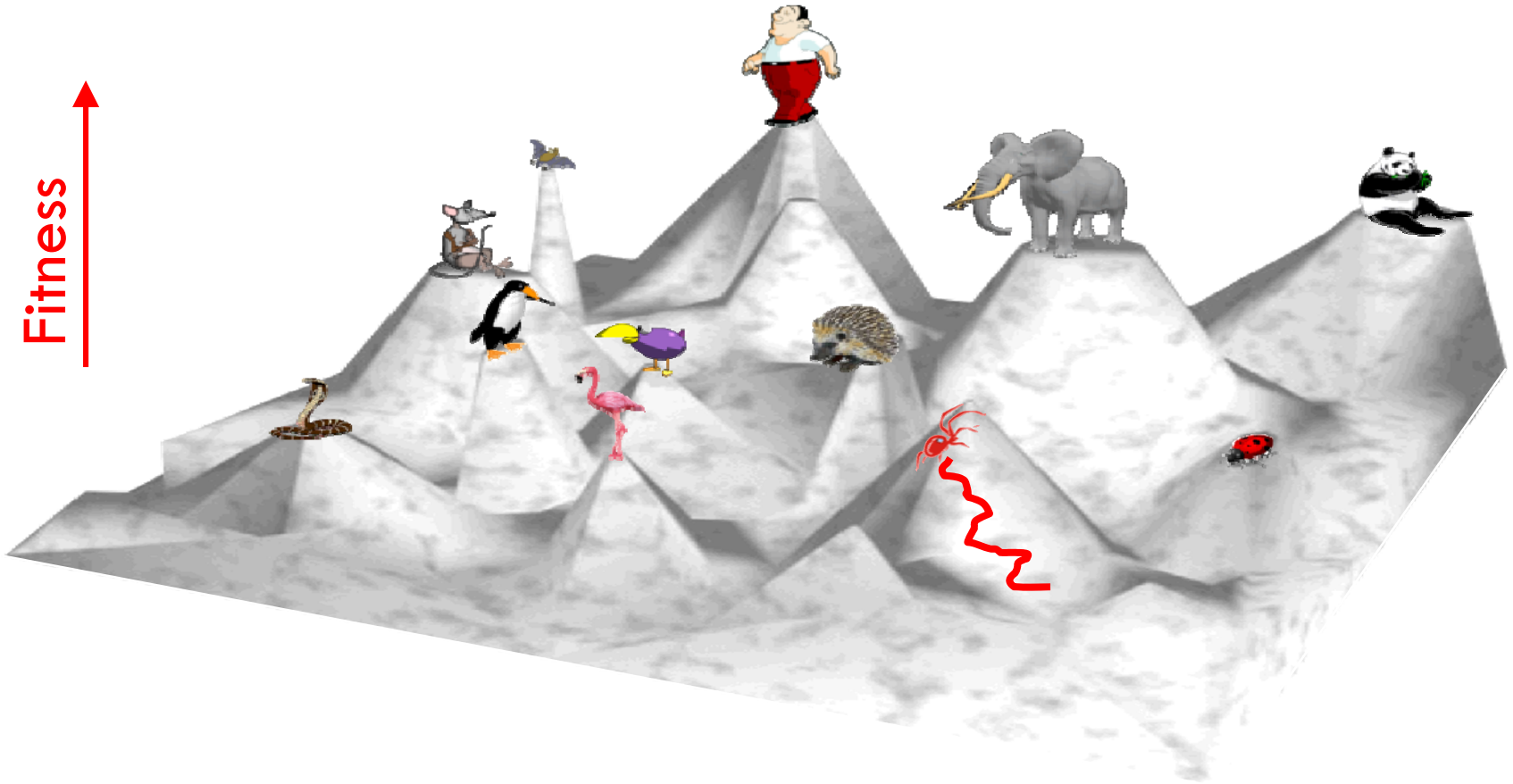
Genetic Algorithms

Arturo S. Leon, Ph.D., P.E., D.WRE

Part of the material presented herein was adapted from:
A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing; Aleksandra Popovic et al.; Wendy Williams (Metaheuristic Algorithms); S. J. Van Vuuren, University of Pretoria; Mostafa Ranjbar (Yildirim Beyazit University)

