

# NUMERICAL METHODS-LECTURE 6: OTHER METHODS

Trevor Gallen

# MOTIVATION

- ▶ Newton's Method has flaws
- ▶ Mostly, computing Jacobian or Hessian is hard, not always possible
- ▶ Want to consider derivative-free alternatives

## LIST

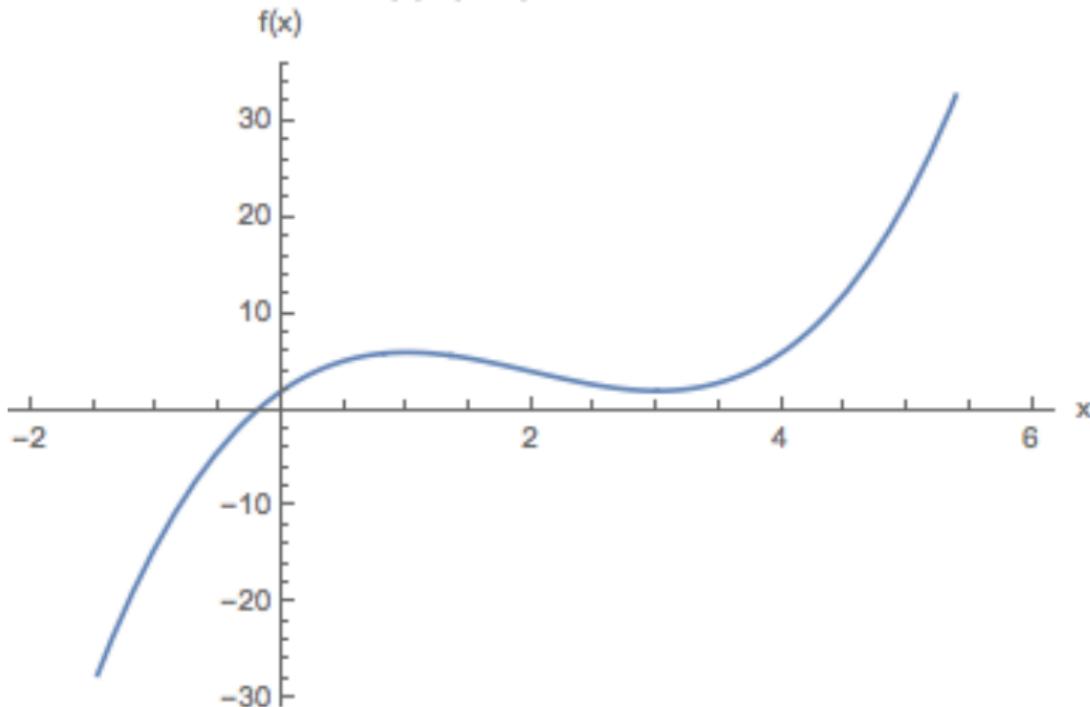
- ▶ Bisection
- ▶ Golden section search
- ▶ Nelder-Mead Simplex
- ▶ Conjugate Gradient Method
- ▶ Evolutionary, Simulated Annealing

## BISECTION METHOD

- ▶ The Bisection method is what you use naturally when looking for a zero.
- ▶ Univariate
- ▶ Want to find a root between  $a$  and  $b$
- ▶ Denote new values of  $a$  and  $b$  with  $a'$  and  $b'$ .
  1. Take two points,  $a$  and  $b$ , where  $f(a) < 0 < f(b)$ .
  2. Evaluate  $f\left(\frac{a+b}{2}\right)$  (Let  $c = \frac{a+b}{2}$ ).
  3. If  $f(c) < 0$ , then  $a' = c$ . If  $f(c) > 0$ , then  $b' = c$ .
  4. Repeat.

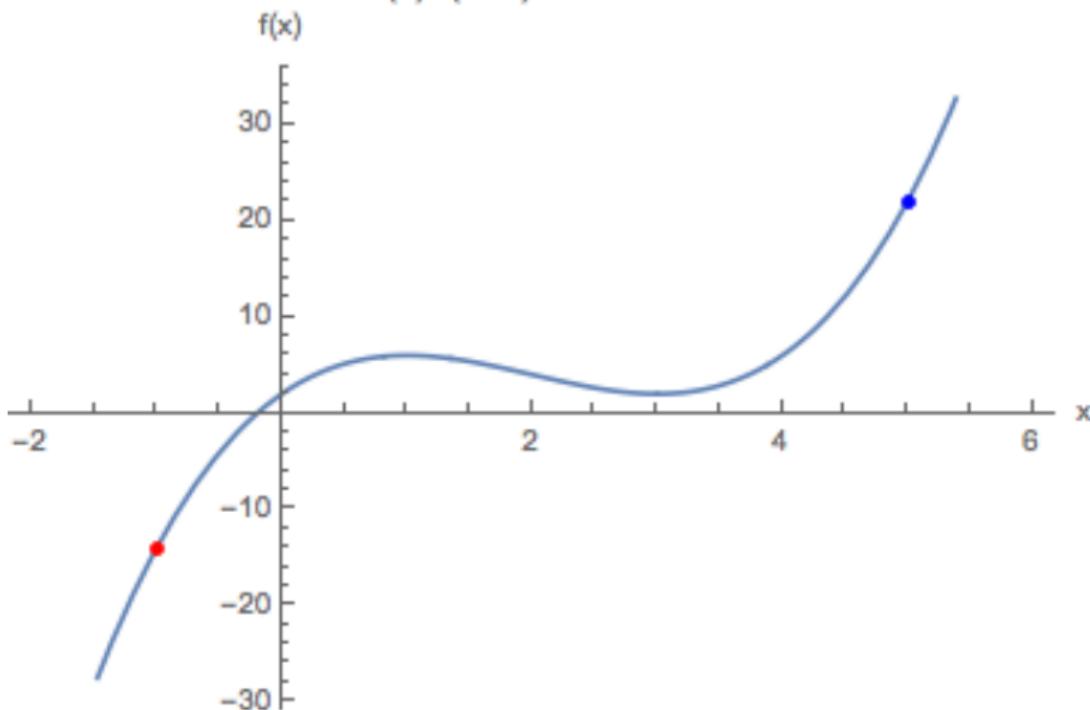
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



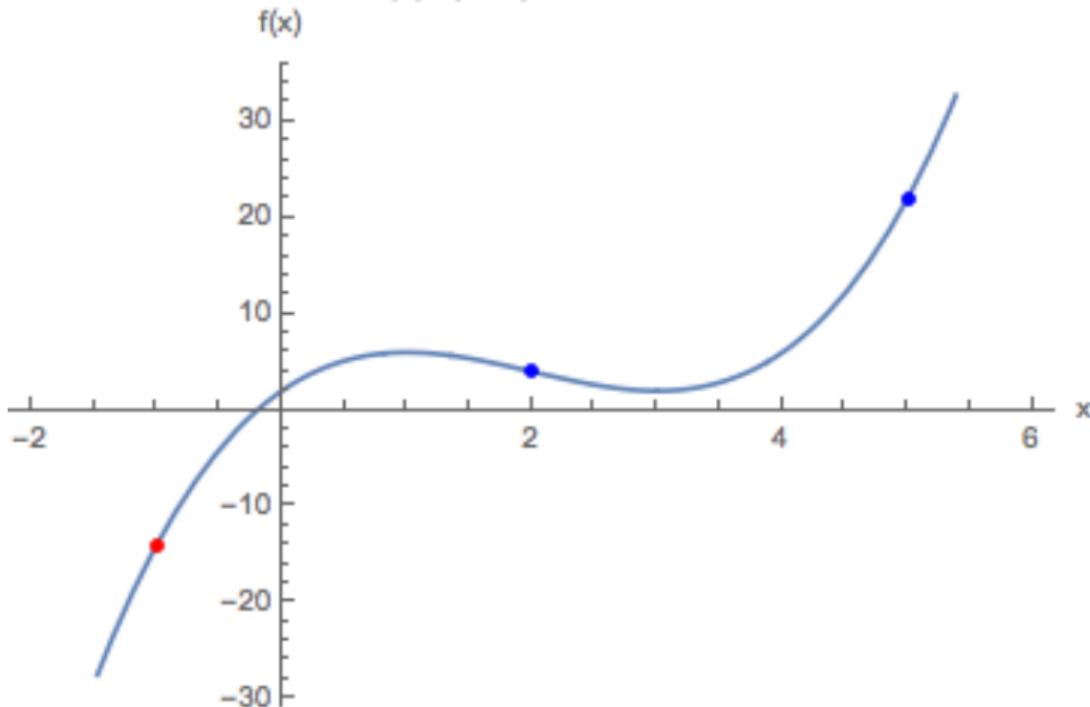
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



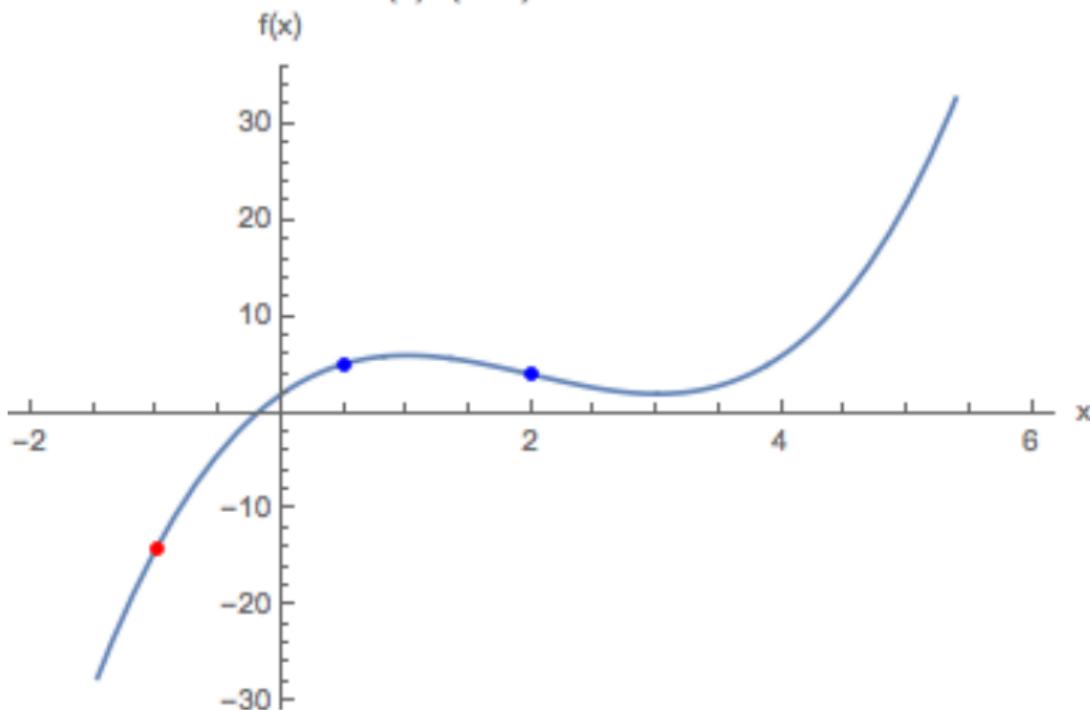
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



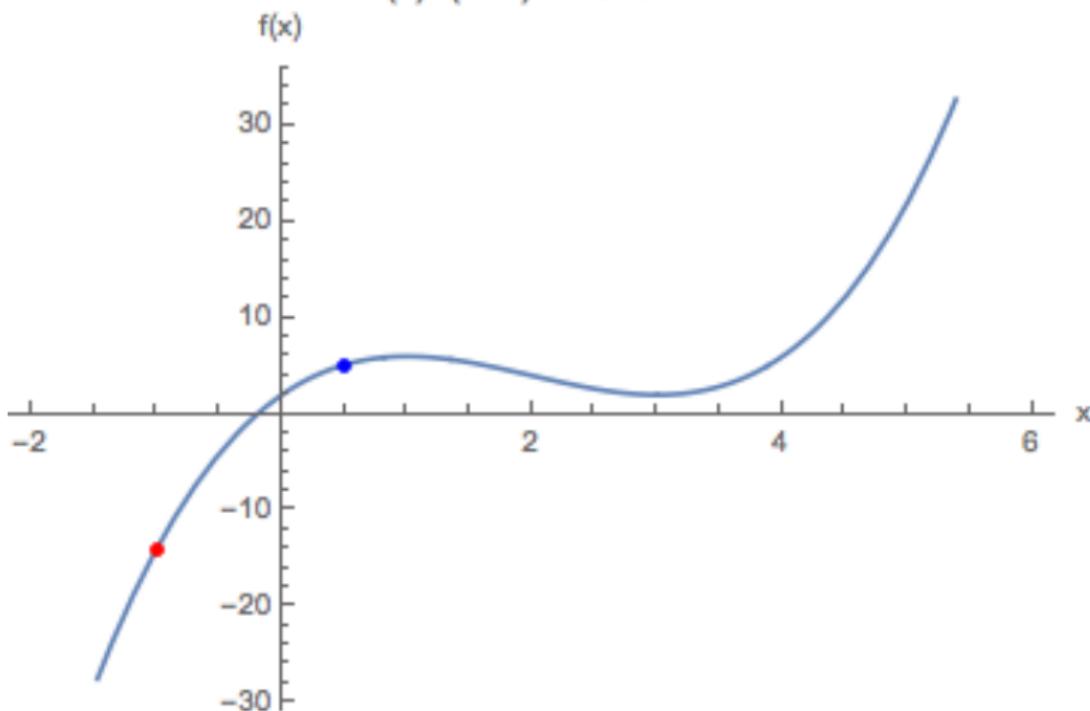
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



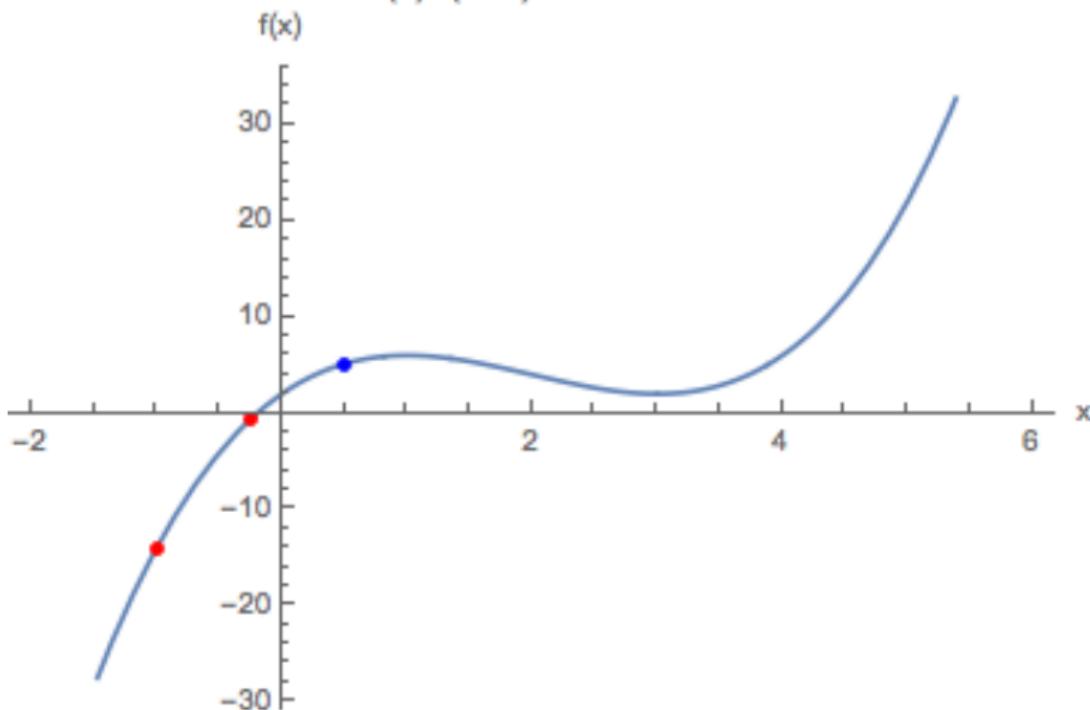
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



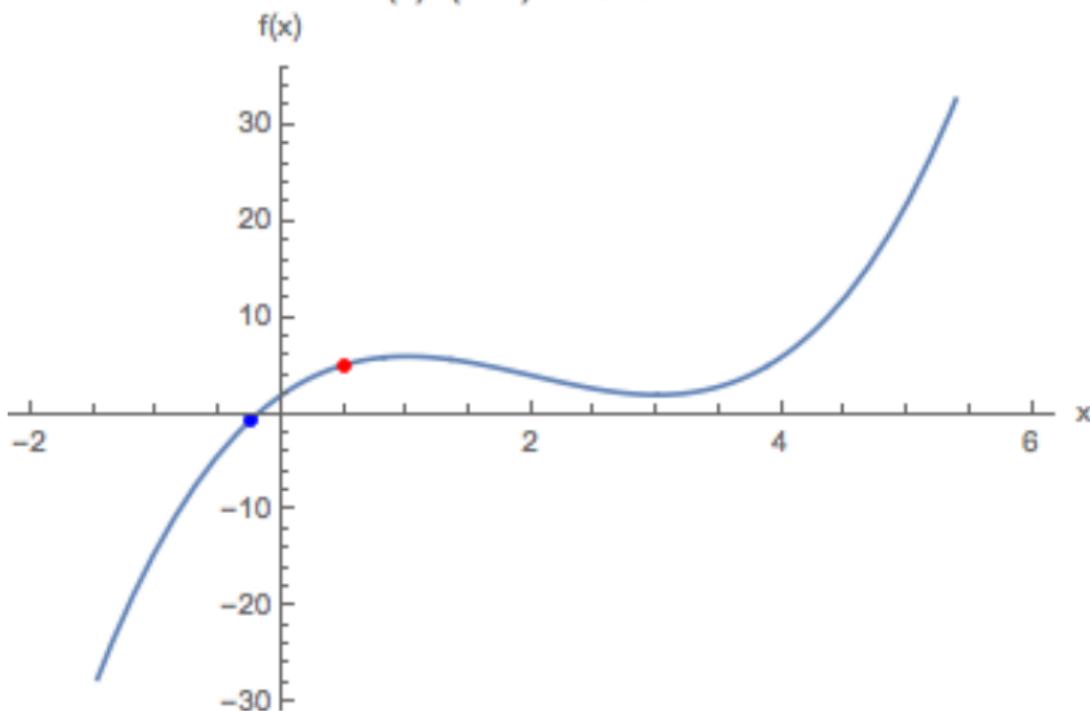
## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$



## BISECTION METHOD, GRAPHICALLY

$$f(x) = (x-2)^3 + 10 - 3x$$

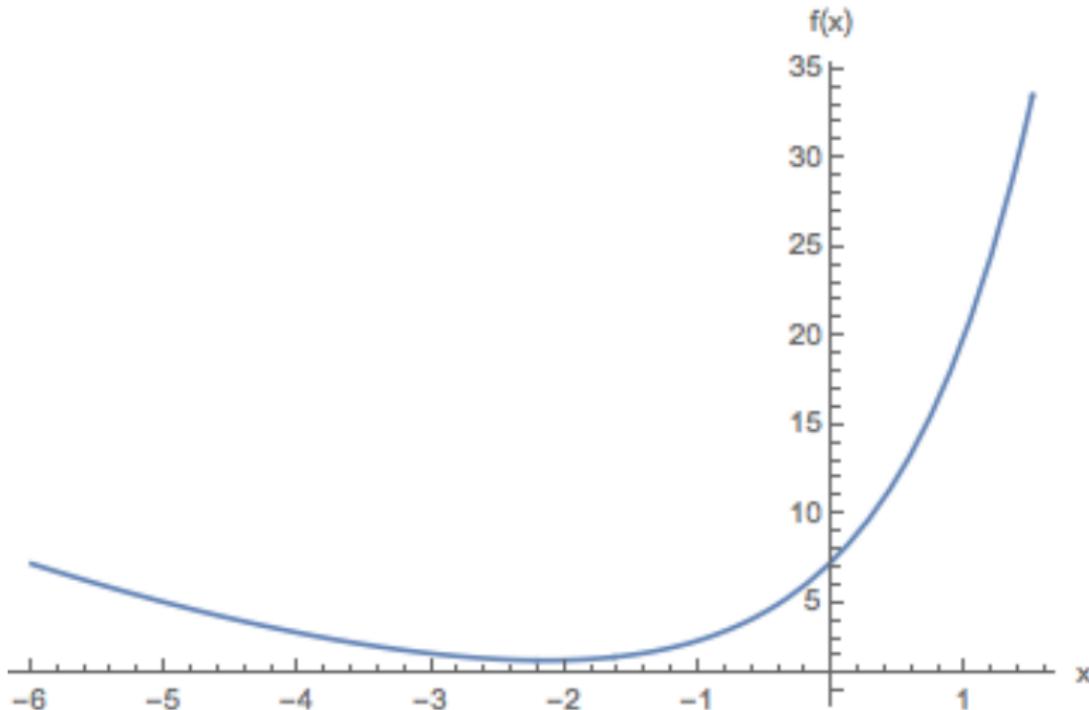


## GOLDEN SECTION SEARCH

- ▶ Golden Section Search is roughly what you use naturally when looking for a minimum.
- ▶ Univariate
- ▶ Want to find a minimum  $a$  and  $b$
- ▶ Denote new values of  $a$  and  $b$  with  $a'$  and  $b'$ .
  1. Take interval defined by  $(a, b)$ ,  $a < b$ .
  2. Evaluate  $c = b - \phi(b - a)$ ,  $d = a + \phi(b - a)$
  3. If  $f(c) > f(d)$ , then  $a' = c$ ,  $b' = b$ . If  $f(c) < f(d)$ ,  $a' = a$ ,  $b' = d$ .
  4. Repeat.
- ▶ Initial choice of three points: given  $a$  and  $c$ , choose  $b = 1.618a$ , golden ratio.
- ▶ One can show that this ensures that no matter which point is best,  $a$ ,  $b$ ,  $c$ ,  $d$  always keep same relative distance: this avoids just using one interval all the time because we were too close to minimum with first point.