What is GA?

https://www.youtube.com/watch?v=XcinBPhgT7M



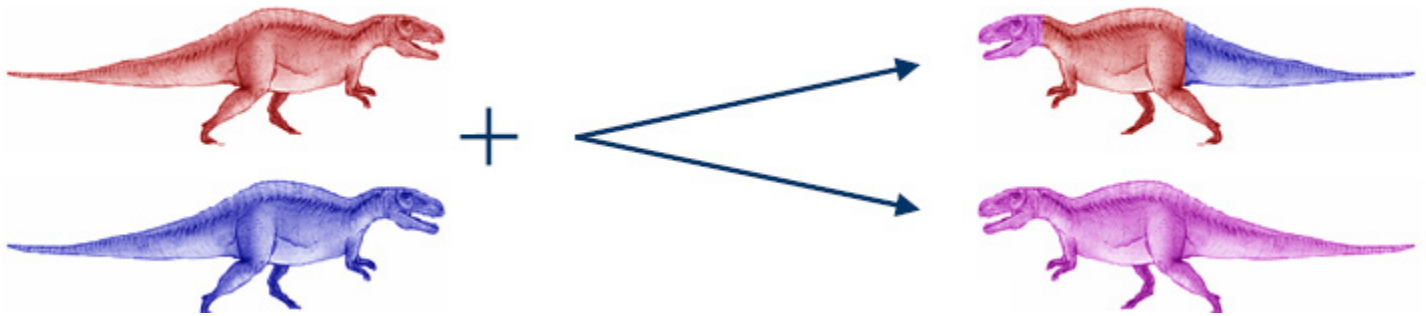
Herein, evolution means

climbing a fitness-hill

Evolution in Biology

- Organisms produce a number of offspring similar to themselves but can have variations due to:
  - **Mutations** (random changes)

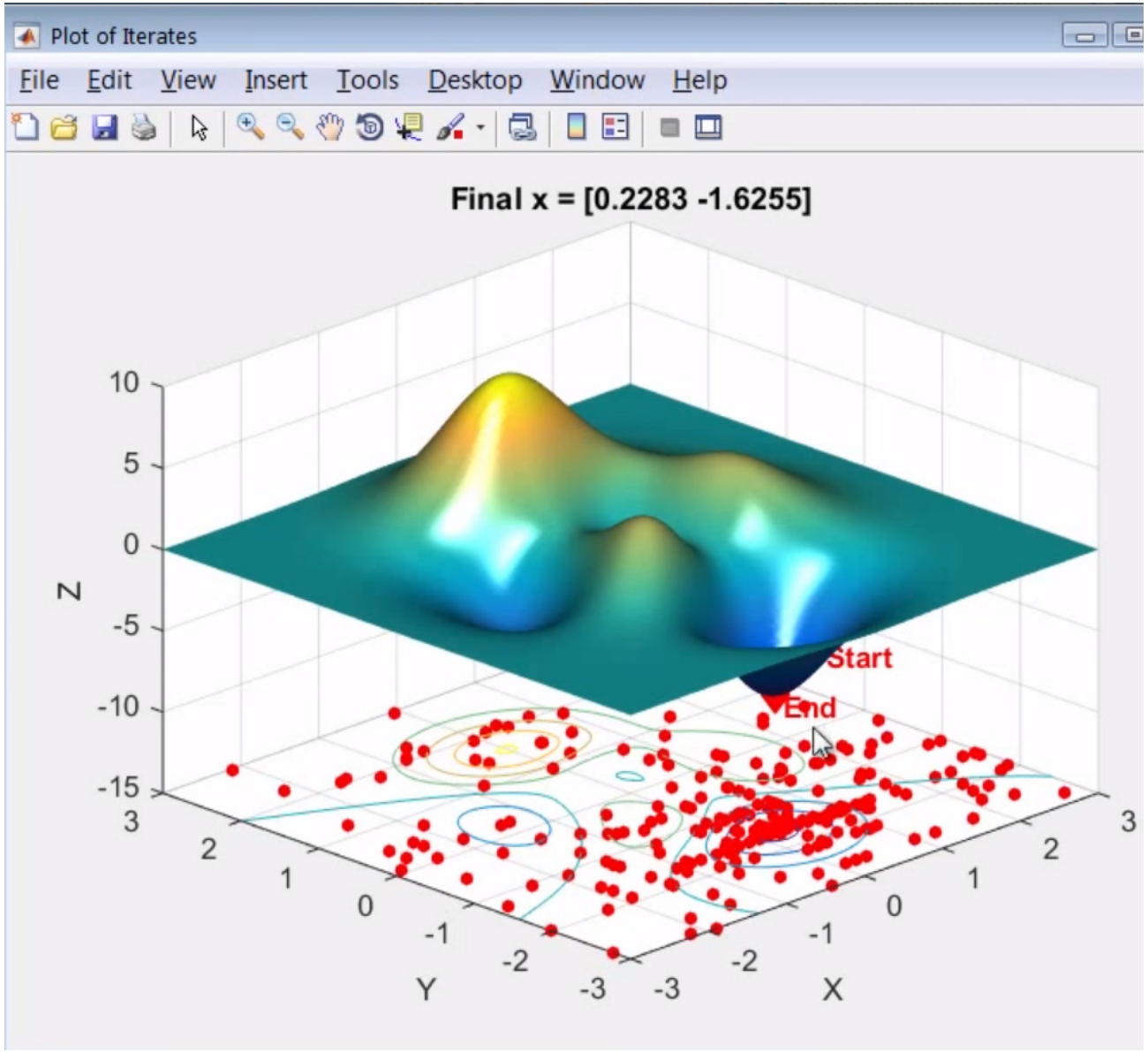
  - **Sexual reproduction** (offspring have combinations of features inherited from each parent)



Evolution in Biology (Cont.)

- Some offspring survive, and produce next generations, and some don't:
  - The organisms **adapted to the environment** better have higher chance to **survive**
  - Over time, the generations become more and more adapted because the **fittest** organisms **survive**

GA Brief Introduction



https://www.youtube.com/watch?v=1i8muvzZkPw

Nature Vs Computer - Mapping

Nature	Computer
Population	Set of solutions.
Individual	Solution to a problem.
Fitness	Quality of a solution.
Chromosome	Encoding for a Solution.
Gene	Part of the encoding of a solution.
Reproduction	Crossover

GA: Population operators

SELECTION:

- Identify the good solutions in a population
- Make multiple copies of the good solutions
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population

CROSSOVER:

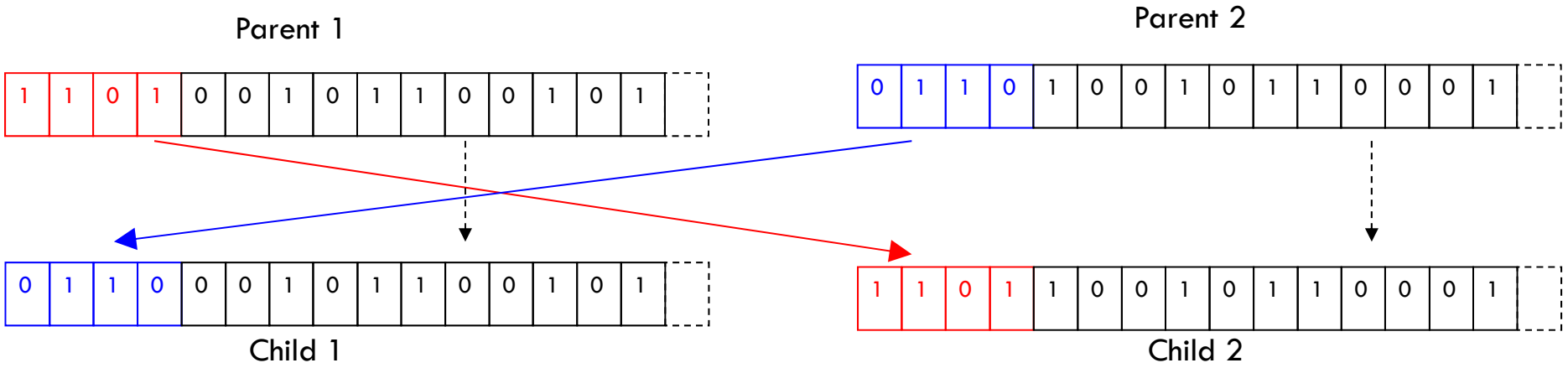
- Randomly exchange genes of different parents
- Many possibilities: how many genes, parents, children …