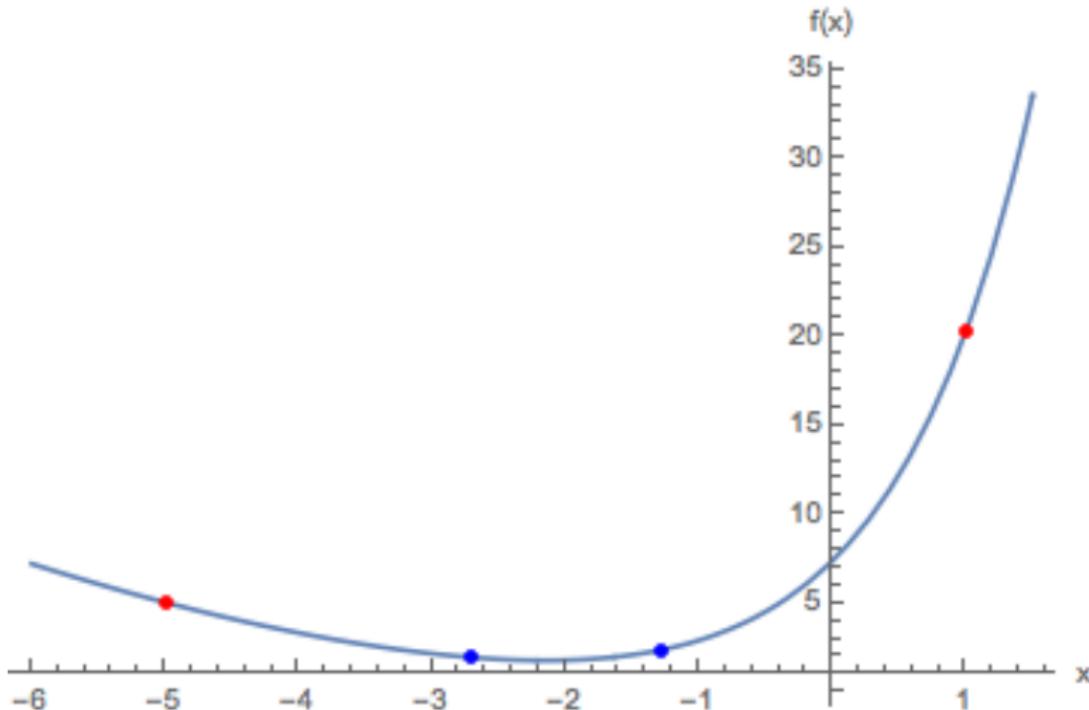
## GOLDEN SECTION SEARCH, GRAPHICALLY

$$f(x) = \frac{x^2}{5} + e^{x+2}$$



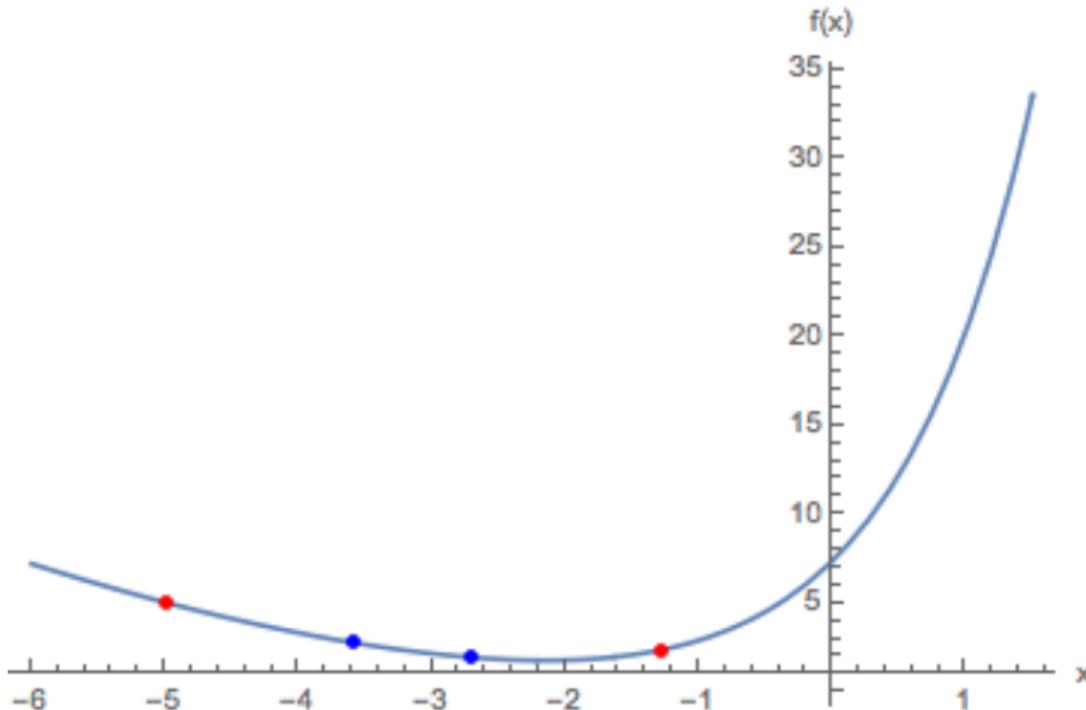
## GOLDEN SECTION SEARCH, GRAPHICALLY

$$f(x) = \frac{x^2}{5} + e^{x+2}$$



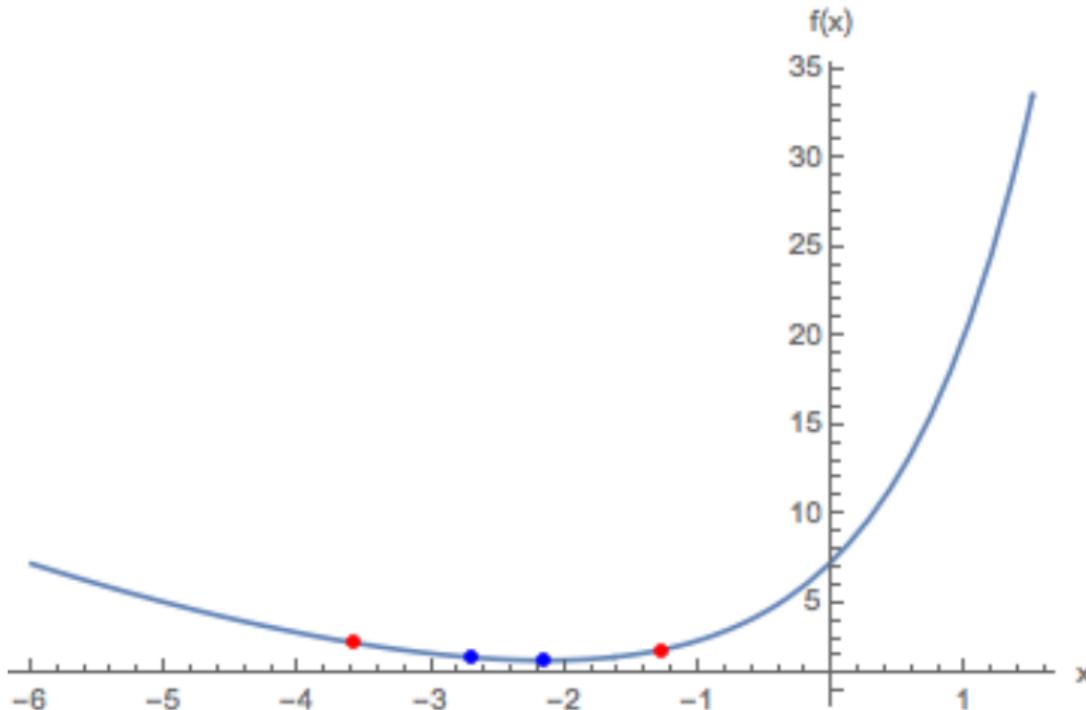
## GOLDEN SECTION SEARCH, GRAPHICALLY

$$f(x) = \frac{x^2}{5} + e^{x+2}$$



## GOLDEN SECTION SEARCH, GRAPHICALLY

$$f(x) = \frac{x^2}{5} + e^{x+2}$$



## UNIVARIATE METHODS

- ▶ Univariate methods have limited usefulness
- ▶ Bisection frequently useful when you have bounds and a simple problem: negotiation weights problem, how much to work, etc.
- ▶ Alternatively, Golden section search is more often used by taking multivariate problem, finding direction, then minimizing using Golden section search, repeating

## CONJUGATE GRADIENT METHOD

- ▶ Conjugate gradient doesn't require Jacobian, just gradient.
- ▶ We're going to keep track of search direction  $S_n$  (a vector)
- ▶ For legibility, define steepest direction  $\Delta X_n = -\nabla_x f(X_N)$
- ▶ Start first search direction as steepest descent:  $S_0 = \Delta X_0$

# CONJUGATE GRADIENT METHOD

- ▶ Starting at  $t = 0$ , with  $S_0 = -\nabla f(X_0)$ 
  1. Calculate  $\Delta X_t$  by taking gradient.
  2. If  $t > 0$ , update search direction:  $S_t = \Delta X_t + \frac{\Delta X'_t \Delta X_t}{\Delta X'_{t-1} \Delta X_{t-1}} S_{t-1}$
  3. Along the plane defined by  $S_t$ , find  $\alpha^*$  to minimize  $f$ :

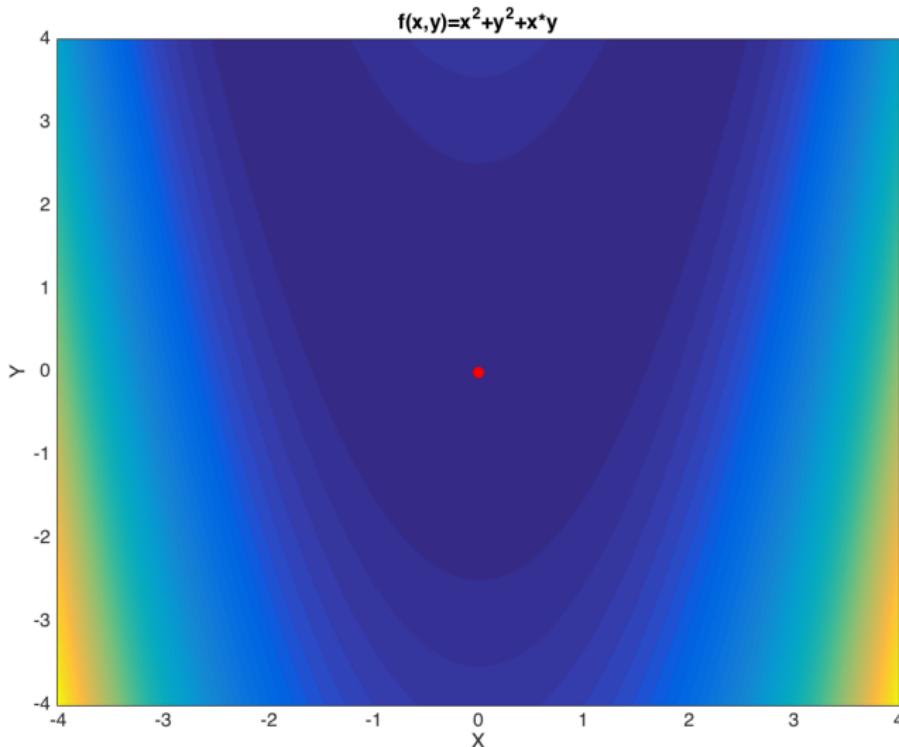
$$\arg \min_{\alpha} f(X_t + \alpha_t S_t)$$