MUTATION:

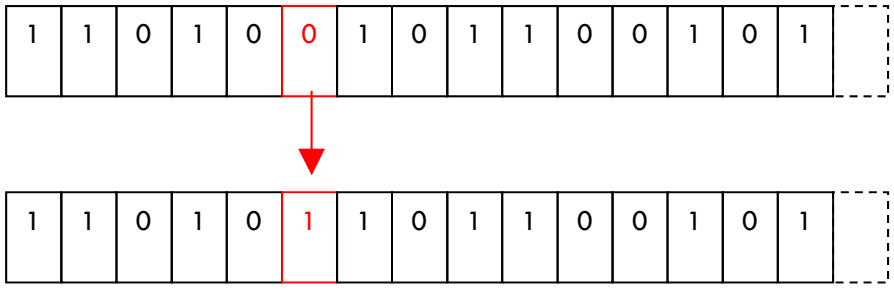
- Randomly flip some bits of a gene string
- Used sparingly, but important to explore new designs

GA: Population operators

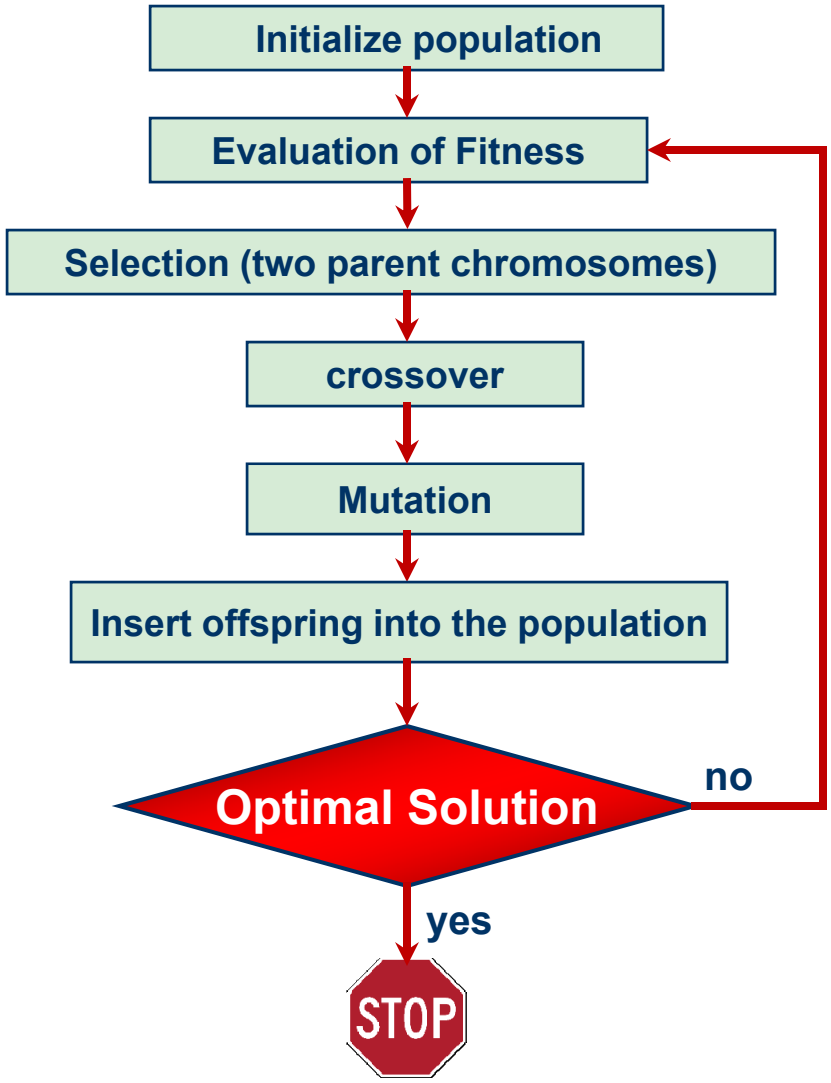
- **CROSSOVER:**



- **MUTATION:**



GA Flow Chart



Characterizing a GA Via an Example

- *maximize* $F(x) = x^2$

  s.t. $x \in [0, 31]$

Binary String Representation of an Integer Number

- Any integer number can be written in **decimal system**

  x = 2,765 can be written as

  $2{,}765 = 2\times10^3 + 7\times10^2 + 6\times10^1 + 5\times10^0$

- It is also possible to code a number in **binary form**

  $x = 39 = 1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

  or simply x can be represented by a string of 6 bits as

  $x = (100111)$

Binary String Representation of an Integer Number

- Estimation on the length of the binary string for an integer number

  an integer variable $x \in [a, b]$

  Length of binary string > $\log_2(b-a)$

- For example, $x \in [0, 31]$

len> $\log_2(31-0) = \log_2 31 = 4.96$

$31 = 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$31 = (11111)$

Binary String Representation of a Continuous Function

- For functional optimization

  Maximize $F(x)$ where $x \in [a, b]$

- Generate a bit string of length k, say 22.

  For instance, this gives $x' = (01011 \ldots 0110)$, hence

  $$x' \in [0, 2^{22} - 1]$$

- Translate x' into $x \in [a, b]$

  $$\frac{b-a}{2^{22}-1}$$

- precision or accuracy=

  $$x = a + x' \times \frac{b - a}{2^{22} - 1}$$

Binary String Representation of a Continuous Function

- Accuracy estimation

  A continuous variable $x \in [a, b]$

  Length of binary string = m

  Accuracy = $(b\text{-}a)/(2^m\text{-}1)$

- Estimation on the length of the binary string

$$length > \log_2 \frac{b - a}{\text{accuracy required}}$$

- For example, $x \in [4.1, 6.8]$, accuracy required=10^{-4}

  len> $\log_2[(6.8\text{-}4.1)/10^{-4}] = \log_2 17000 = 14.1$

Binary String Representation of a Continuous Function

- In the example of the continuous function optimization

  maximize $F(x) = x^2$ where $x \in [0, 31]$

  we use a binary coding, set accuracy=1, then the string length is

$$\text{length} > \log_2 \frac{b-a}{\text{accuracy}} = \log_2 \frac{31}{1} = \log_2 31 = 4.96$$

Use 5

maximum $x = 31$ can be represented by

$31 = (11111)$

x^2 example: selection

$0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$

Rounded to two decimals

String no.	Initial population	x Value	Fitness $f(x)=x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

X^2 example: crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

X^2 example: mutation

String no.	Offspring after crossover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

: Keep iterating

optimal: $x^2 = 961$

Advantages and disadvantages

Advantages:

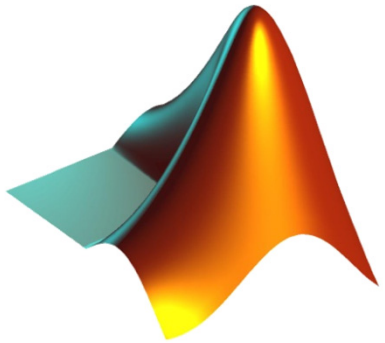
- Always an answer; answer gets better with time
- Good for "noisy" environments
- Inherently parallel; easily distributed

Issues:

- Performance
- Solution is only as good as the evaluation function
- Termination Criteria

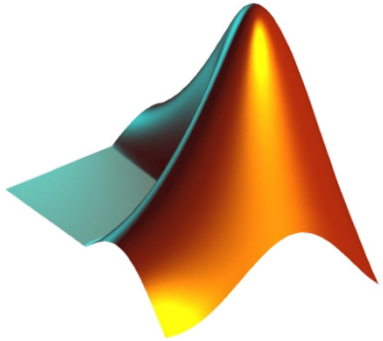
GENETIC ALGORITHM IN MATLAB

SYNTAX



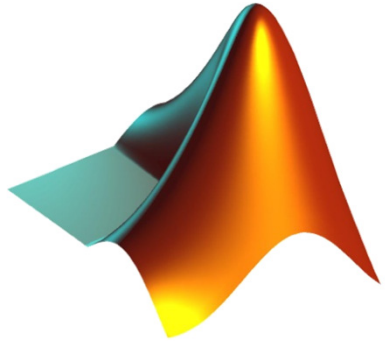
- x = ga(fitnessfcn,nvars)
- x = ga(fitnessfcn,nvars,A,b)
- x = ga(fitnessfcn,nvars,A,b,Aeq,beq)
- x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB)
- x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon)
- x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon,options)

MATLAB (cont.)



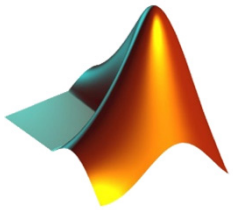
Fitnessfcn	Fitness function
Nvars	Number of design variables
Aineq	A matrix for linear inequality constraints
Bineq	B vector for linear inequality constraints
Aeq	A matrix for linear equality constraints
Beq	b vector for linear equality constraints
Lb	Lower bound on x
Ub	Upper bound on x
Nonlcon	Nonlinear constraint function
Randstate	Optional field to reset rand state
Randnstate	Optional field to reset randn state
Solver	'ga'
Options	Options structure created using gaoptimset

MATLAB (cont.)



```
nvars = 17;                                                    % NUMBER OF VARIABLES
fitnessFunction = @MY_FUNCTION;
options = gaoptimset;                    % %Start with default options      %Modify some parameters
options = gaoptimset(options,'PopInitRange' ,[lb;ub]); % options = gaoptimset(options,'InitialPopulation' ,[InitPop]);
options = gaoptimset(options,'PopulationSize' ,100);                          % POPULATION SIZE
options = gaoptimset(options,'Generations' ,100);                             % NUMBER OF GENERATIONS
options = gaoptimset(options,'StallGenLimit' ,50);                            % STALL GENERATION LIMIT
options = gaoptimset(options,'StallTimeLimit' ,200000000);                    % STALL TIME LIMIT
options = gaoptimset(options,'TolFun' ,1e-9);                                 % FUNCTION TOLERANCE
options = gaoptimset(options,'TolCon' ,1e-9);                                 % CONSTRAINT TOLERANCE
options = gaoptimset(options,'CrossoverFcn' ,@crossovertwopoint);             % CROSSOVER TYPE
options = gaoptimset(options,'CrossoverFraction' ,0.8);                       % CROSSOVER FRACTION
options = gaoptimset(options,'SelectionFcn' ,{ @selectiontournament 4  });    % SELECTION TYPE
options = gaoptimset(options,'MutationFcn' ,{ @mutationuniform 0.25641 });    % MUTATION TYPE
options = gaoptimset(options,'Display' ,'iter');                              % DISPLAY OPTIONS
 [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] = ga(fitnessFunction,nvars,options); %Run GA
```

MATLAB (cont.)