4. Using  $\alpha_t^*$ , update  $X_{t+1}$ :

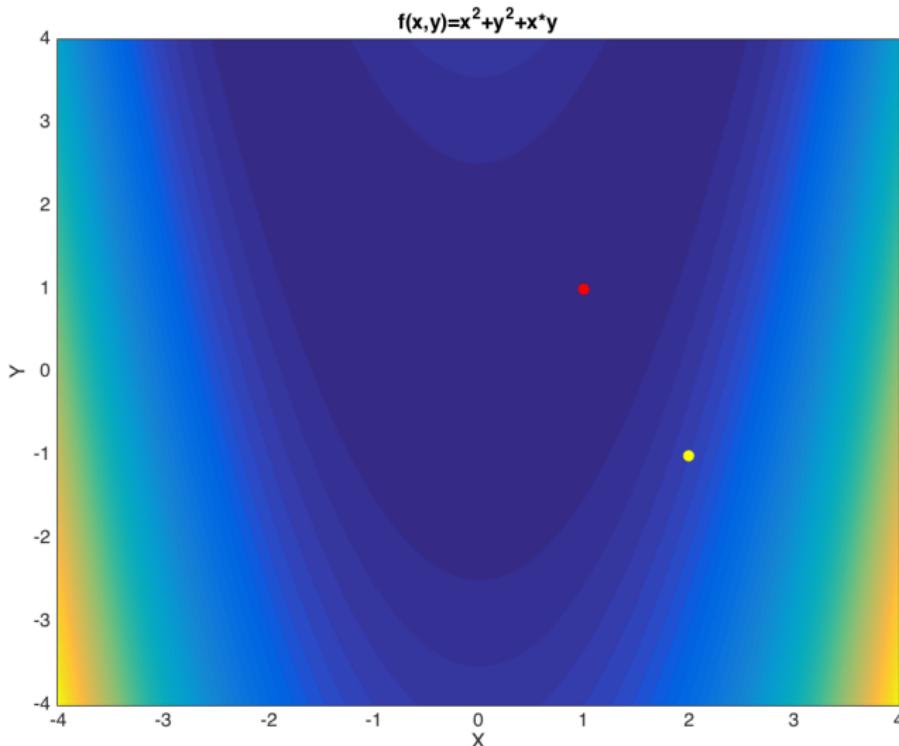
$$X_{t+1} = X_t + \alpha_t^* S_t$$

5. Repeat

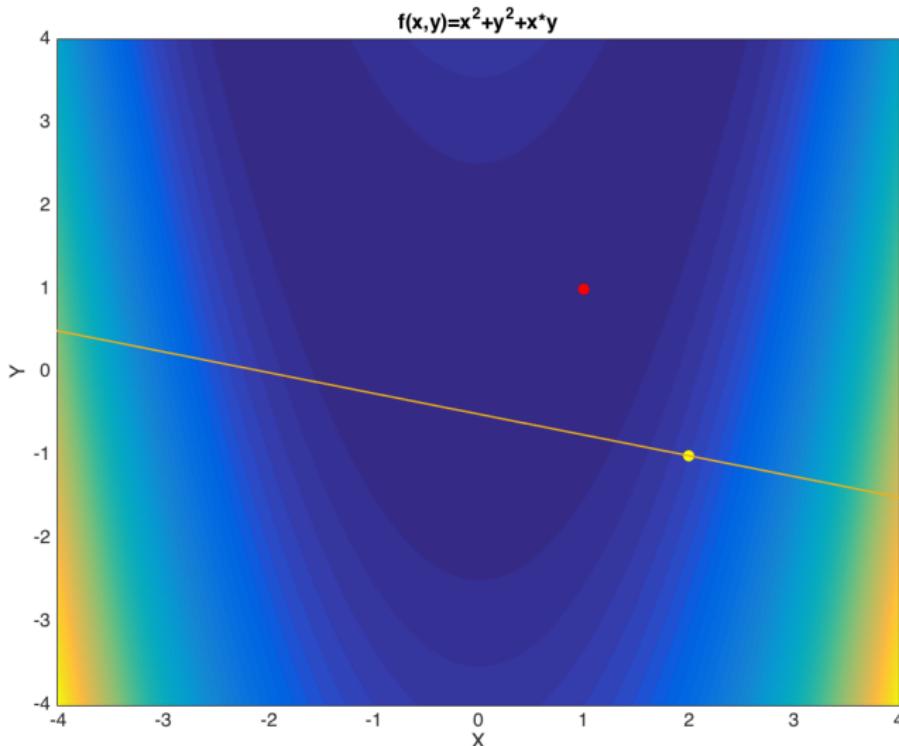
# CONJUGATE GRADIENT: EXAMPLES



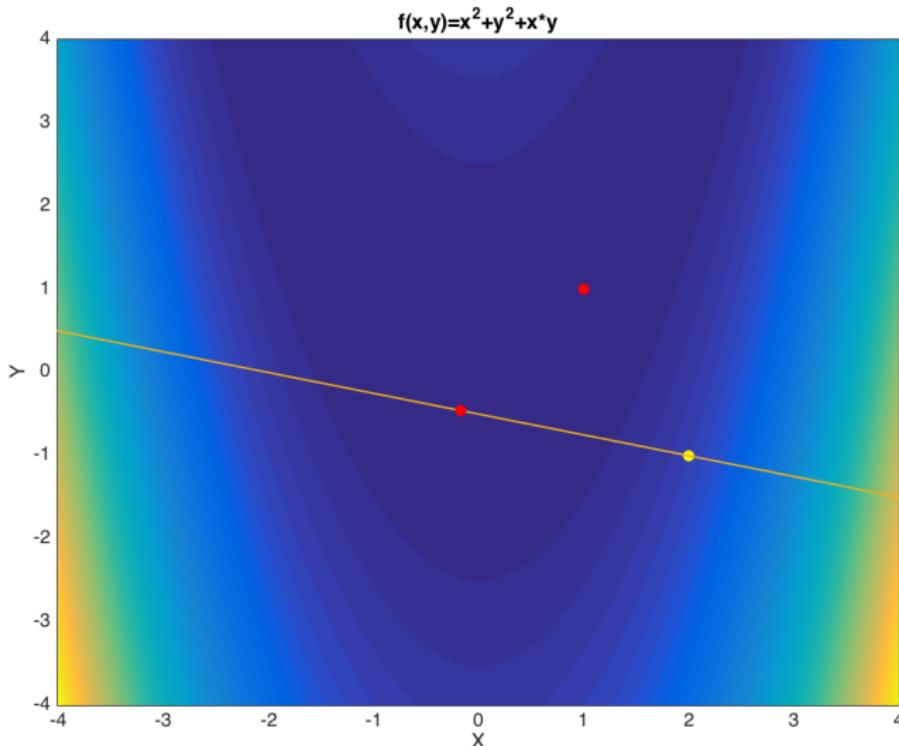
# CONJUGATE GRADIENT: EXAMPLES



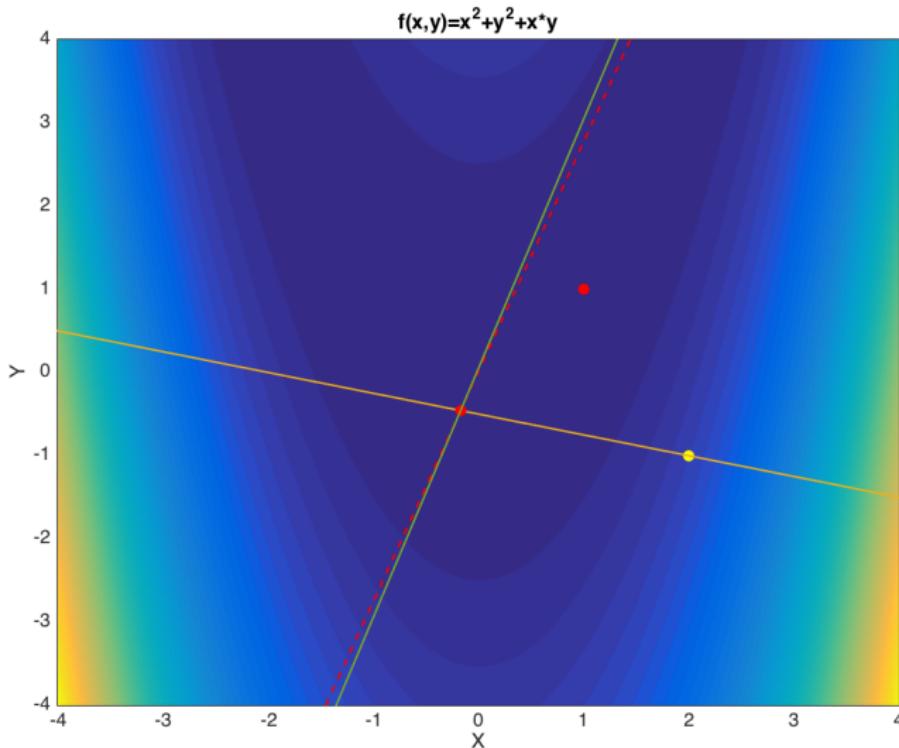
# CONJUGATE GRADIENT: EXAMPLES



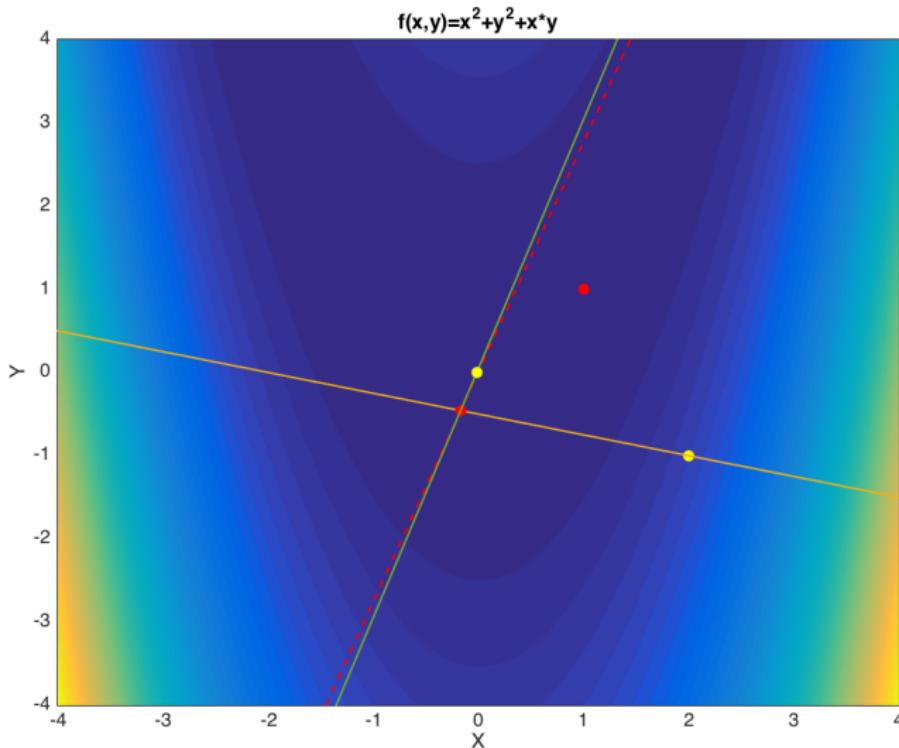
# CONJUGATE GRADIENT: EXAMPLES



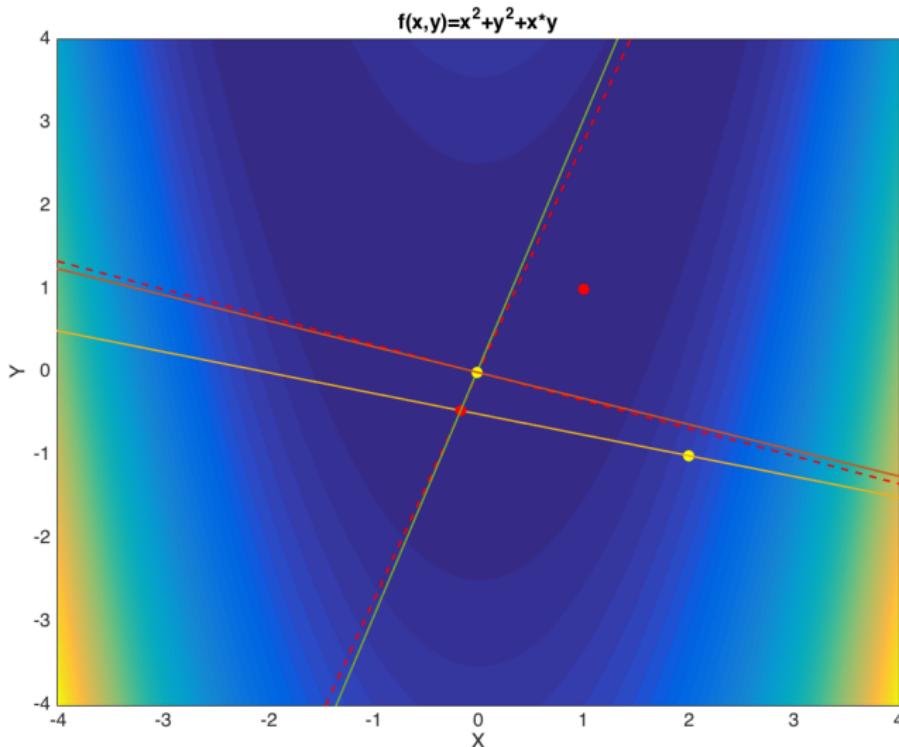
# CONJUGATE GRADIENT: EXAMPLES



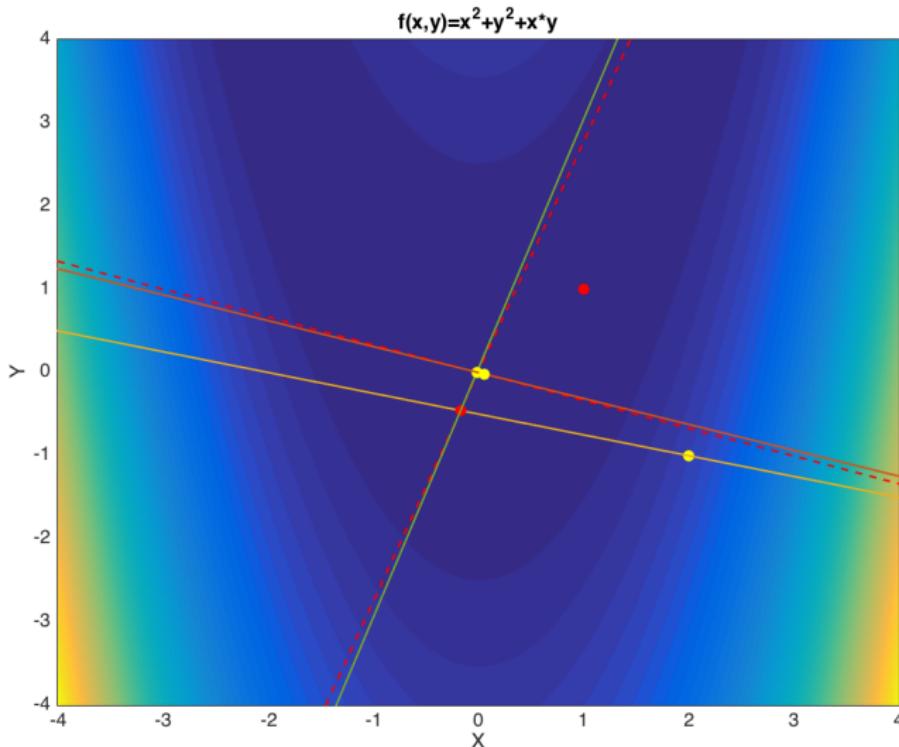
# CONJUGATE GRADIENT: EXAMPLES



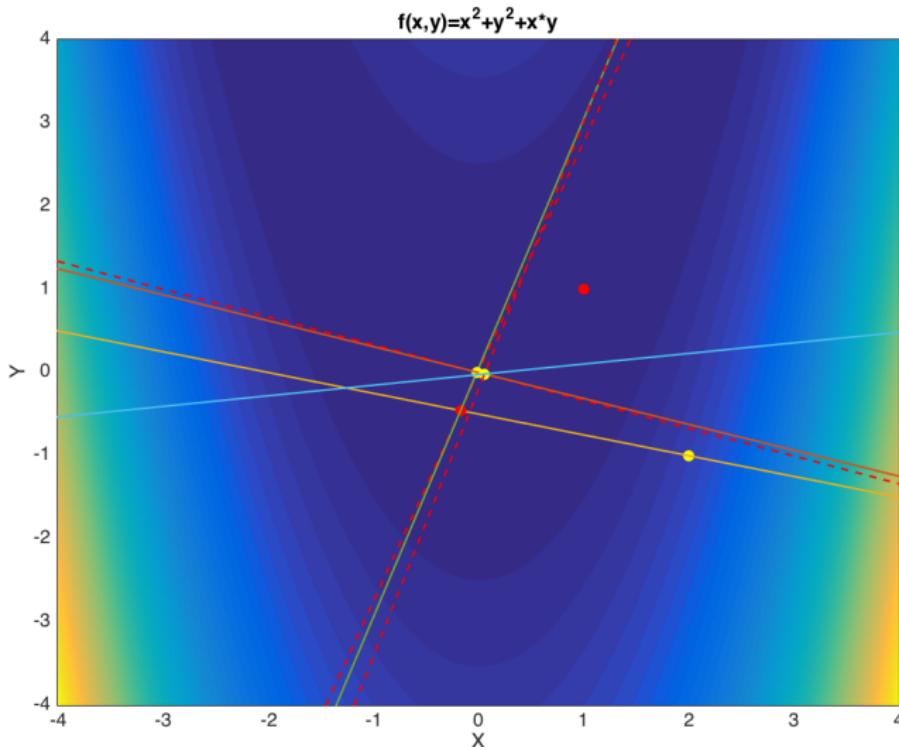
# CONJUGATE GRADIENT: EXAMPLES



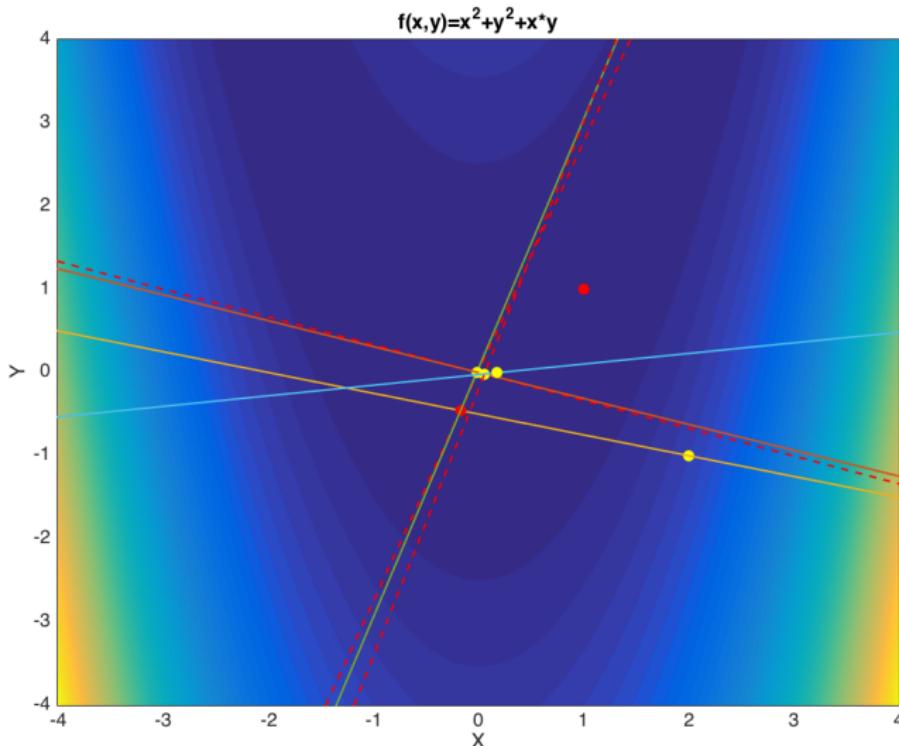
# CONJUGATE GRADIENT: EXAMPLES



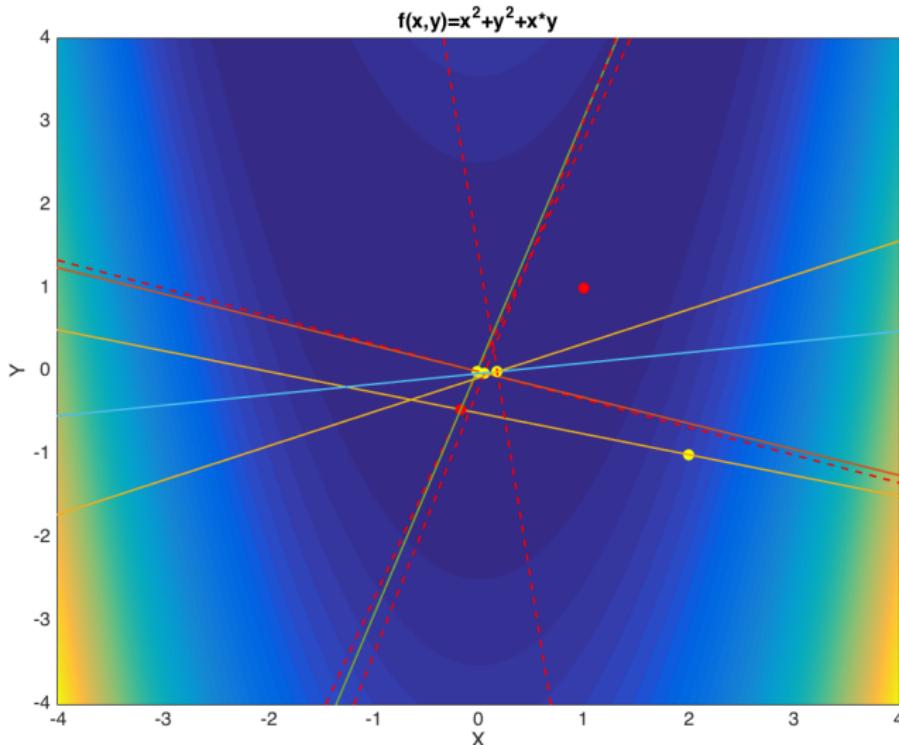
# CONJUGATE GRADIENT: EXAMPLES



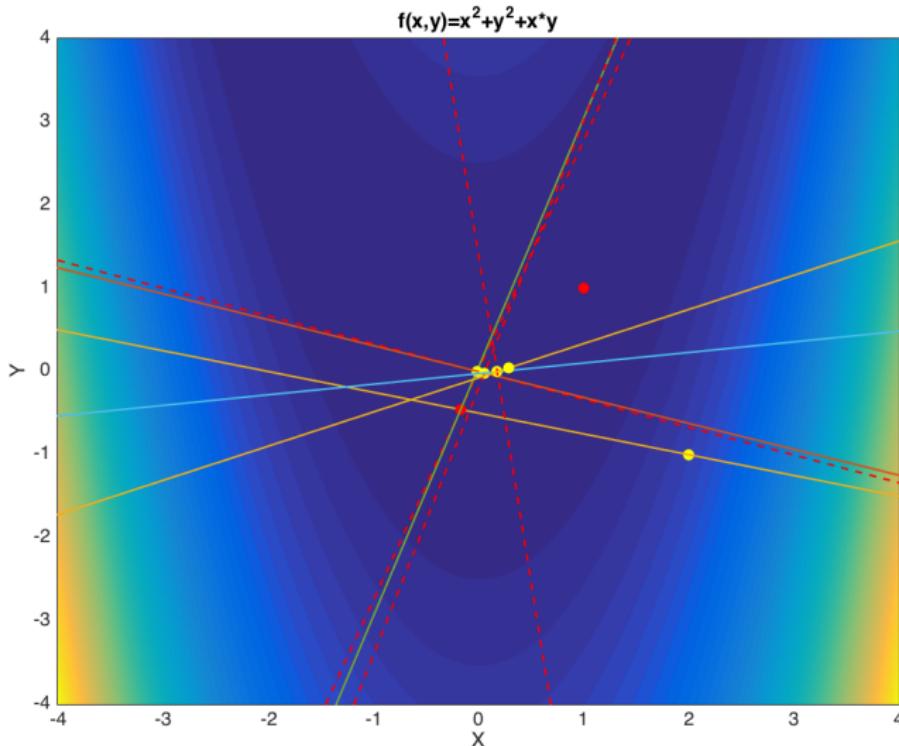
# CONJUGATE GRADIENT: EXAMPLES



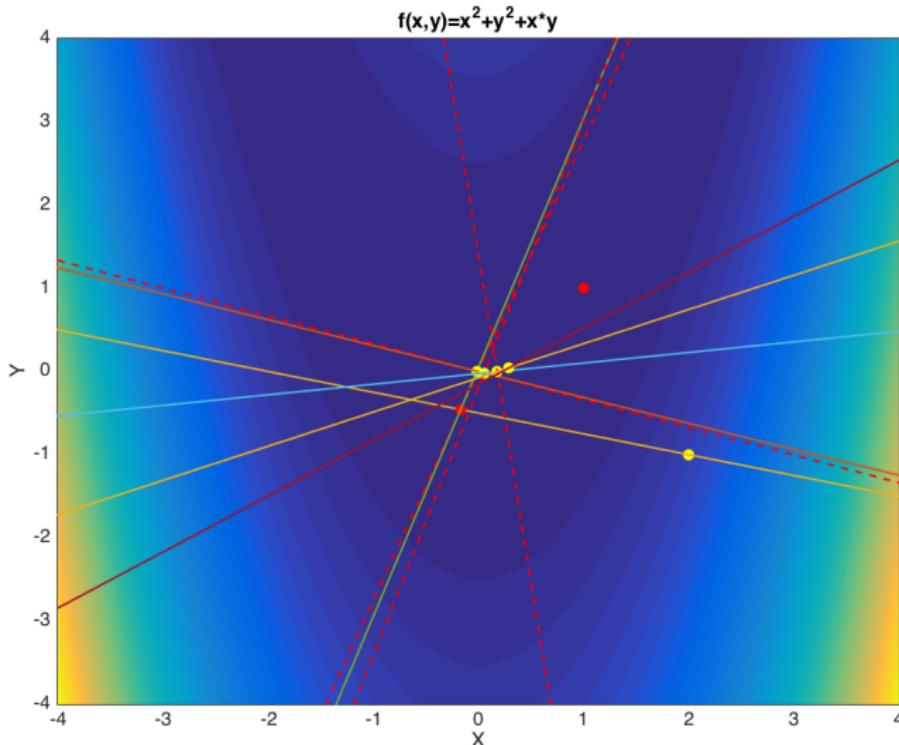
# CONJUGATE GRADIENT: EXAMPLES



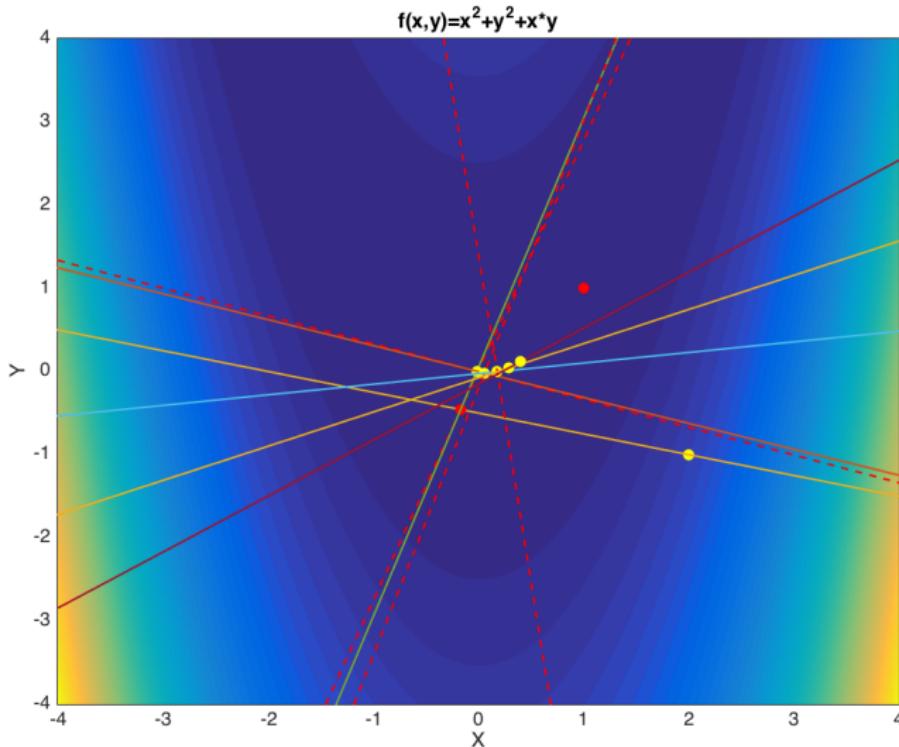
# CONJUGATE GRADIENT: EXAMPLES



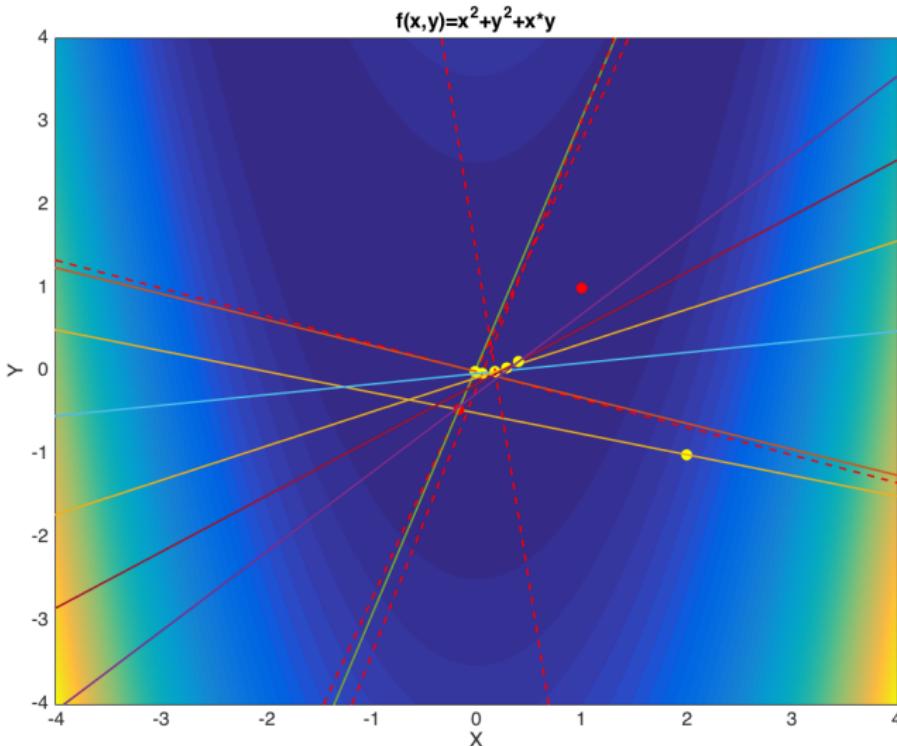
# CONJUGATE GRADIENT: EXAMPLES



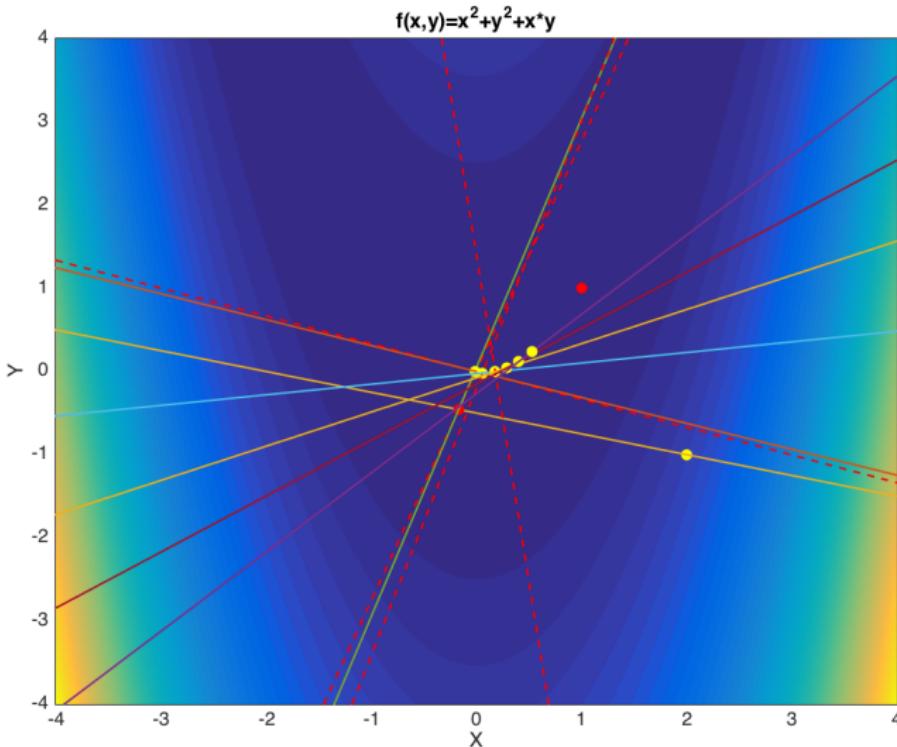
# CONJUGATE GRADIENT: EXAMPLES



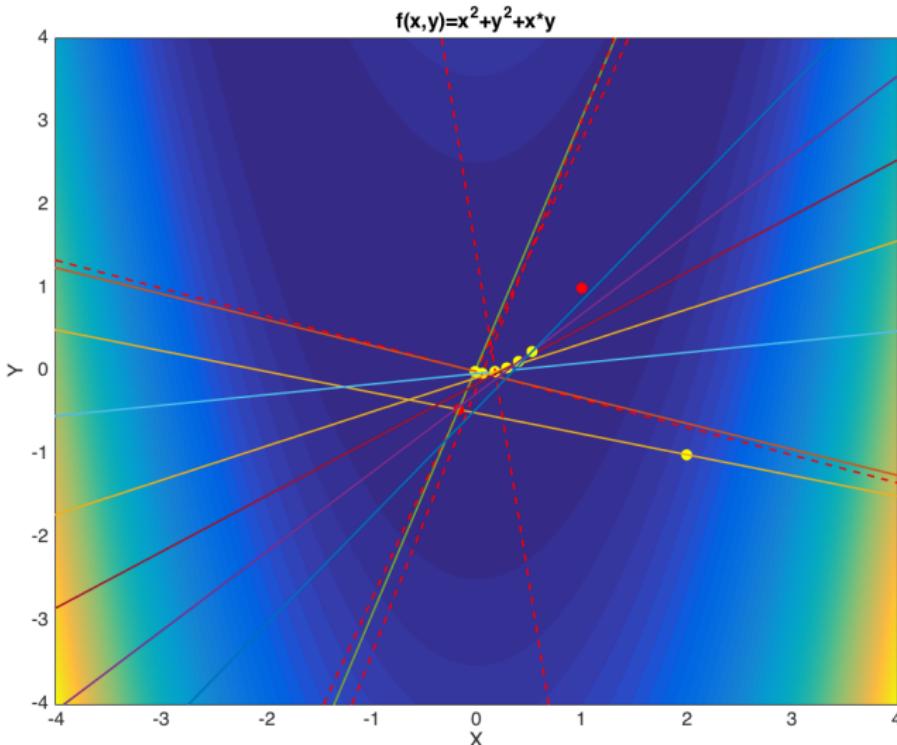
# CONJUGATE GRADIENT: EXAMPLES



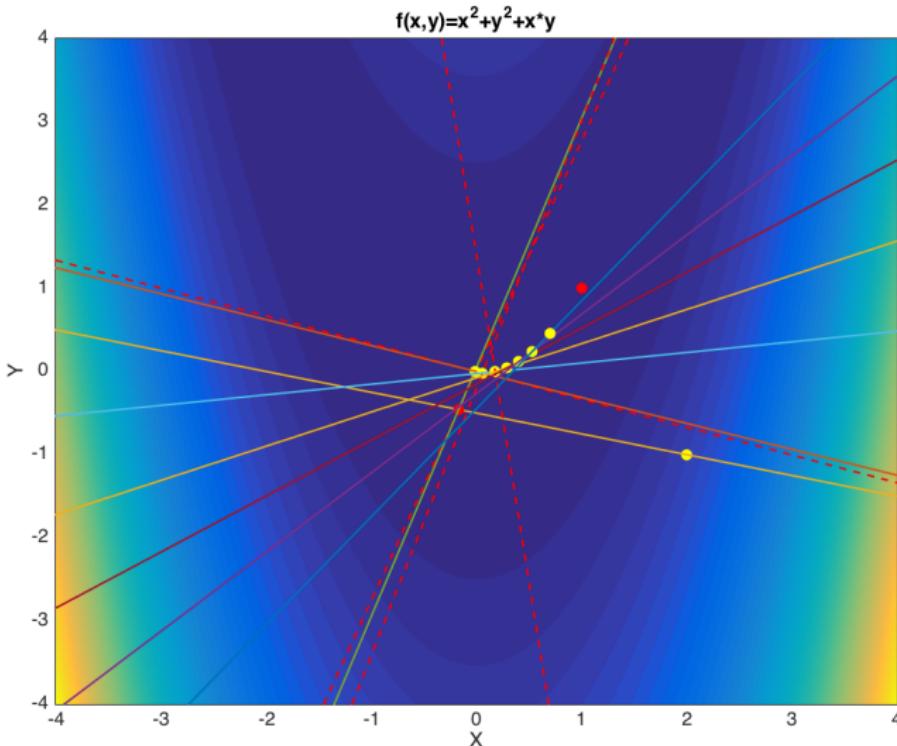
# CONJUGATE GRADIENT: EXAMPLES



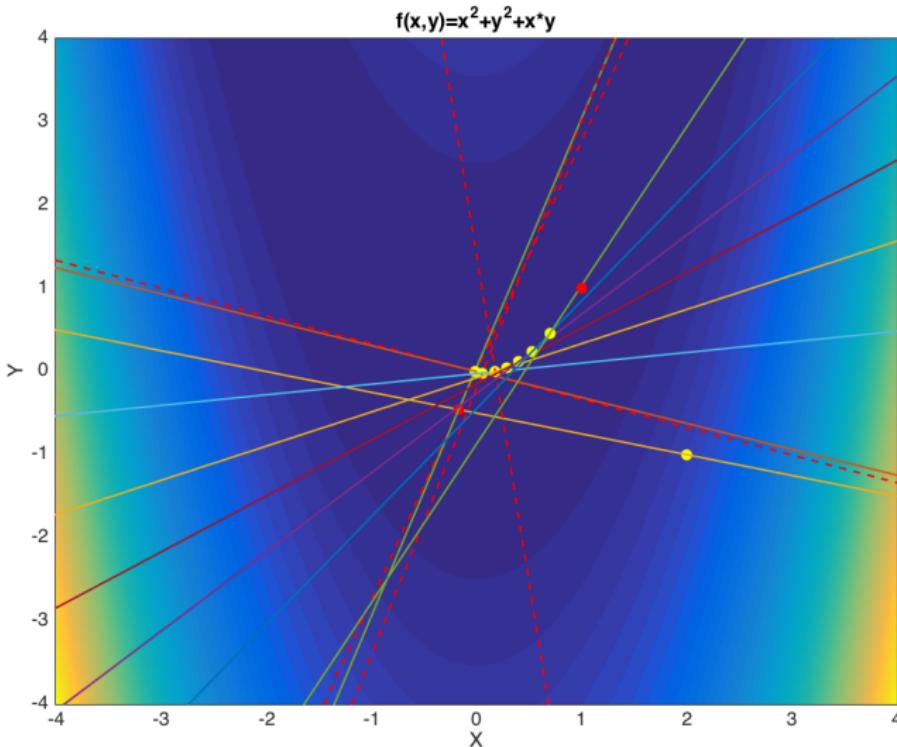
# CONJUGATE GRADIENT: EXAMPLES



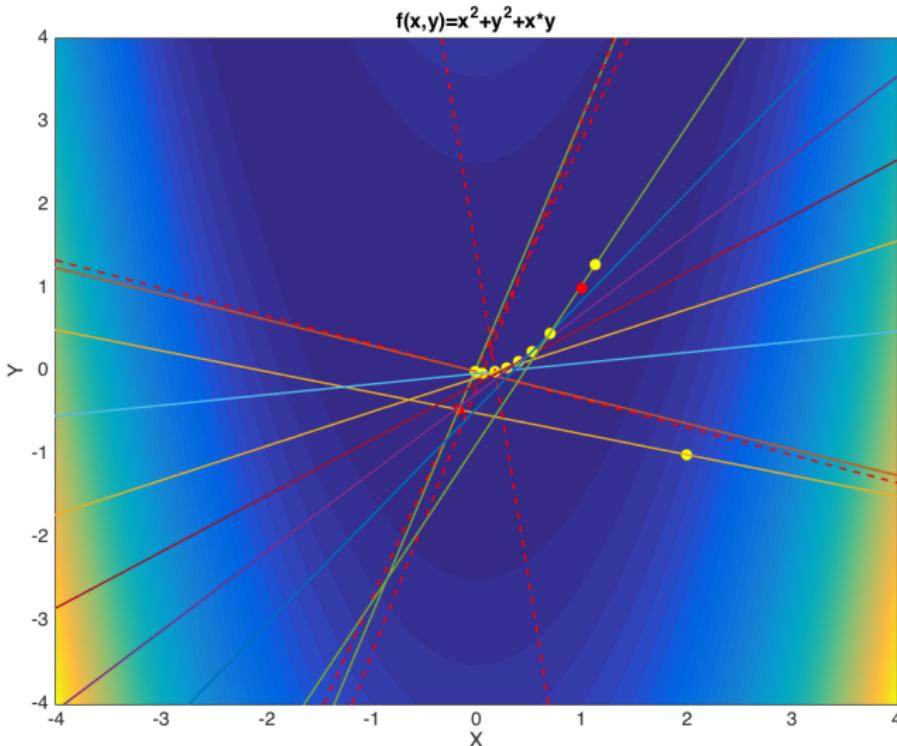
# CONJUGATE GRADIENT: EXAMPLES



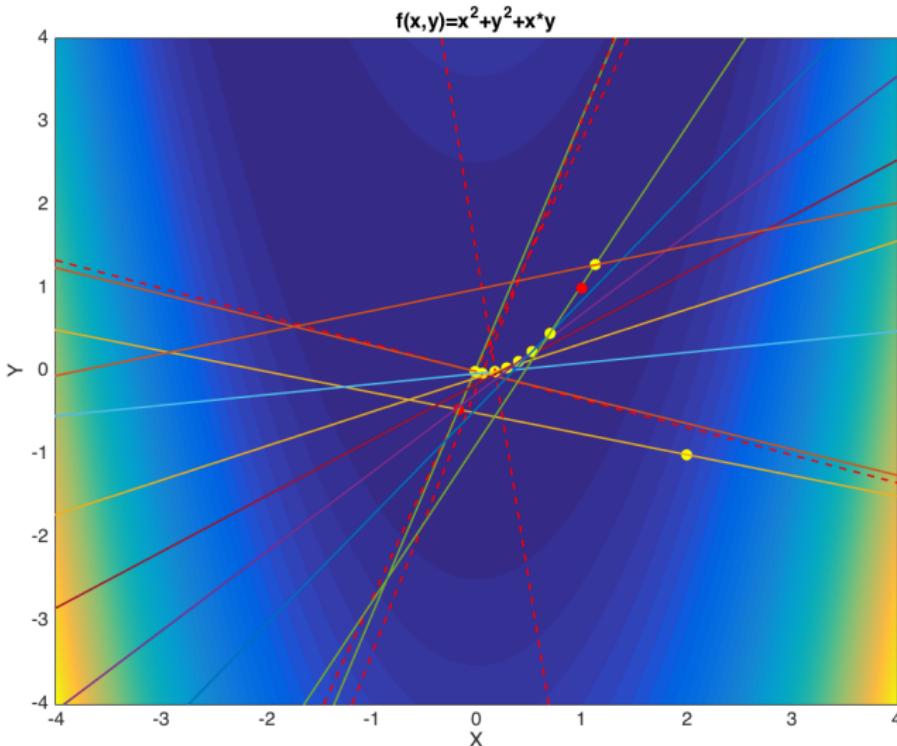
# CONJUGATE GRADIENT: EXAMPLES



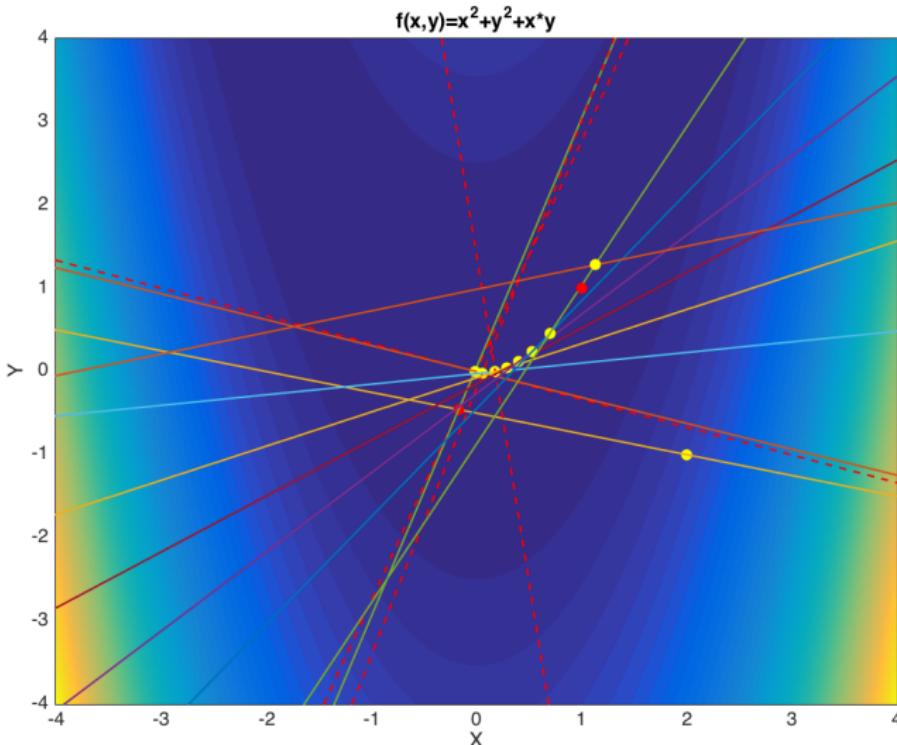
# CONJUGATE GRADIENT: EXAMPLES



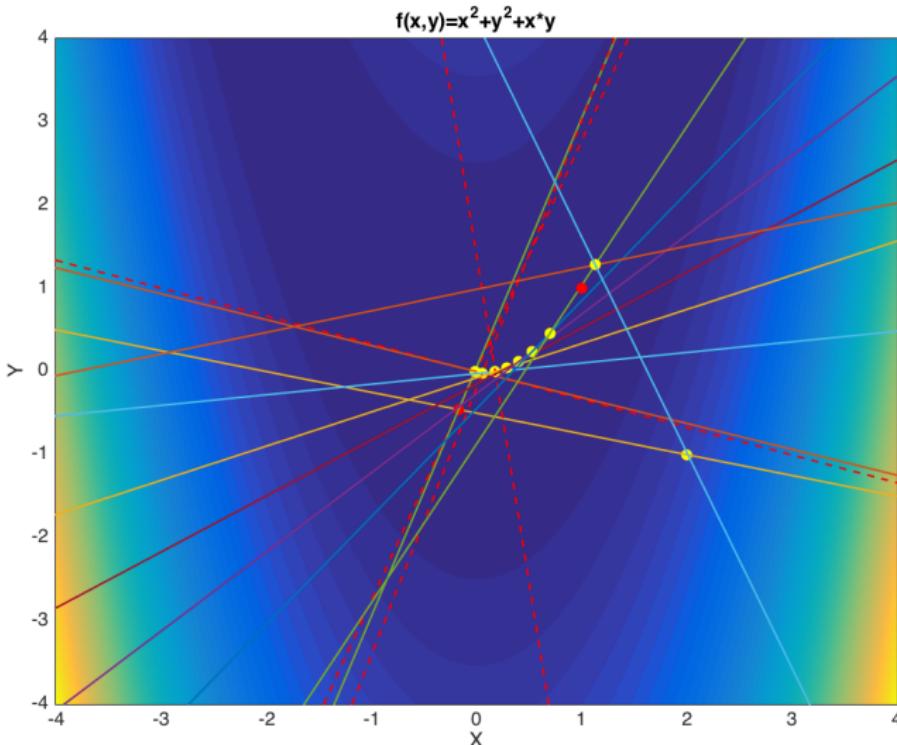
# CONJUGATE GRADIENT: EXAMPLES



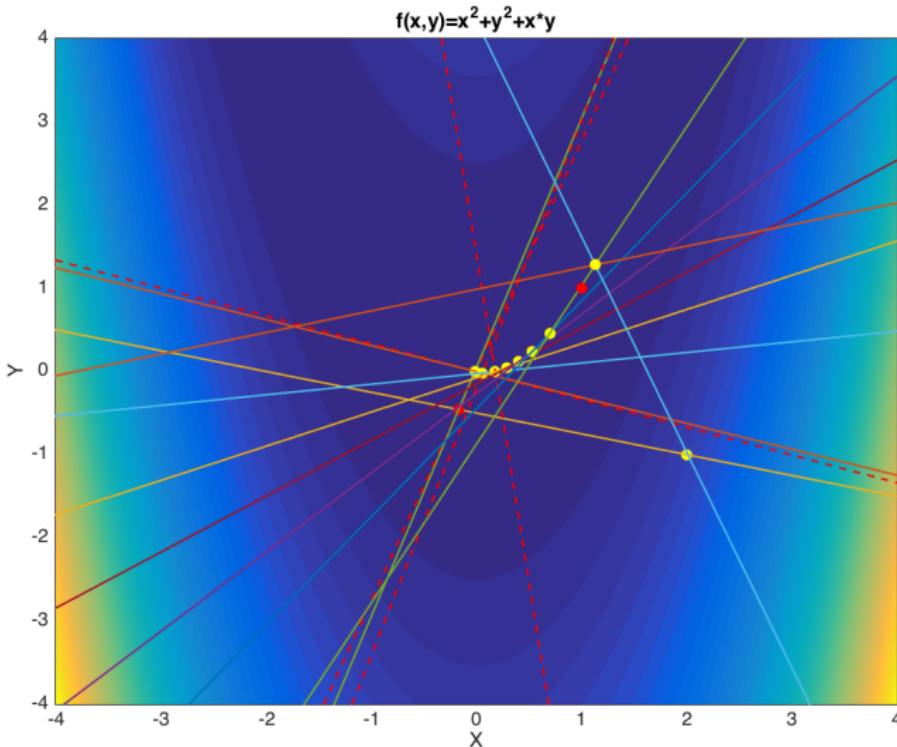
# CONJUGATE GRADIENT: EXAMPLES



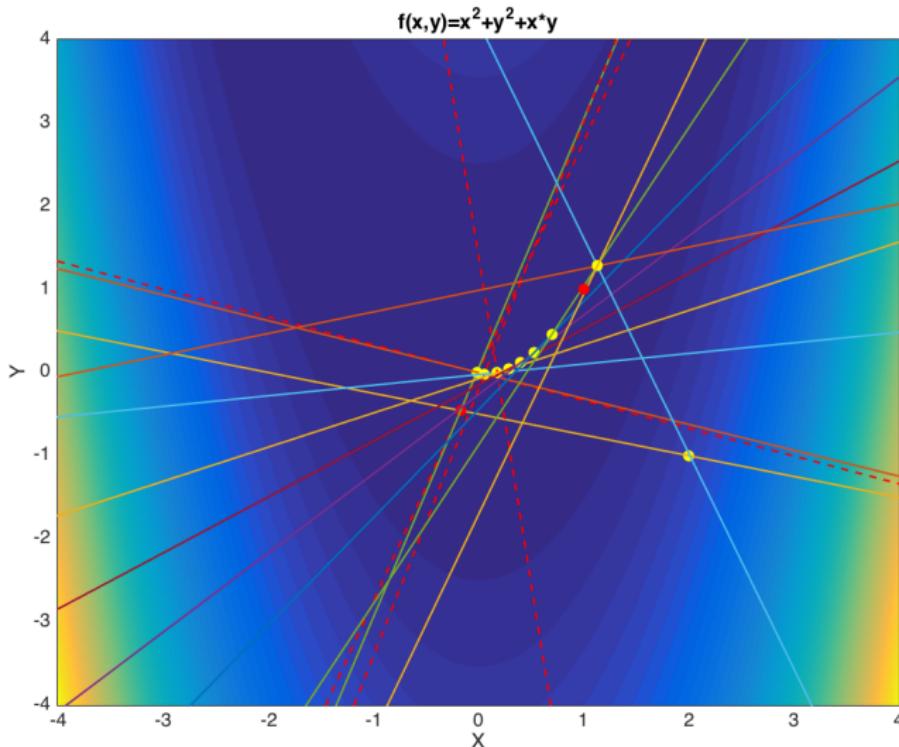
# CONJUGATE GRADIENT: EXAMPLES



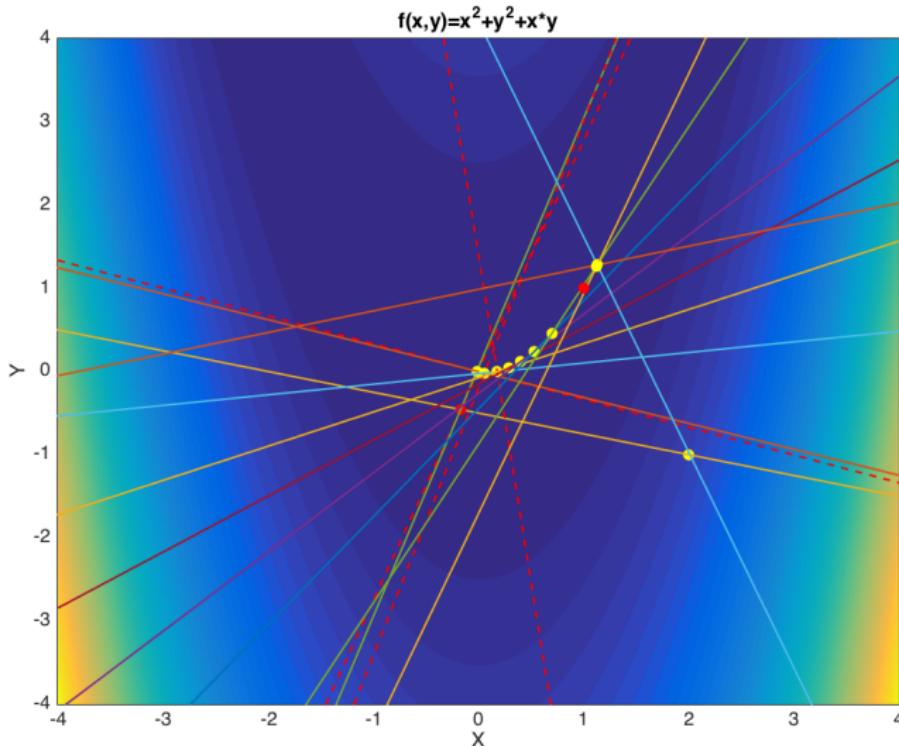
# CONJUGATE GRADIENT: EXAMPLES



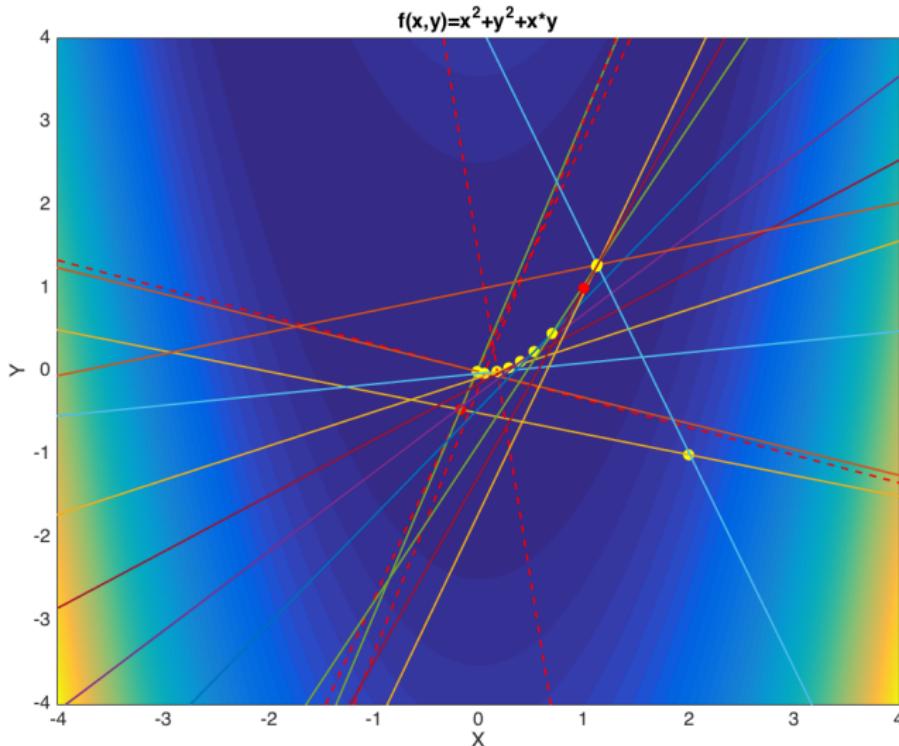
# CONJUGATE GRADIENT: EXAMPLES



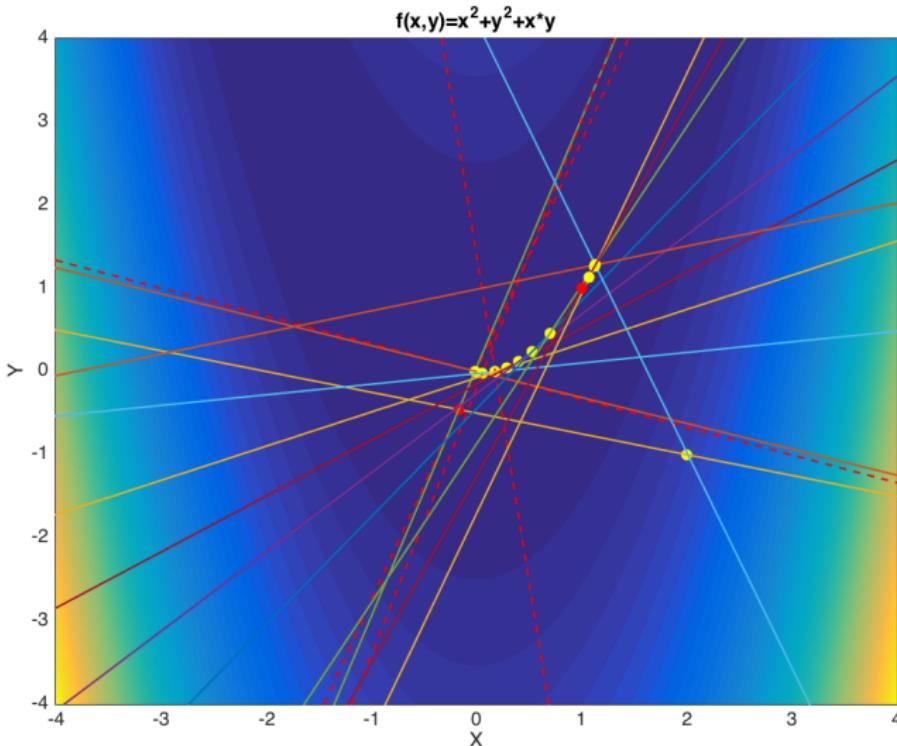
# CONJUGATE GRADIENT: EXAMPLES



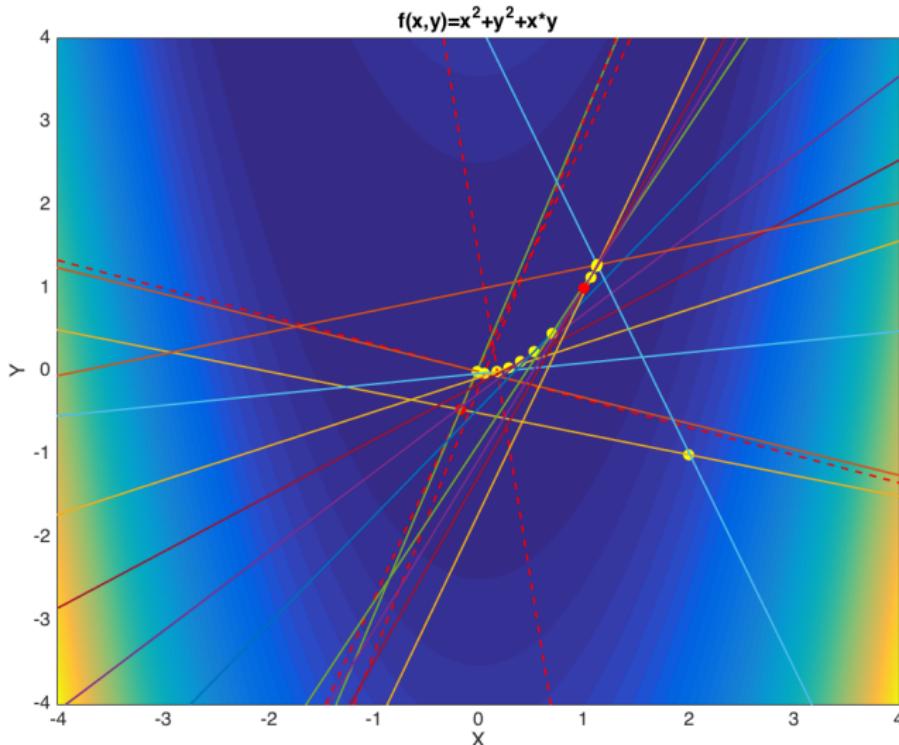
# CONJUGATE GRADIENT: EXAMPLES



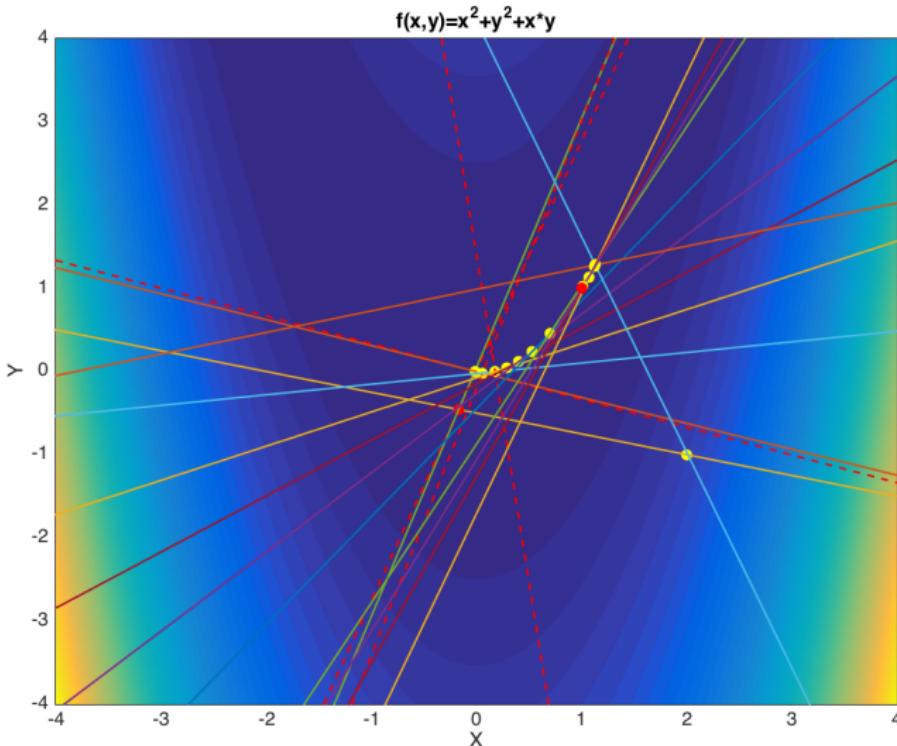
# CONJUGATE GRADIENT: EXAMPLES



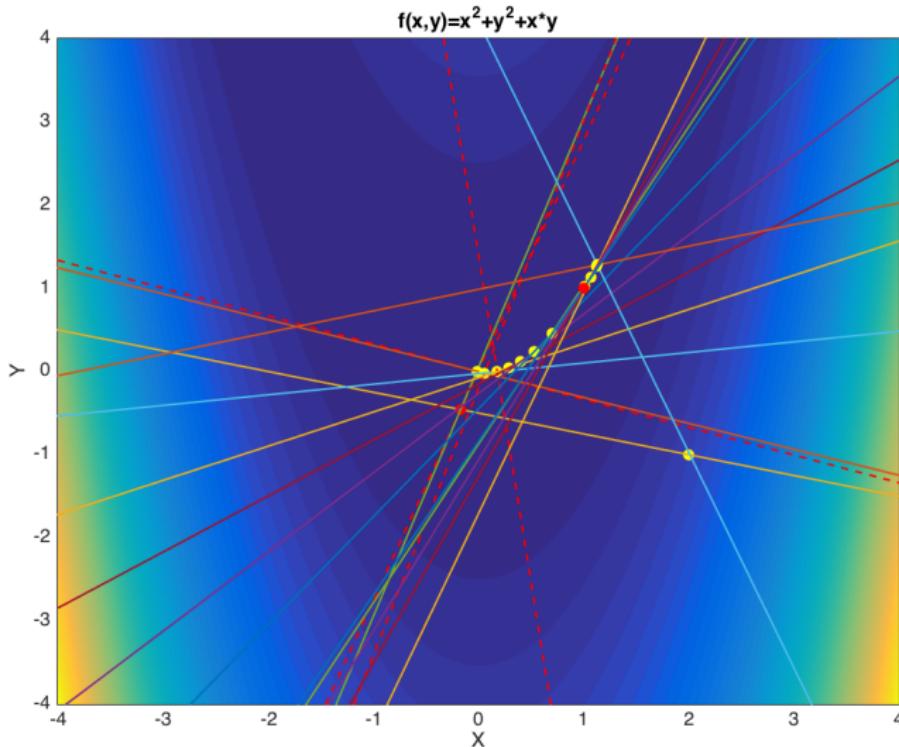
# CONJUGATE GRADIENT: EXAMPLES



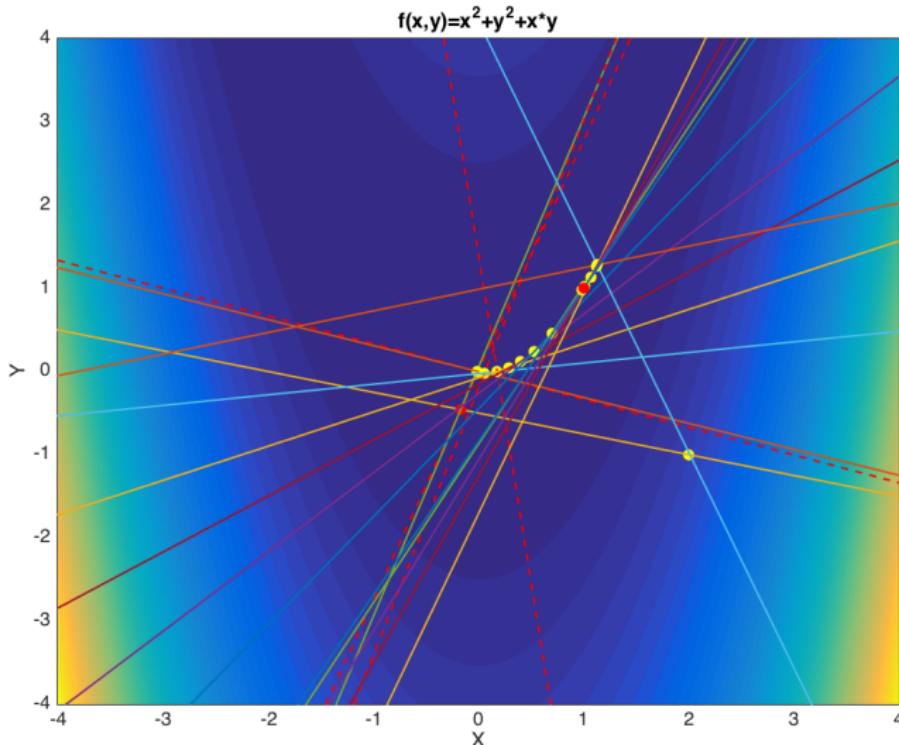
# CONJUGATE GRADIENT: EXAMPLES



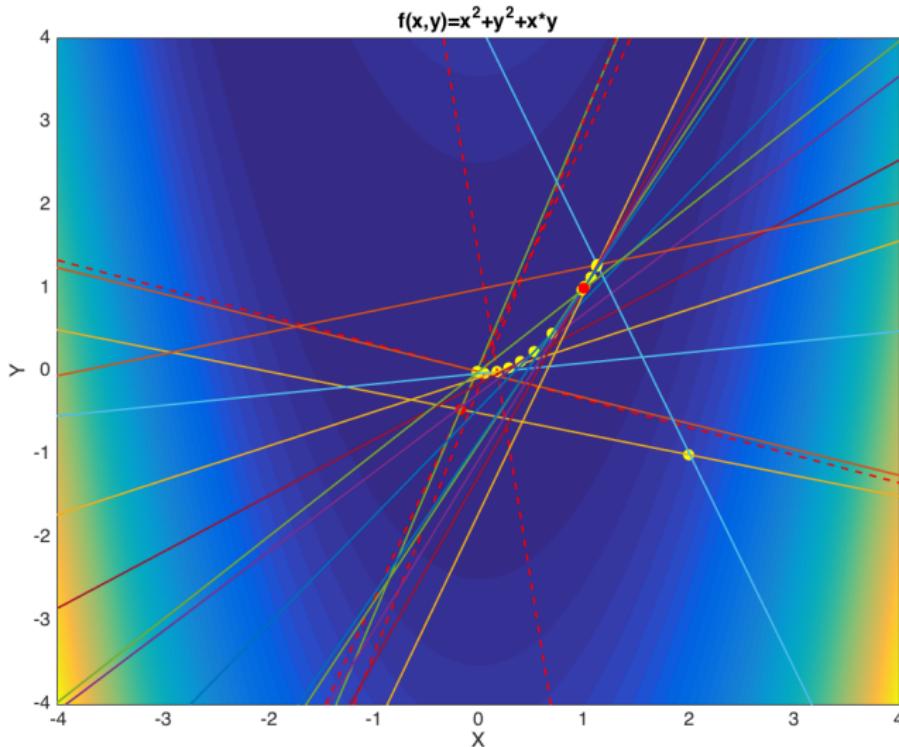