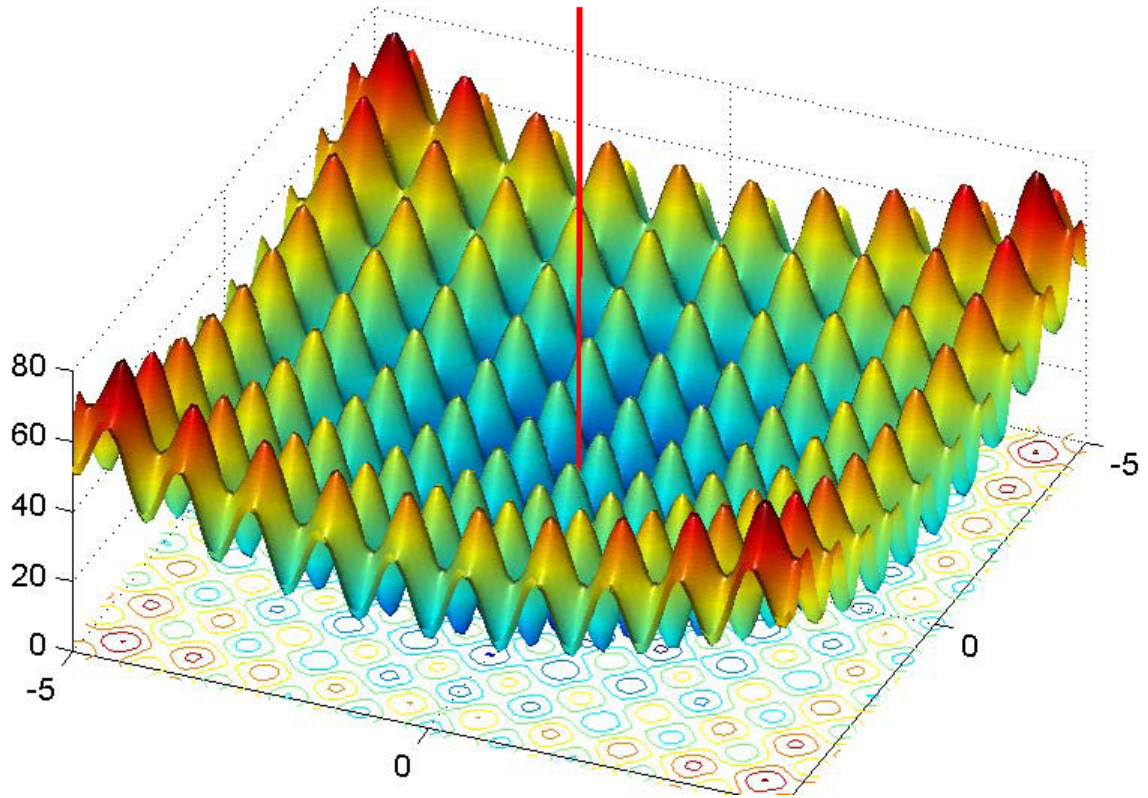
The genetic algorithm uses the following conditions to determine when to stop:

- **Generations — The algorithm stops when the number of generations** reaches the value of **Generations.**
- **Time limit — The algorithm stops after running for an amount of time in** seconds equal to **Time limit.**
- **Fitness limit — The algorithm stops when the value of the fitness function** for the best point in the current population is less than or equal to **Fitness limit.**
- **Stall generations — The algorithm stops when the weighted average** change in the fitness function value over **Stall generations is less than Function tolerance.**
- **Stall time limit — The algorithm stops if there is no improvement in** the objective function during an interval of time in seconds equal to **Stall time limit.**
- **Function Tolerance — The algorithm runs until the weighted average** change in the fitness function value over **Stall generations is less than Function tolerance.**
- **Nonlinear constraint tolerance — The Nonlinear constraint tolerance is not used as stopping criterion. It is used to determine the** feasibility with respect to nonlinear constraints

Example: Rastrigin's Function

- Find the minimum of Rastrigin's function, a function that is often used to test a genetic algorithm.
- For two independent variables, the Rastrigin's function is defined as

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$



Rastrigin's Function (cont.)

Demo folder in Canvas:
GA_Demo_Rastrigin

Main file: Main_file_Rastrigin.m

Plot file: plotfun.m

Activities: Change population size, maximum number of generations, optimization tolerance. Generate plots of solution convergence.