# CONJUGATE GRADIENT: EXAMPLES



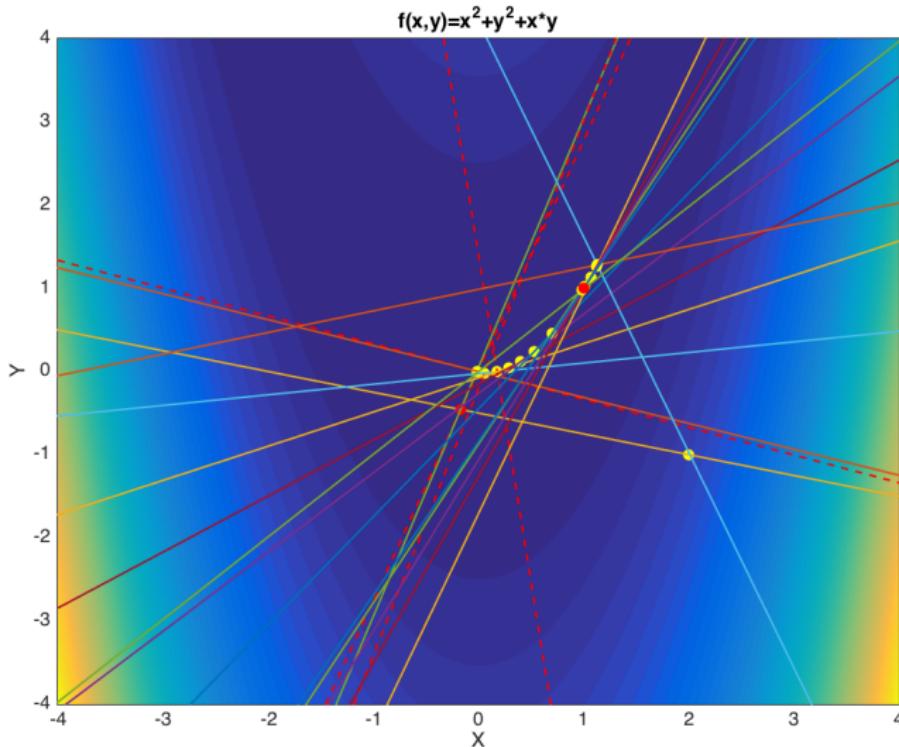
# CONJUGATE GRADIENT: EXAMPLES



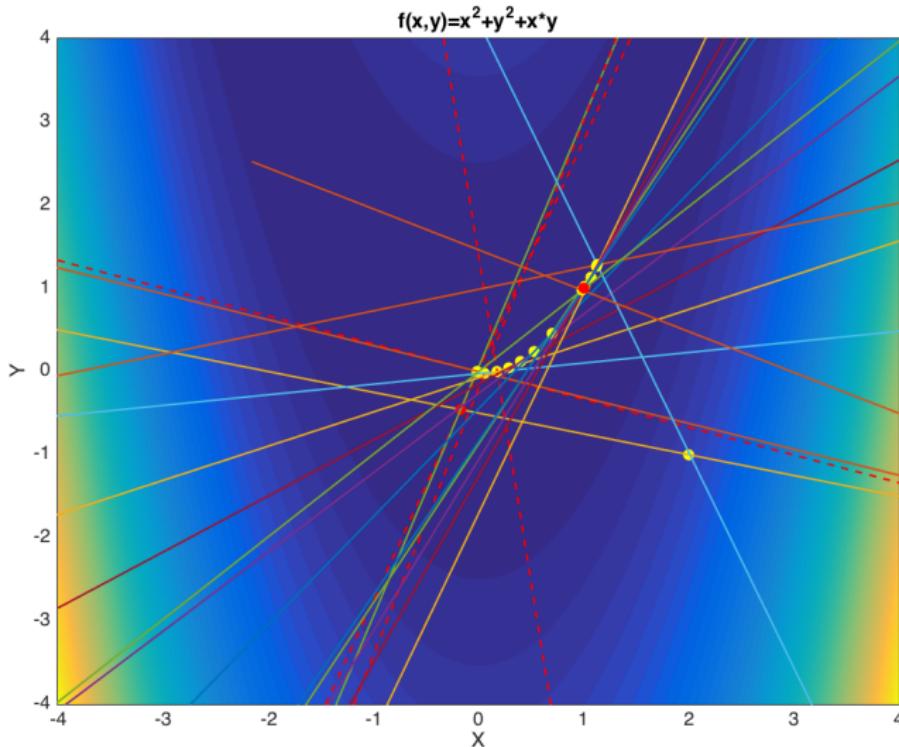
# CONJUGATE GRADIENT: EXAMPLES



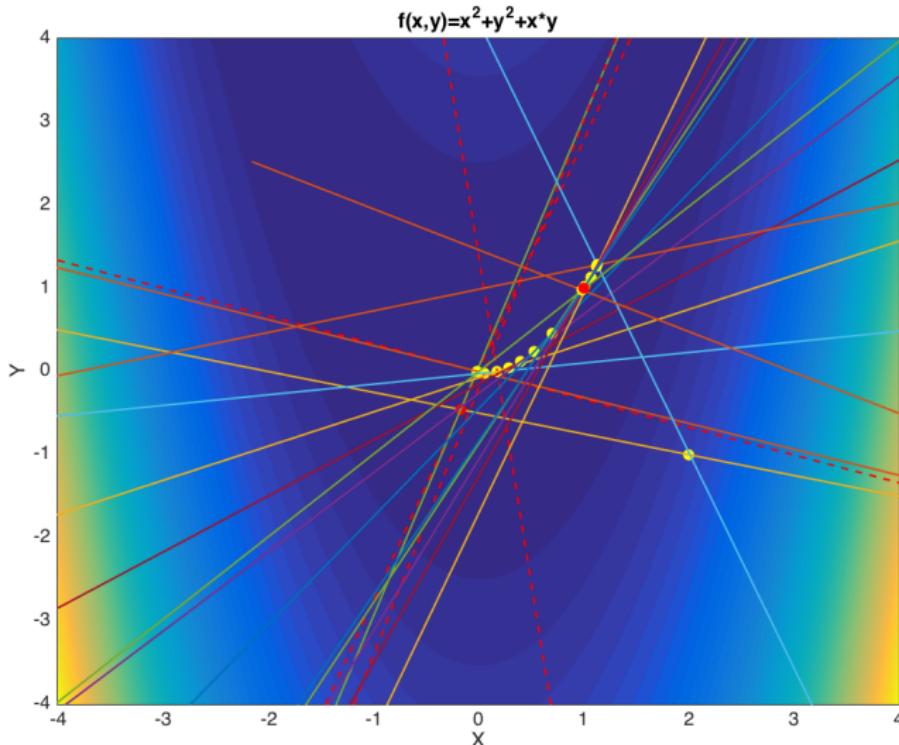
# CONJUGATE GRADIENT: EXAMPLES



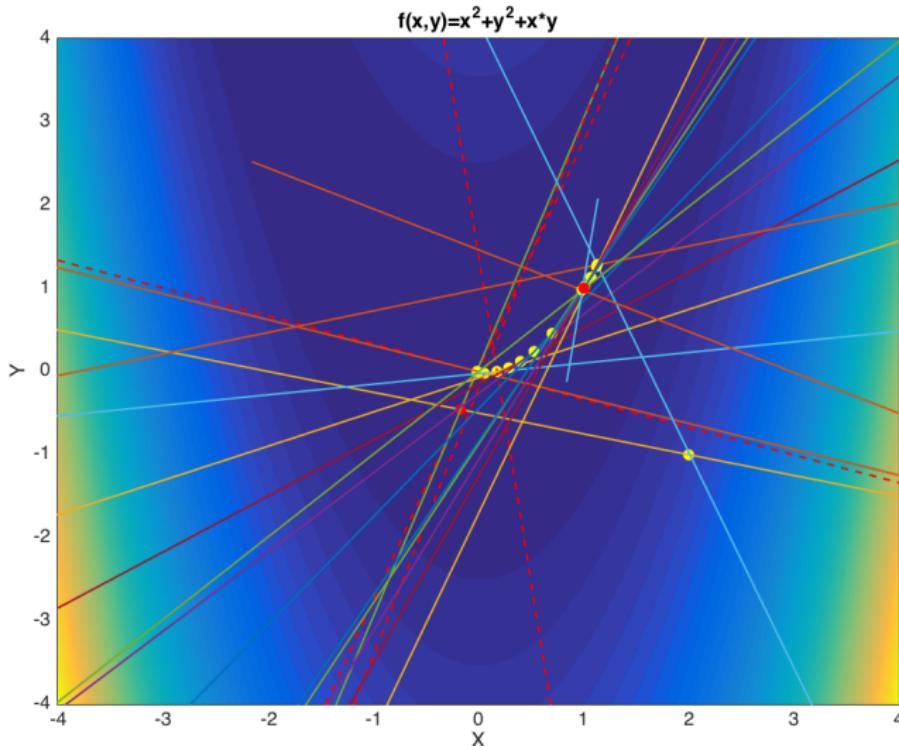
# CONJUGATE GRADIENT: EXAMPLES



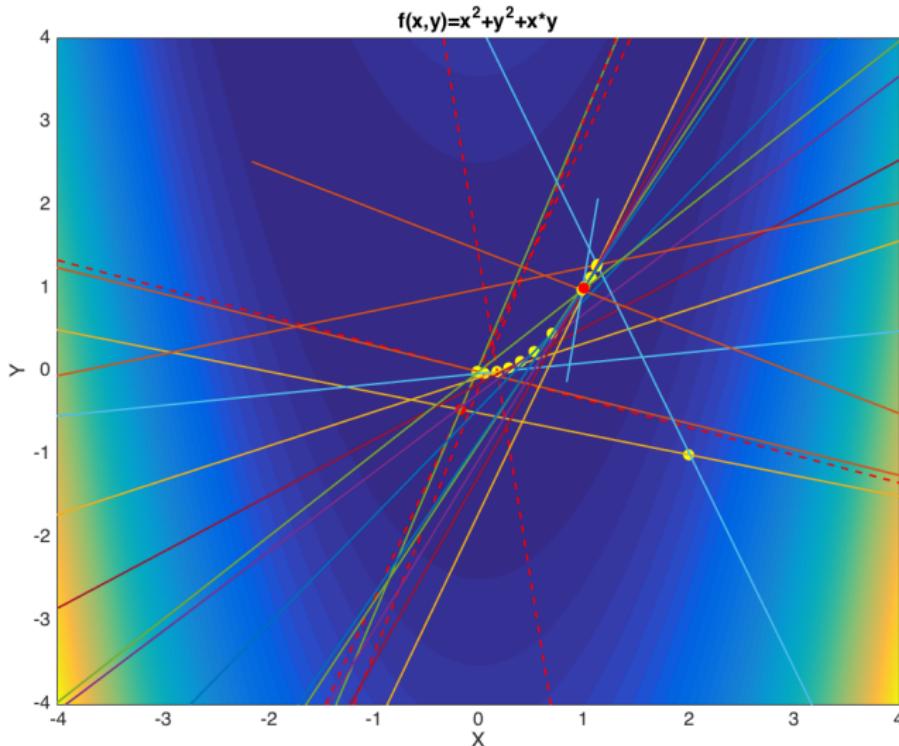
# CONJUGATE GRADIENT: EXAMPLES



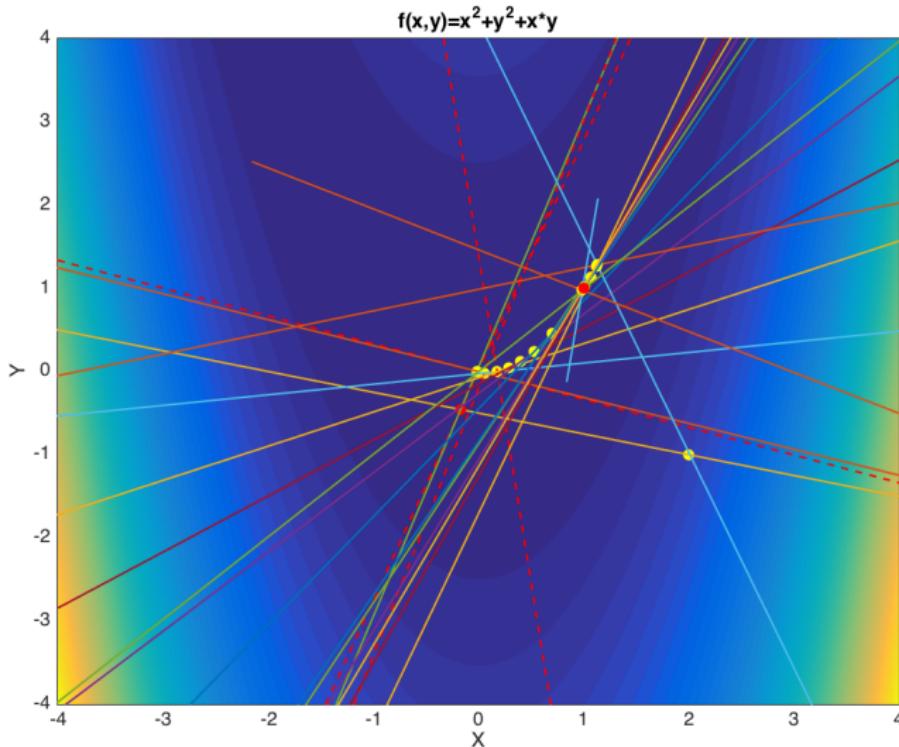
# CONJUGATE GRADIENT: EXAMPLES



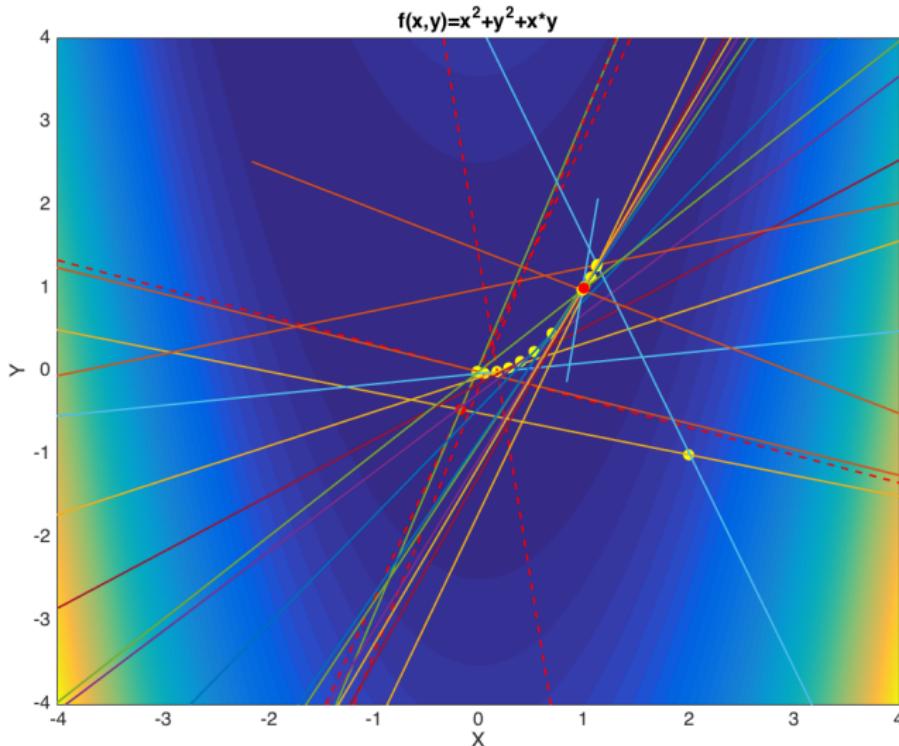
# CONJUGATE GRADIENT: EXAMPLES



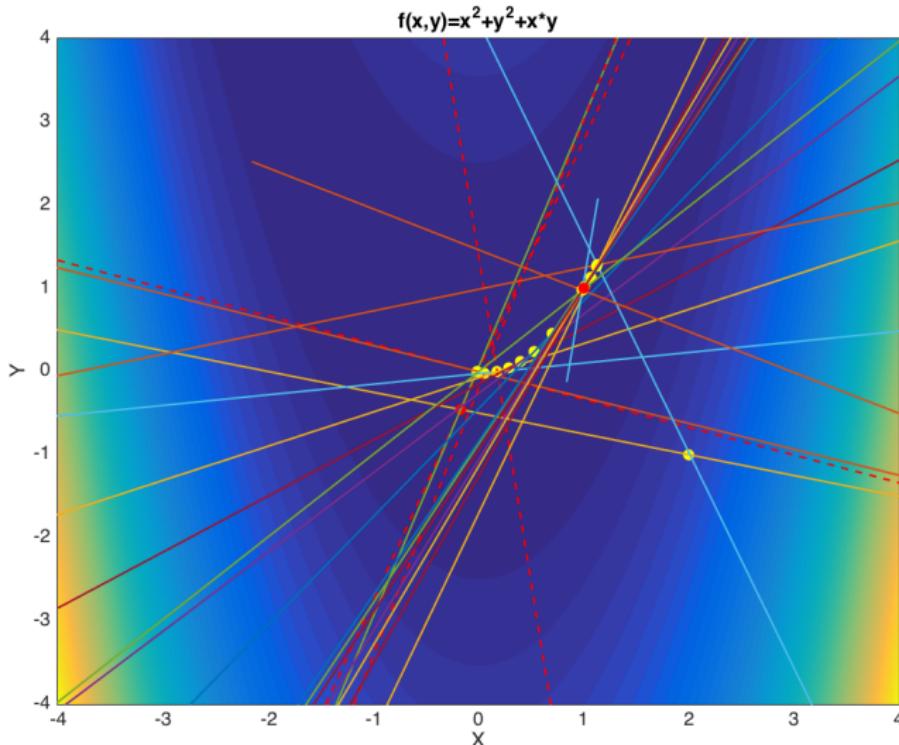
# CONJUGATE GRADIENT: EXAMPLES



# CONJUGATE GRADIENT: EXAMPLES



# CONJUGATE GRADIENT: EXAMPLES



# DERIVATIVE-FREE METHODS

- ▶ Nelder-Mead Simplex
- ▶ Pattern search
- ▶ Differential evolution/Genetic algorithms
- ▶ Simulated Annealing

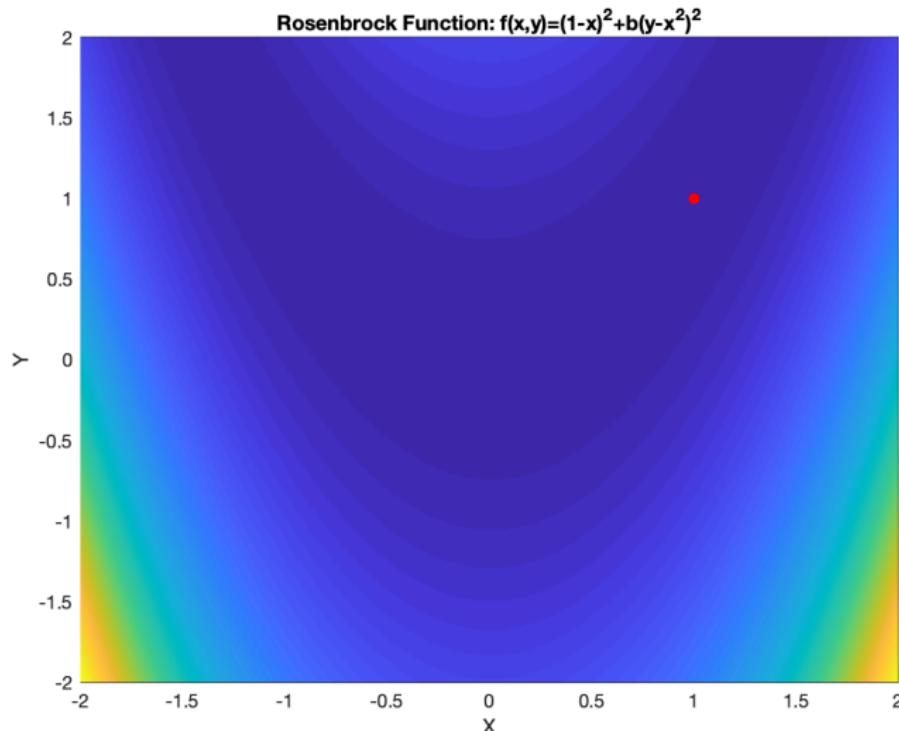
## NELDER MEAD

- ▶ Nelder-Mead is fun, simple, fairly robust
- ▶ Makes great intuitive sense
- ▶ Easy for it to stop at local minima
- ▶ Not as fast as derivative-based
- ▶ Not as slow as differential evolution
- ▶ If it starts out making sense, it rarely ends not making sense

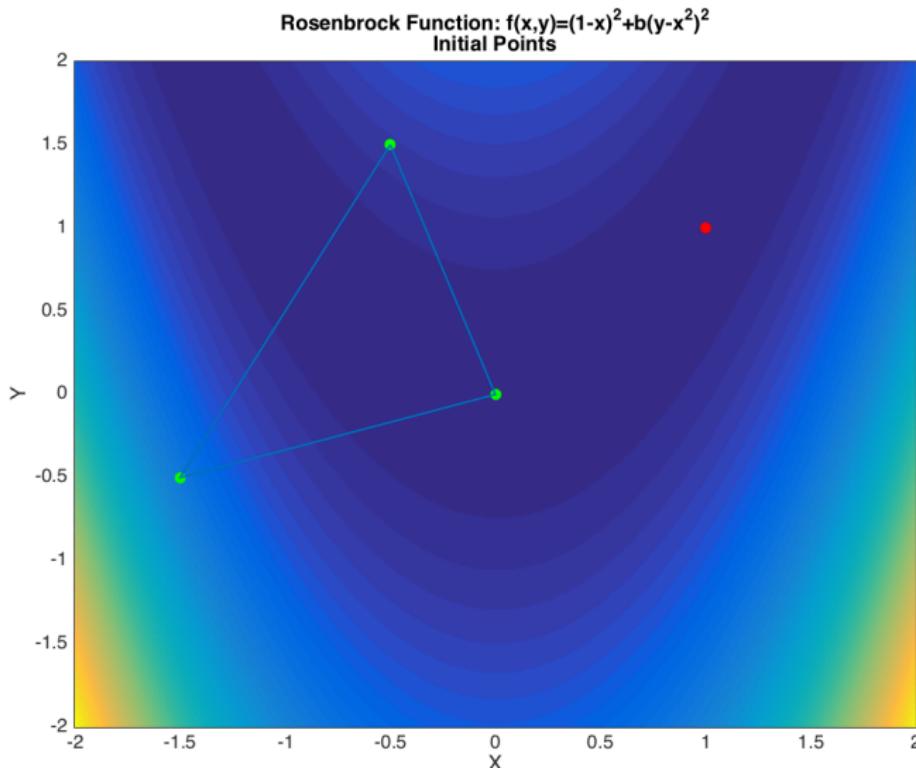
## NELDER MEAD: THE IDEA

- ▶ You have an  $n$ -dimensional problem (for instance, 2-d)
- ▶ Take  $n+1$  vertices in those  $n$  dimensions (for instance, 3 points)
- ▶ Either improve the worst point using all other points or shrink the simplex
  - ▶ “Reflect” or “flip” the worst point over the body of the simplex to look for a better point
  - ▶ If that worked okay, then stop
  - ▶ If that worked very well, move it even further (expansion)
  - ▶ If that improved things, use expanded point, otherwise, use reflected point.
  - ▶ If that didn’t work well, then move point closer to body but don’t flip
  - ▶ If this works, then use it
  - ▶ If this didn’t work, shrink everyone’s distance from the best point
- ▶ Continue until near a minimum/maximum

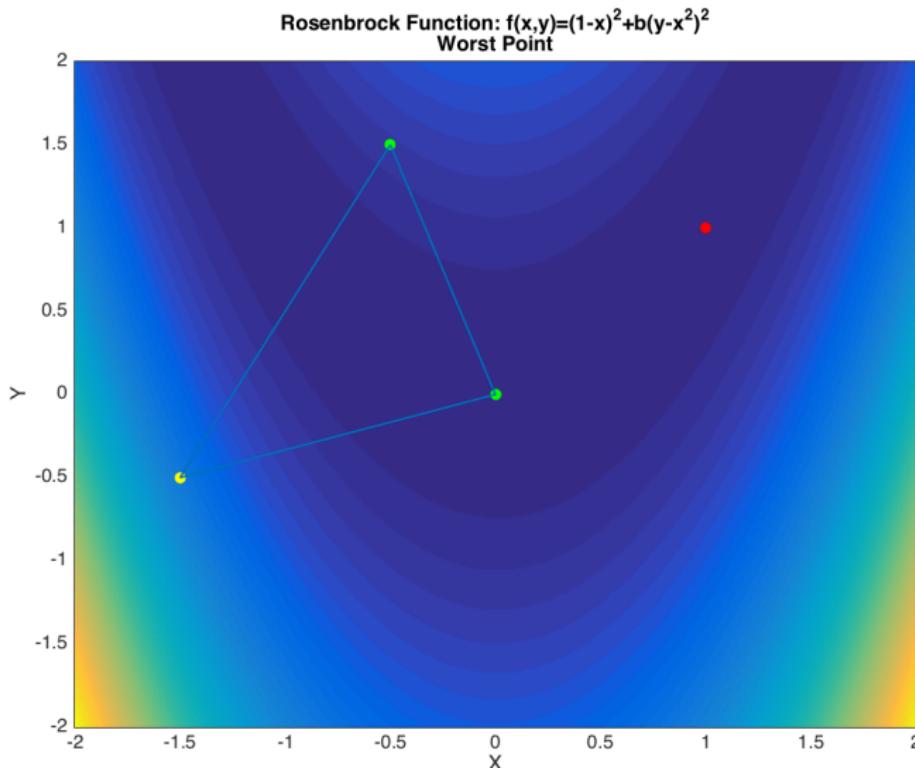
# NELDER MEAD: STEP EXAMPLES



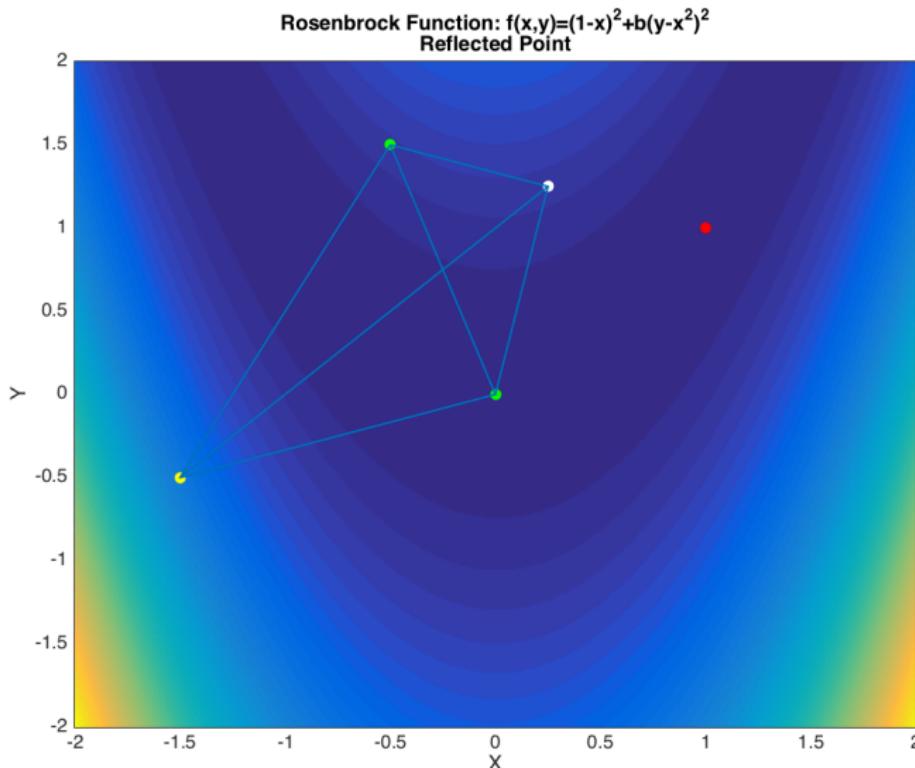
# NELDER MEAD: STEP EXAMPLES



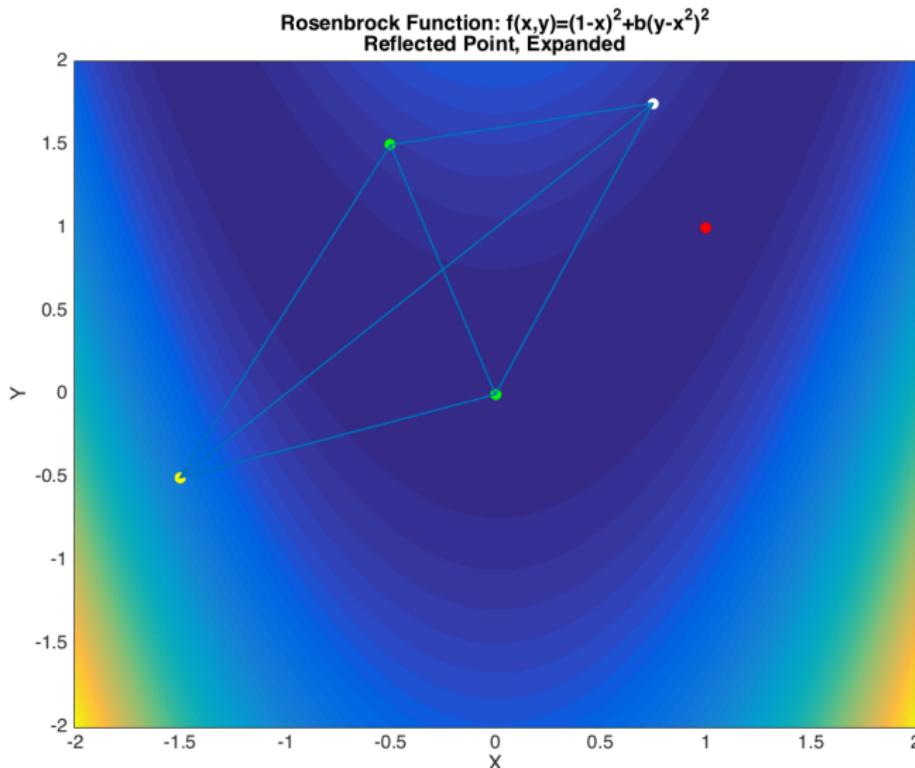
# NELDER MEAD: STEP EXAMPLES



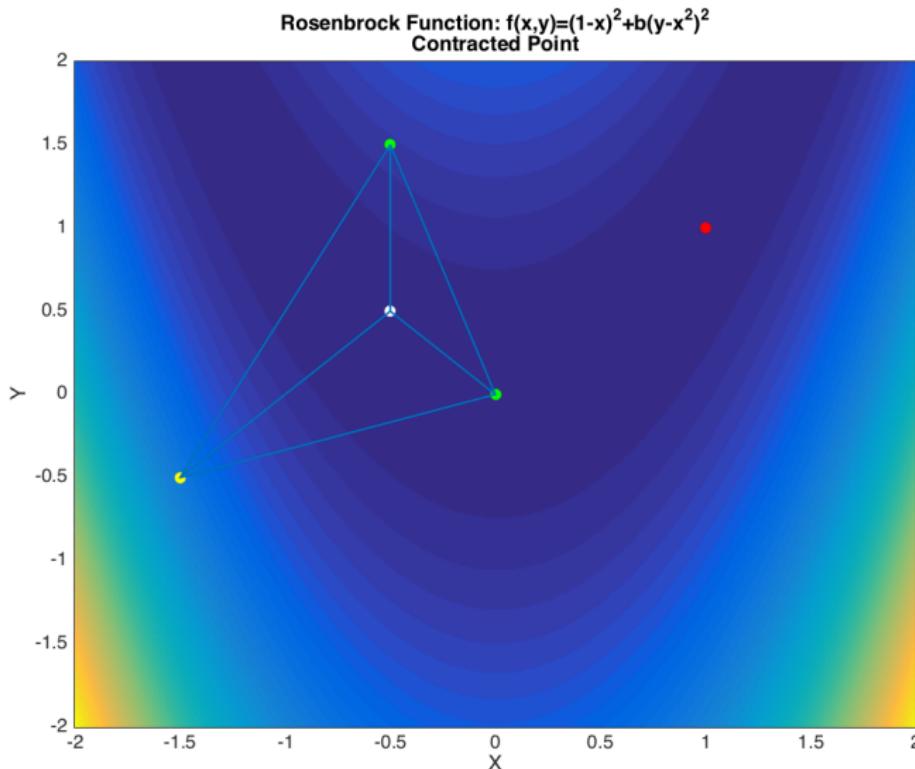
# NELDER MEAD: STEP EXAMPLES



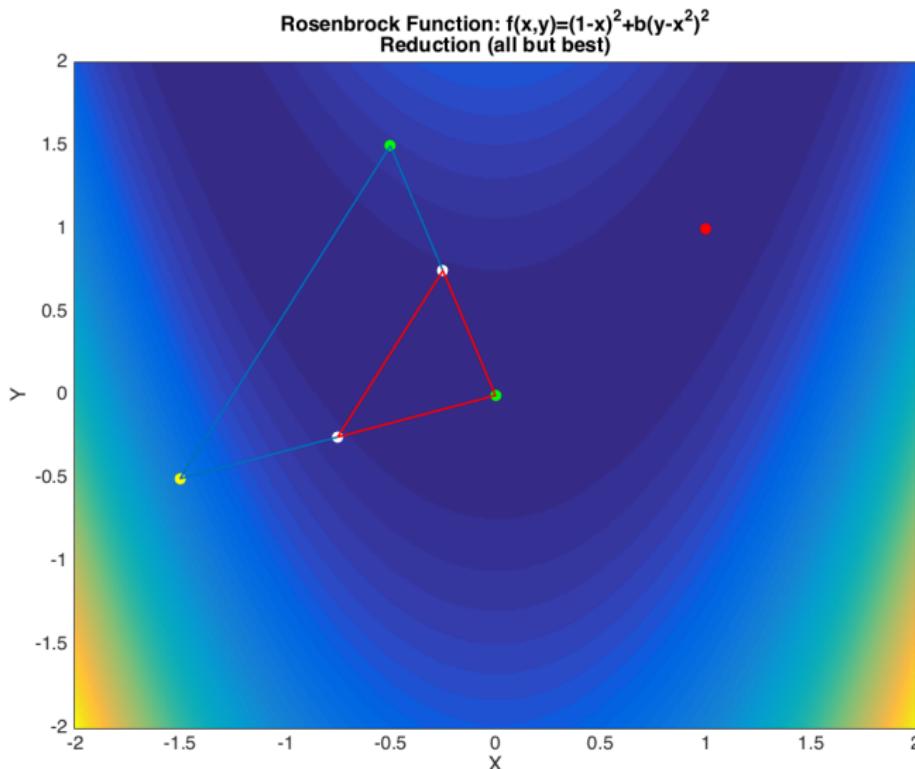
# NELDER MEAD: STEP EXAMPLES



# NELDER MEAD: STEP EXAMPLES



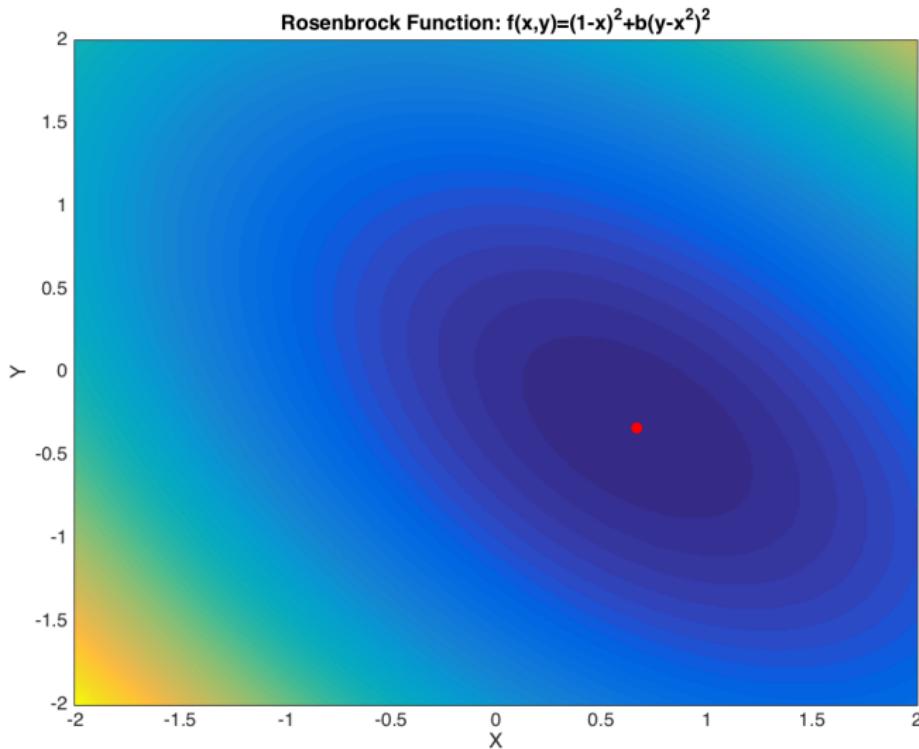
# NELDER MEAD: STEP EXAMPLES



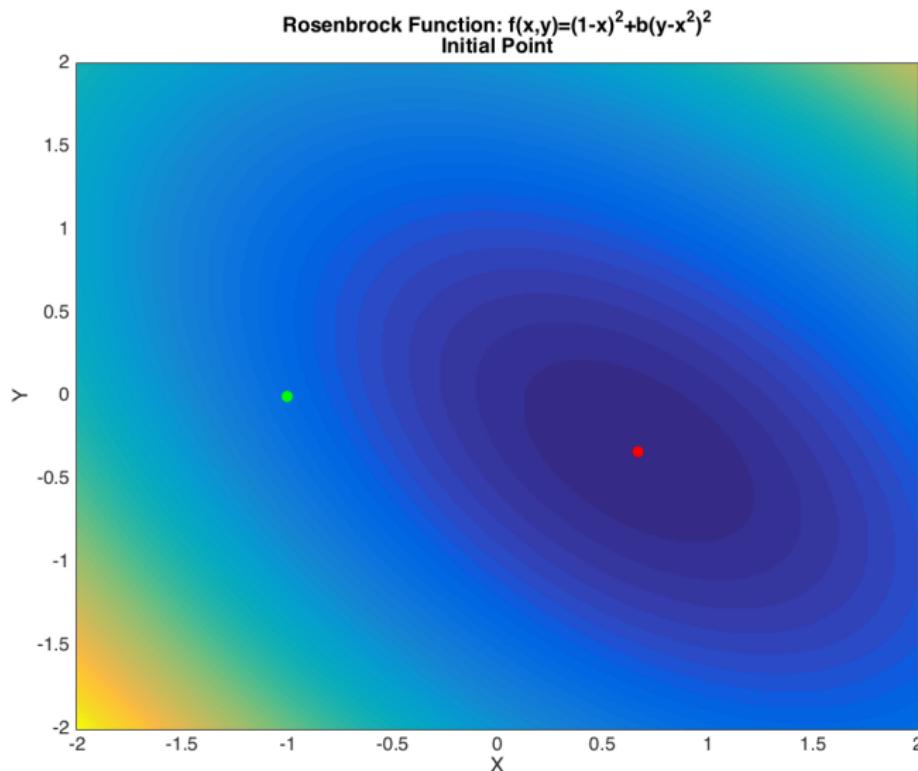
## PATTERN SEARCH

- ▶ Pattern search is what you'd do if you didn't trust the terrain
- ▶ Only thing that competes with steepest descent for most natural
- ▶ Pick a stepsize
- ▶ One at a time, perturb your point in only one direction by the stepsize
- ▶ If things are improved, move there, sample more points
- ▶ If not, shrink the step size
- ▶ There are lots of variations on this theme
  - ▶ Your basis vector doesn't have to be  $x \pm \begin{bmatrix} d \\ 0 \end{bmatrix}$  and  $x \pm \begin{bmatrix} 0 \\ d \end{bmatrix}$
  - ▶ It could be  $x \pm \begin{bmatrix} d \\ 0.5d \end{bmatrix}$  and  $x \pm \begin{bmatrix} -0.5d \\ d \end{bmatrix}$
  - ▶ Could choose random orthogonal vectors
  - ▶ Could immediately take improving step, or could search more, etc.

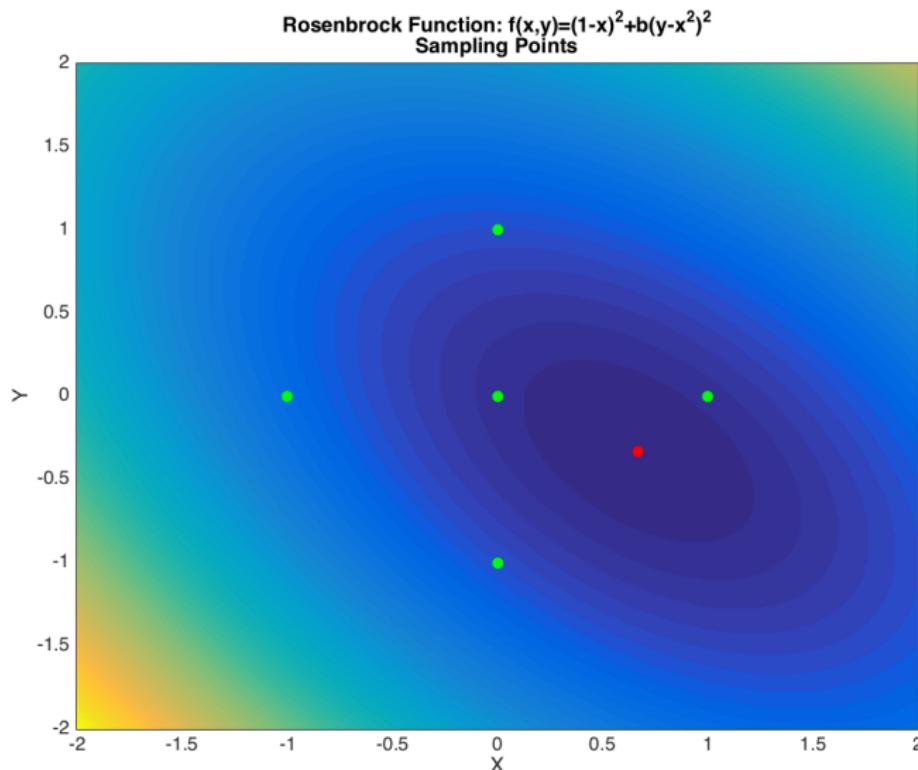
# PATTERN SEARCH: EXAMPLE



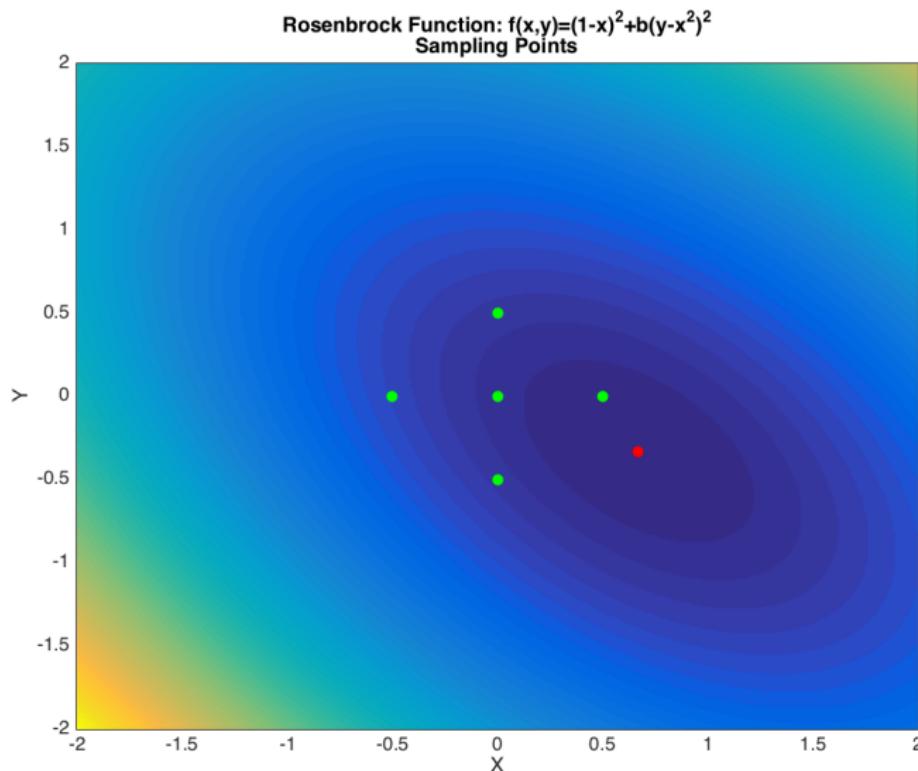
# PATTERN SEARCH: EXAMPLE



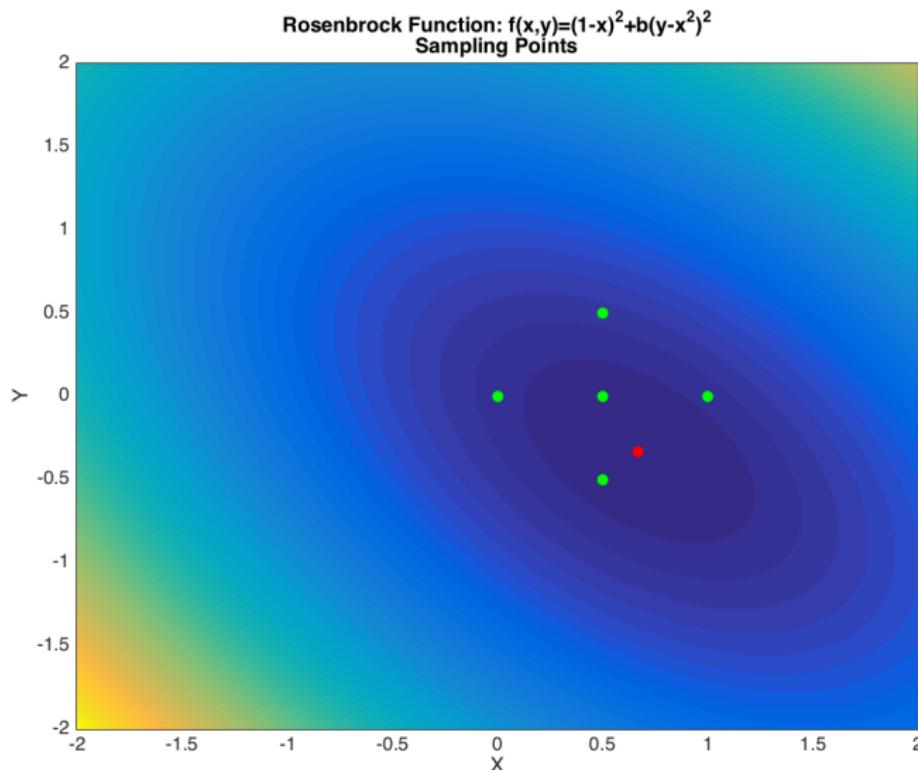
# PATTERN SEARCH: EXAMPLE



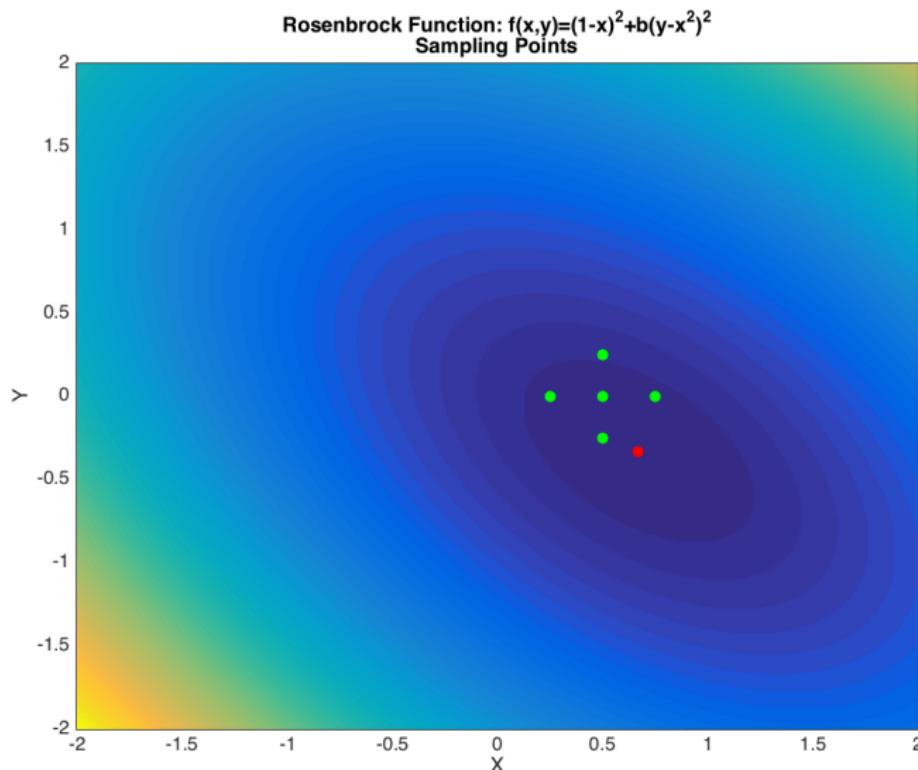
# PATTERN SEARCH: EXAMPLE



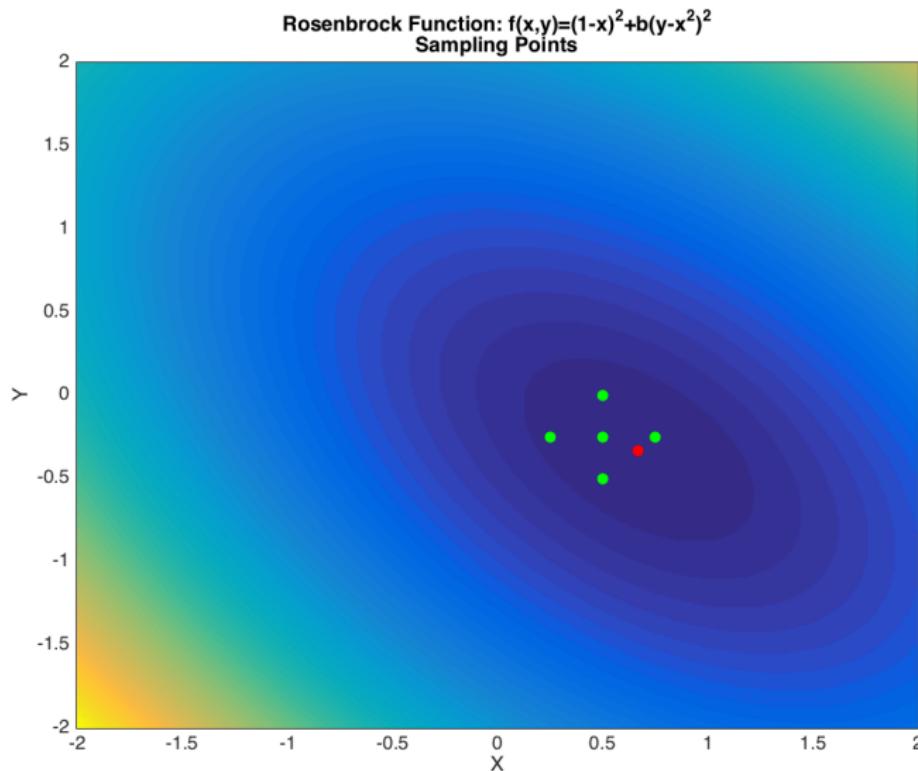
# PATTERN SEARCH: EXAMPLE



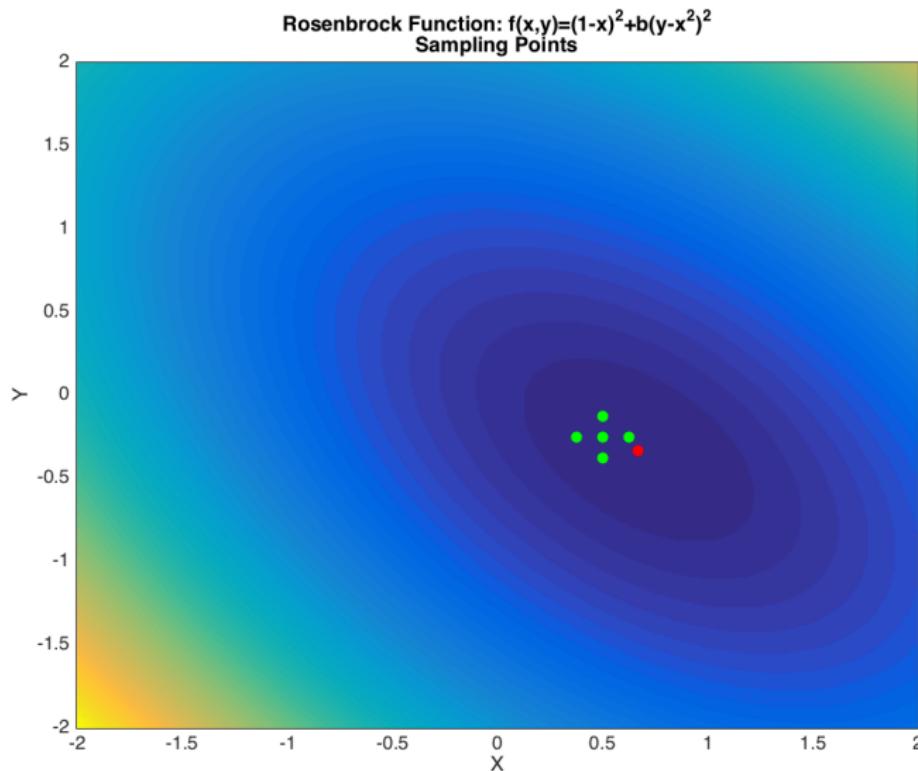
# PATTERN SEARCH: EXAMPLE



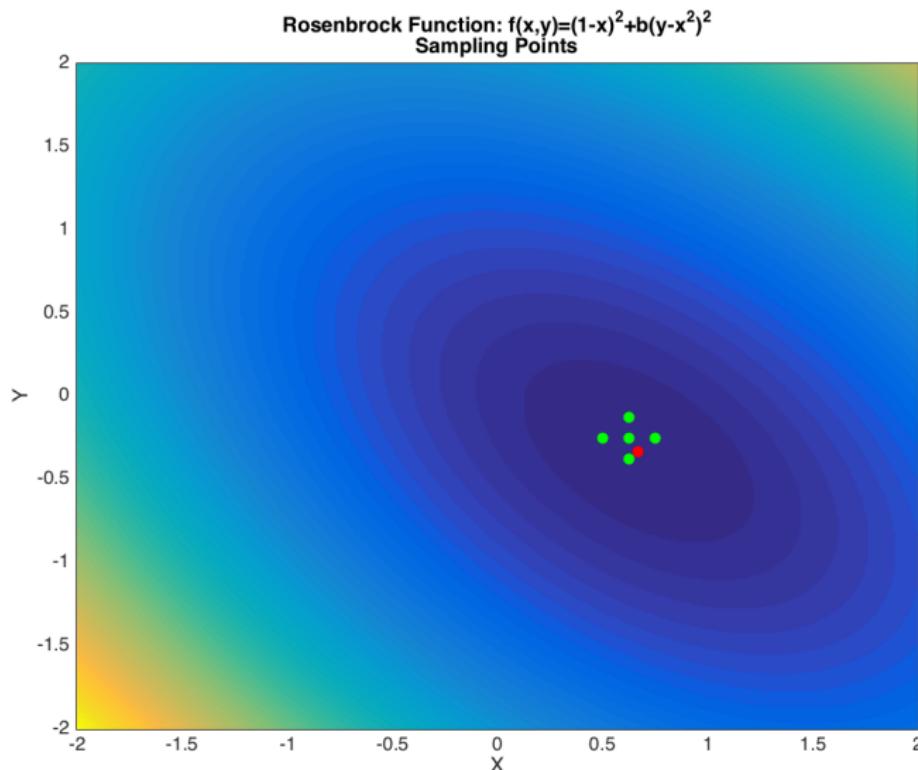
# PATTERN SEARCH: EXAMPLE



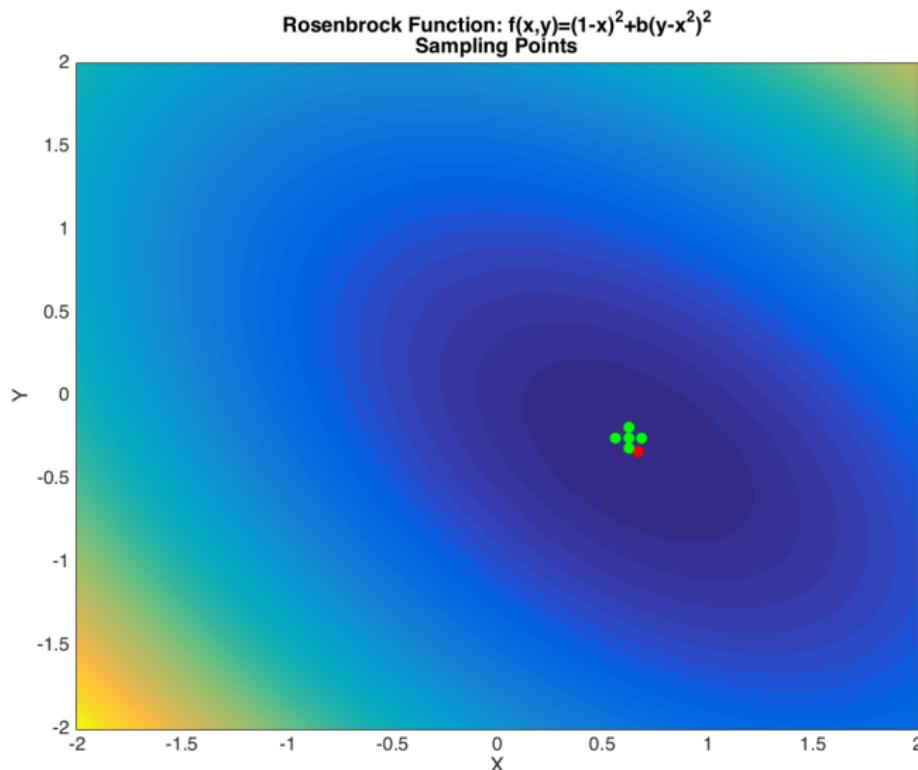
# PATTERN SEARCH: EXAMPLE



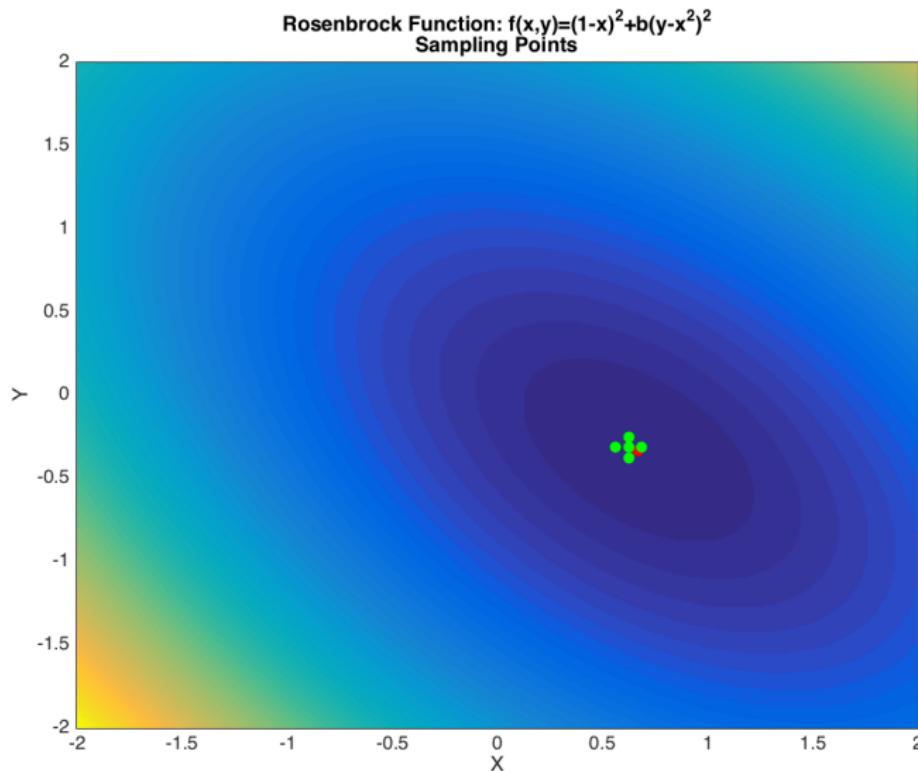
# PATTERN SEARCH: EXAMPLE



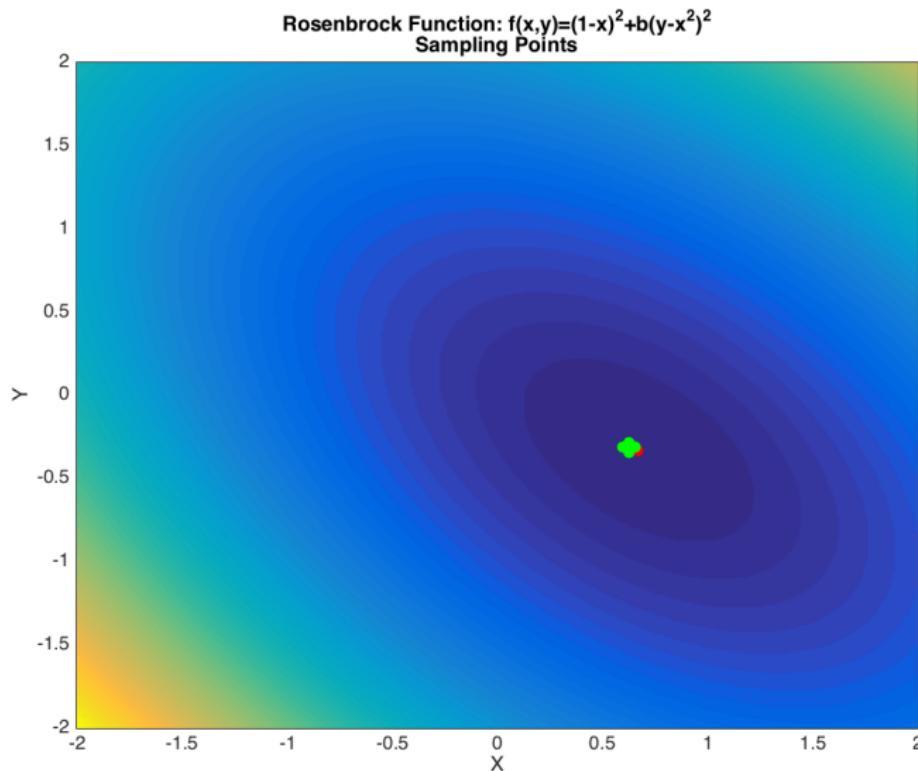
# PATTERN SEARCH: EXAMPLE



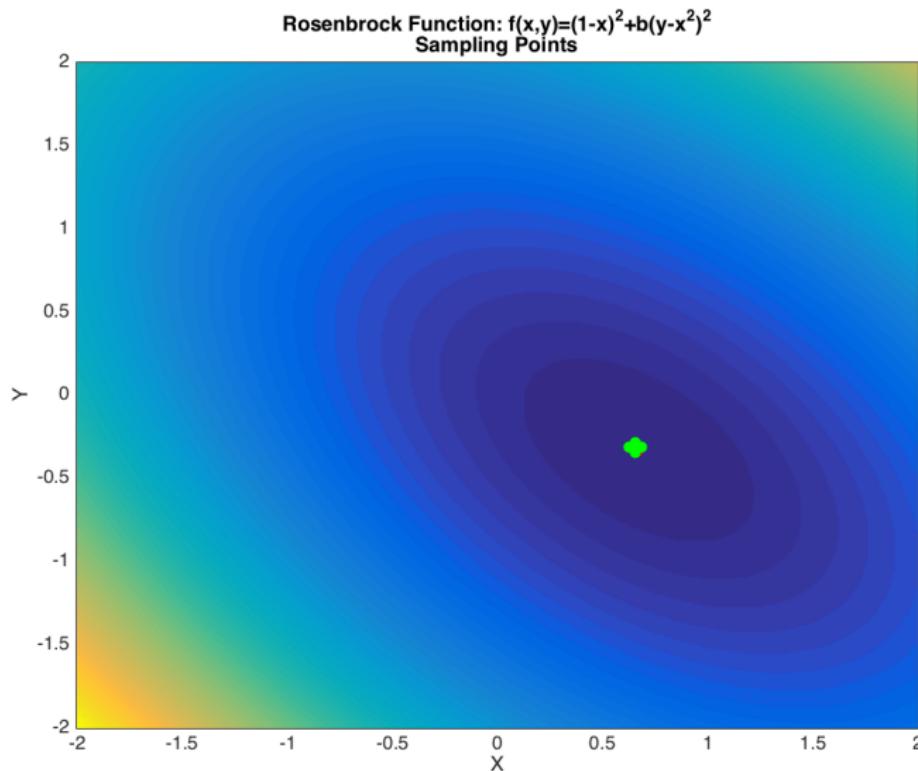
# PATTERN SEARCH: EXAMPLE



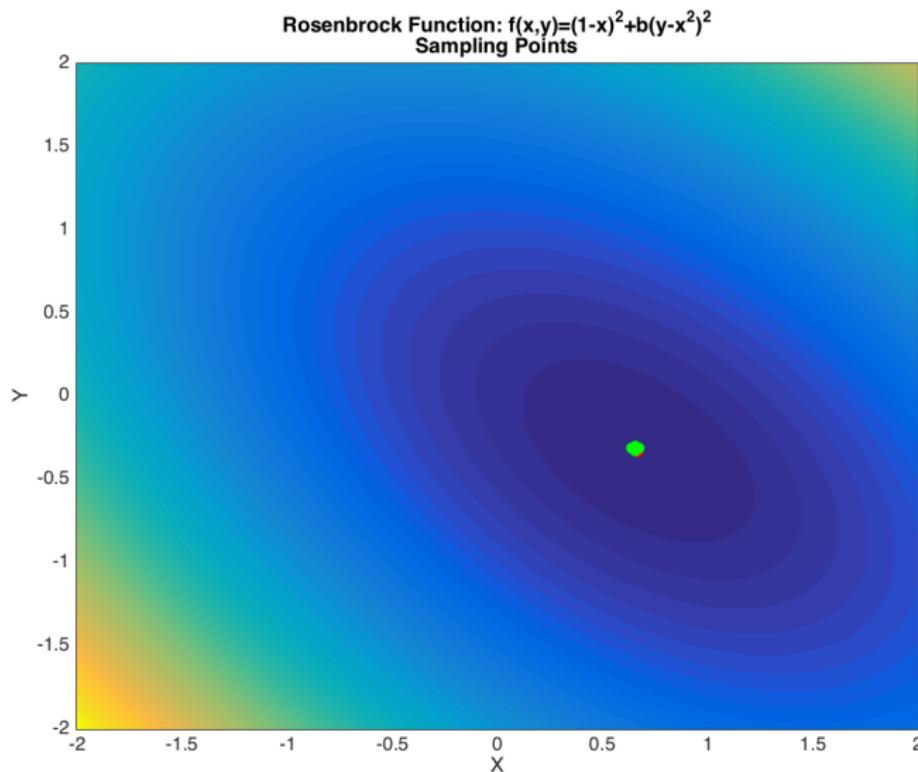
# PATTERN SEARCH: EXAMPLE



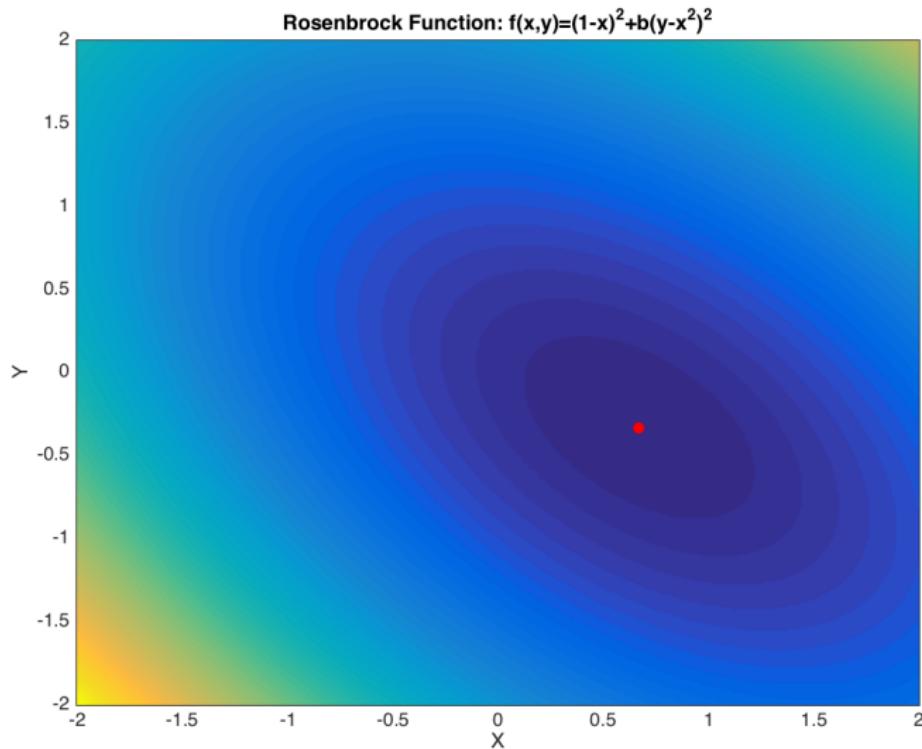
# PATTERN SEARCH: EXAMPLE



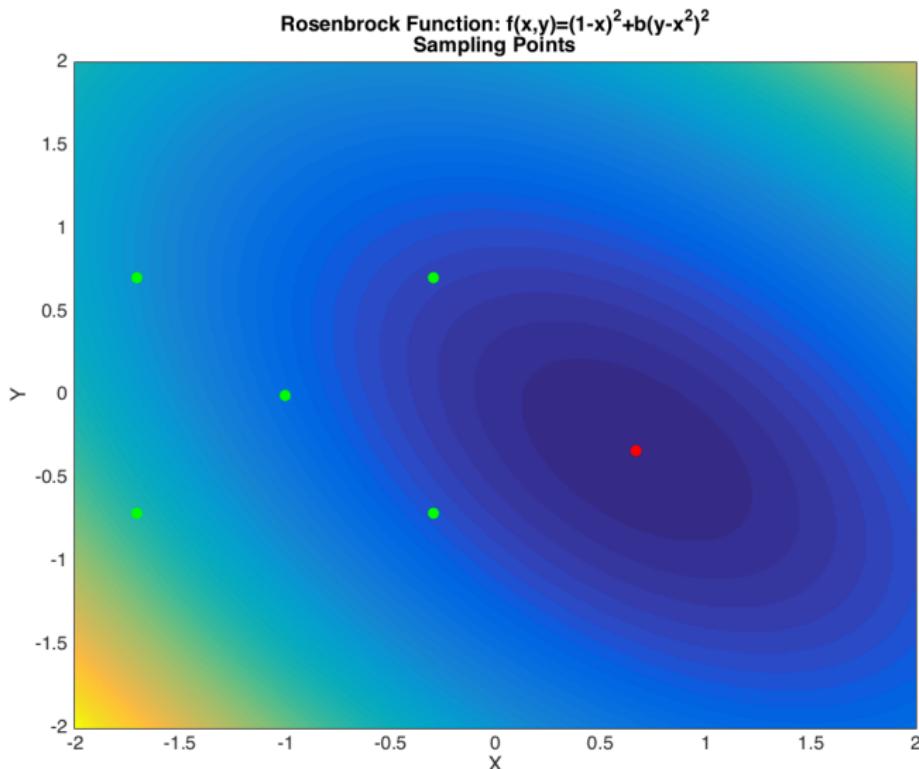
# PATTERN SEARCH: EXAMPLE



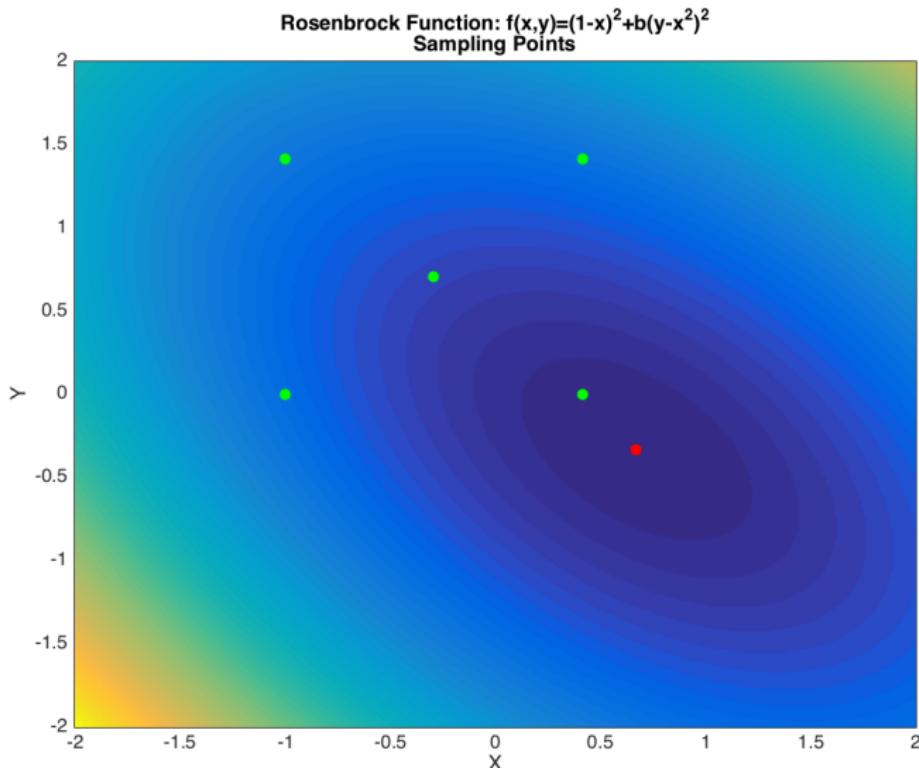
# PATTERN SEARCH: NON-CARTESIAN BASIS



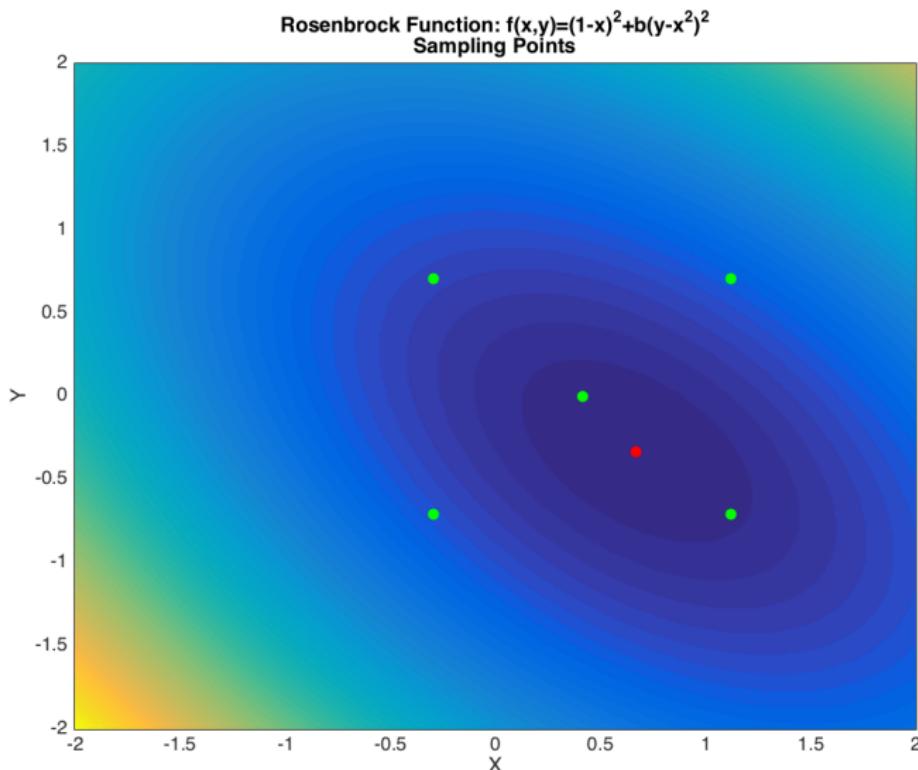
# PATTERN-SEARCH: NON-CARTESIAN BASIS



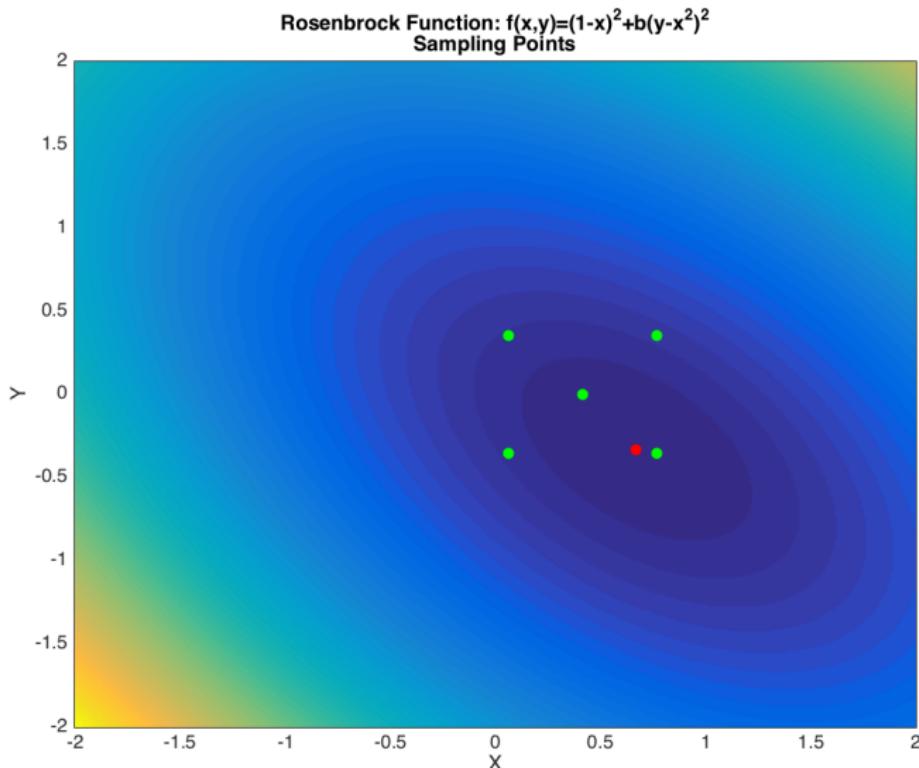
# PATTERN-SEARCH: NON-CARTESIAN BASIS



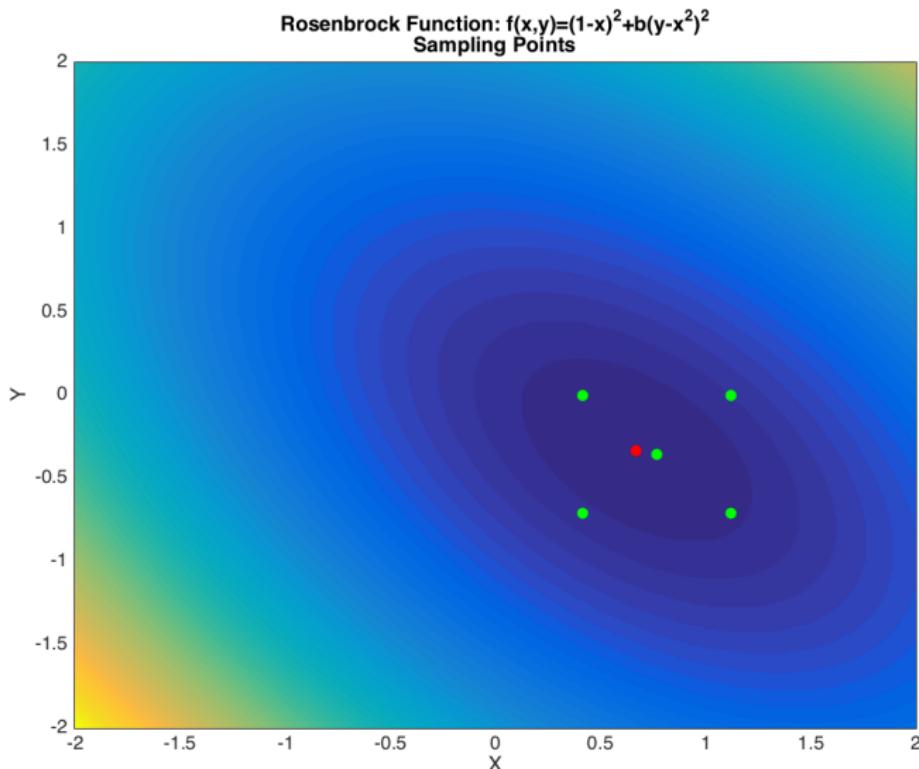
# PATTERN-SEARCH: NON-CARTESIAN BASIS



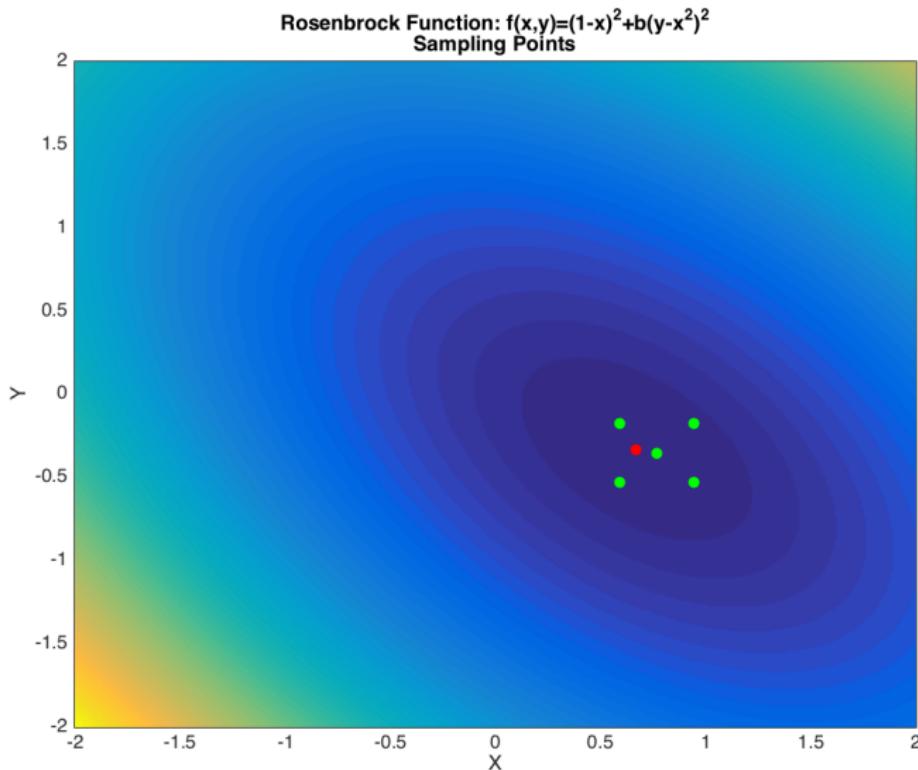
# PATTERN-SEARCH: NON-CARTESIAN BASIS



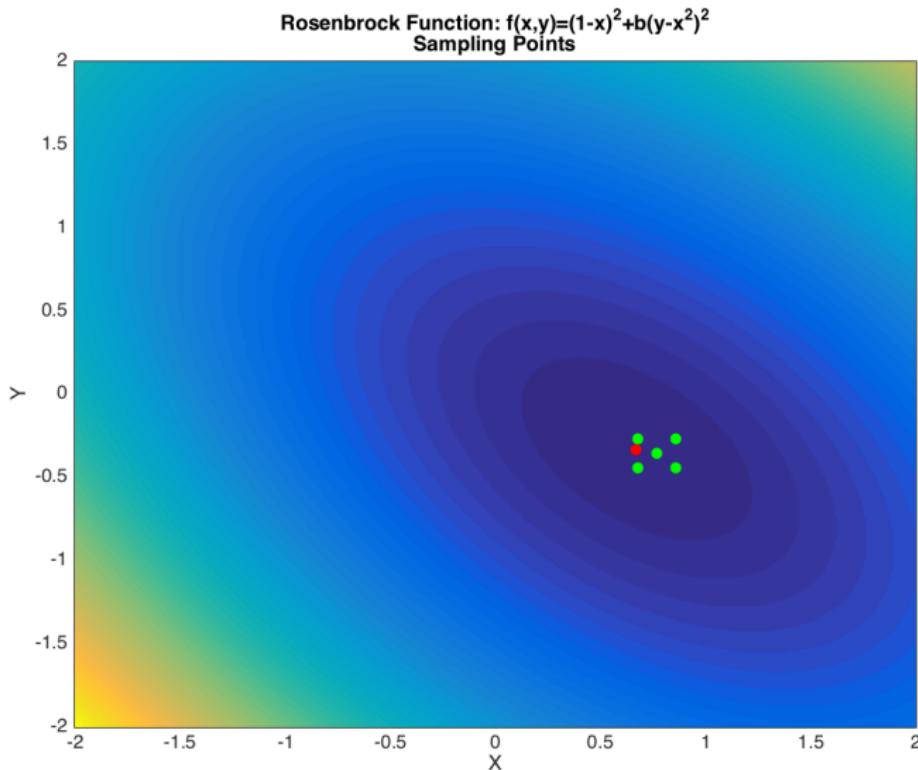
# PATTERN-SEARCH: NON-CARTESIAN BASIS



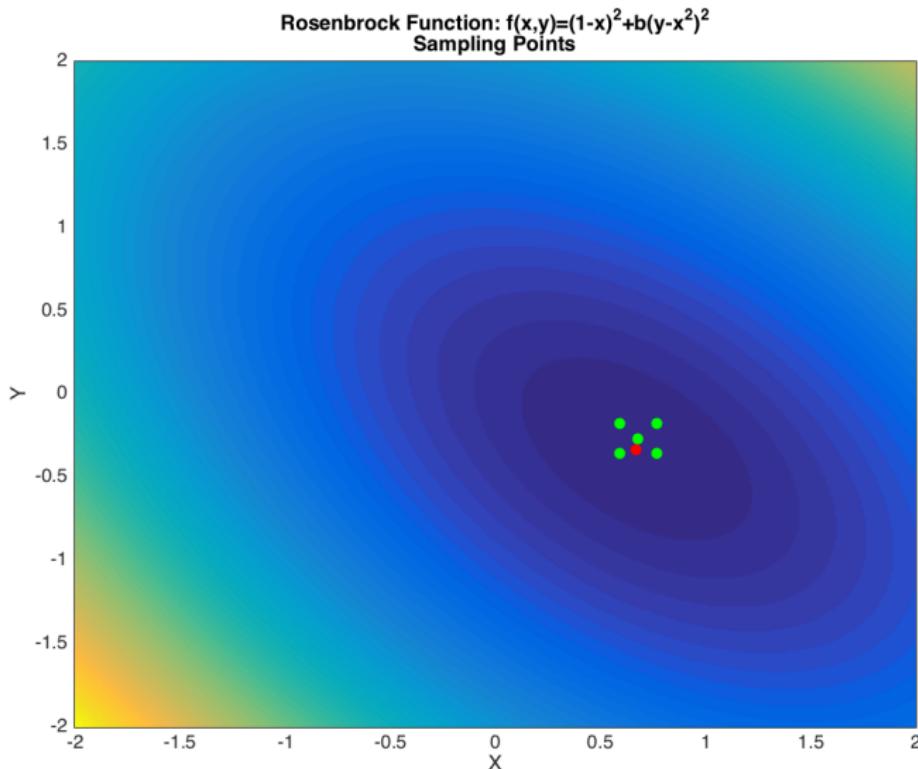
# PATTERN-SEARCH: NON-CARTESIAN BASIS



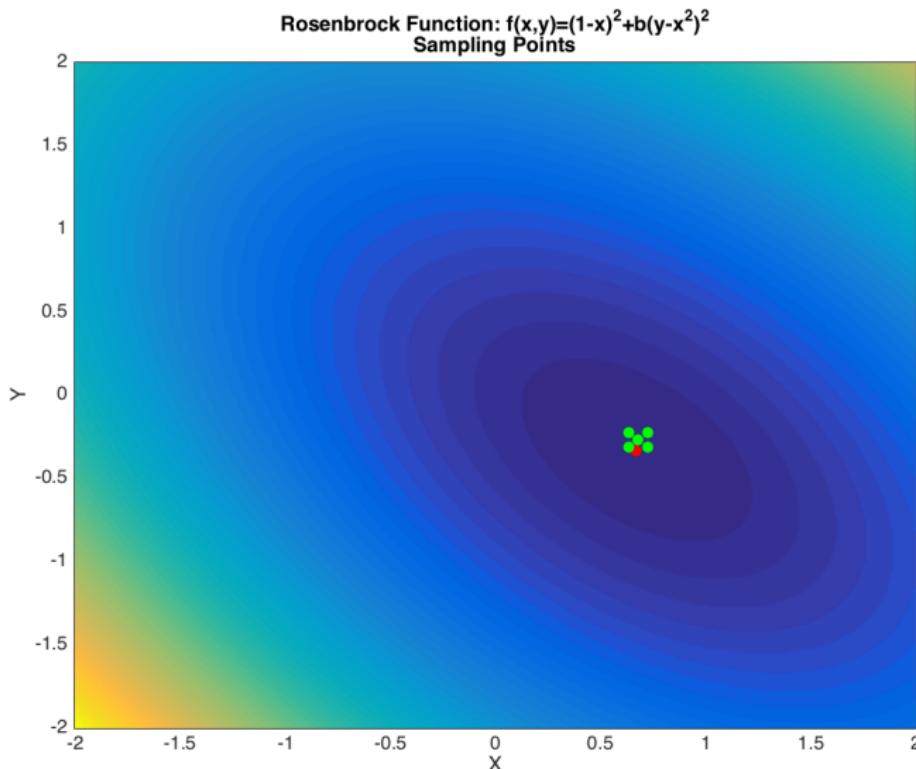
# PATTERN-SEARCH: NON-CARTESIAN BASIS



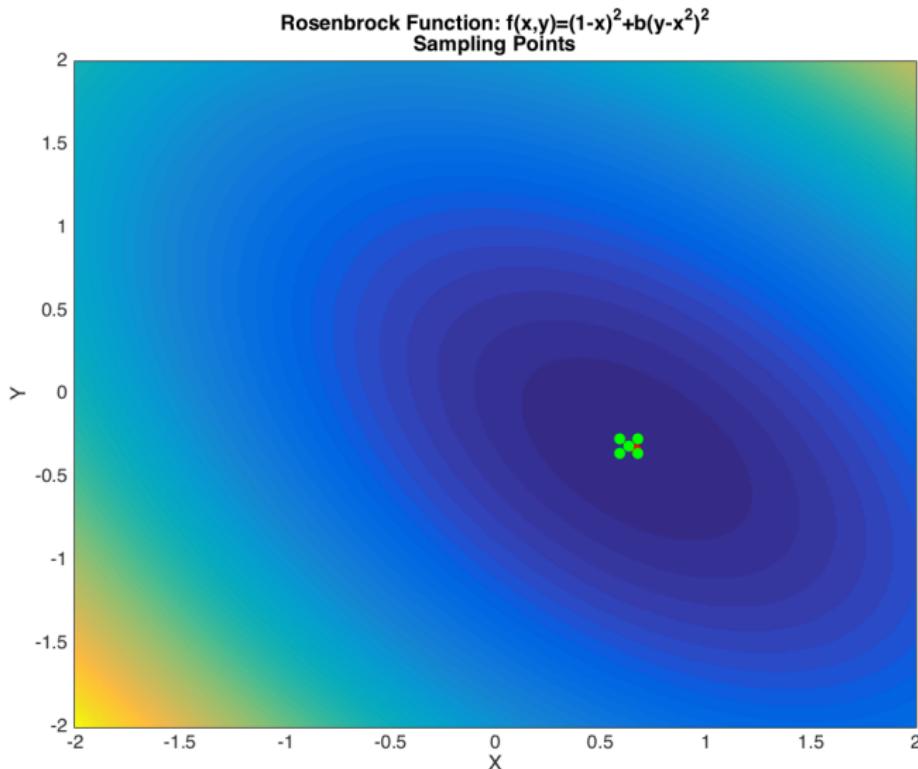
# PATTERN-SEARCH: NON-CARTESIAN BASIS



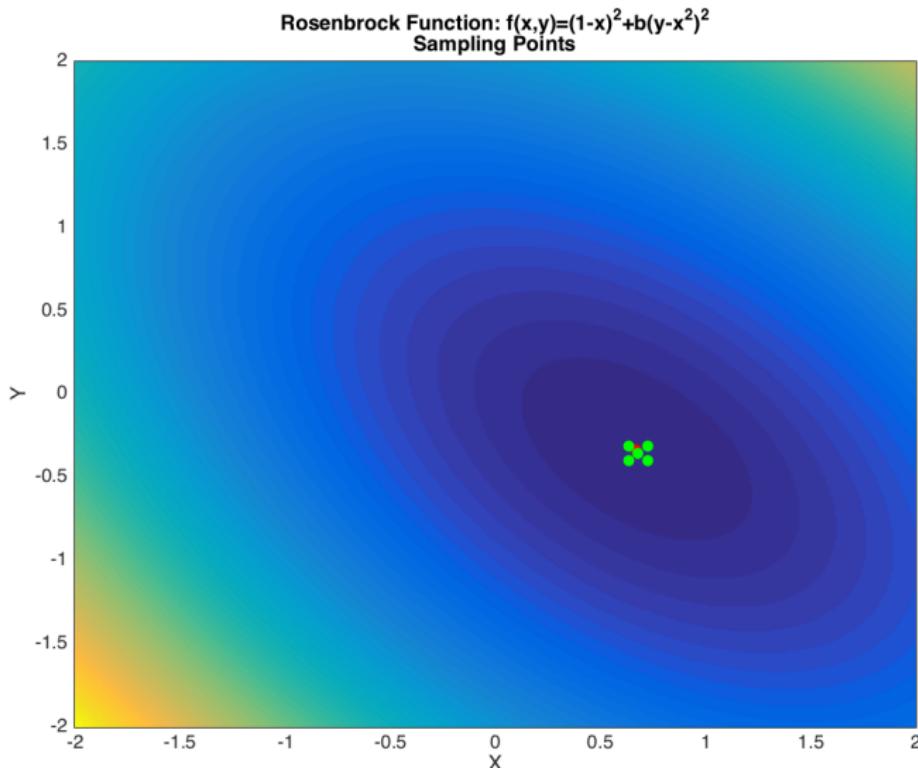
# PATTERN-SEARCH: NON-CARTESIAN BASIS



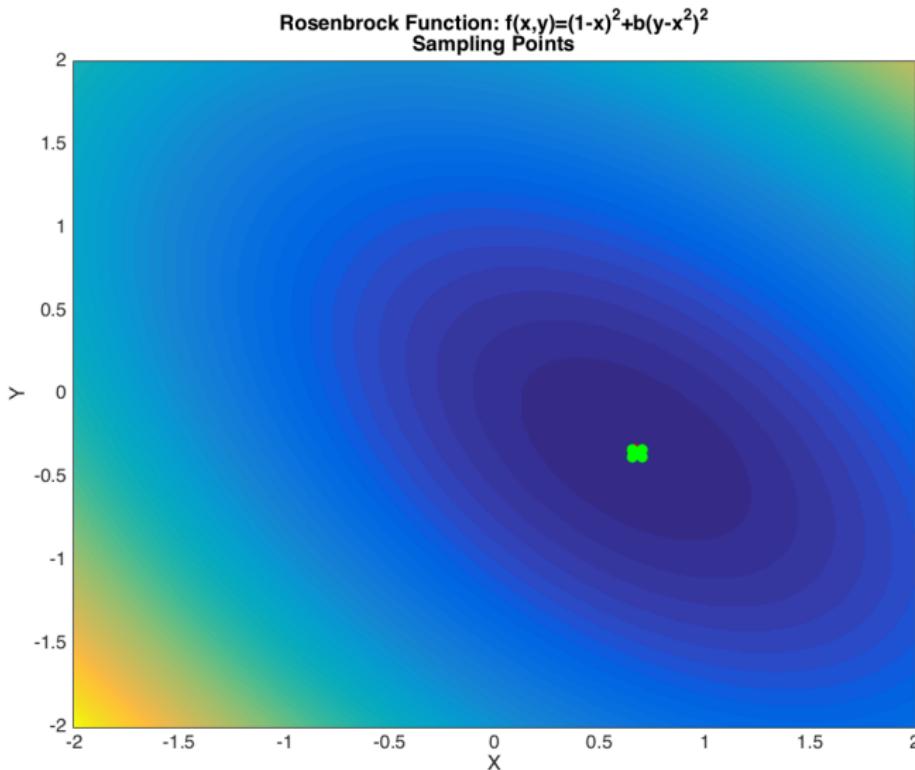
# PATTERN-SEARCH: NON-CARTESIAN BASIS



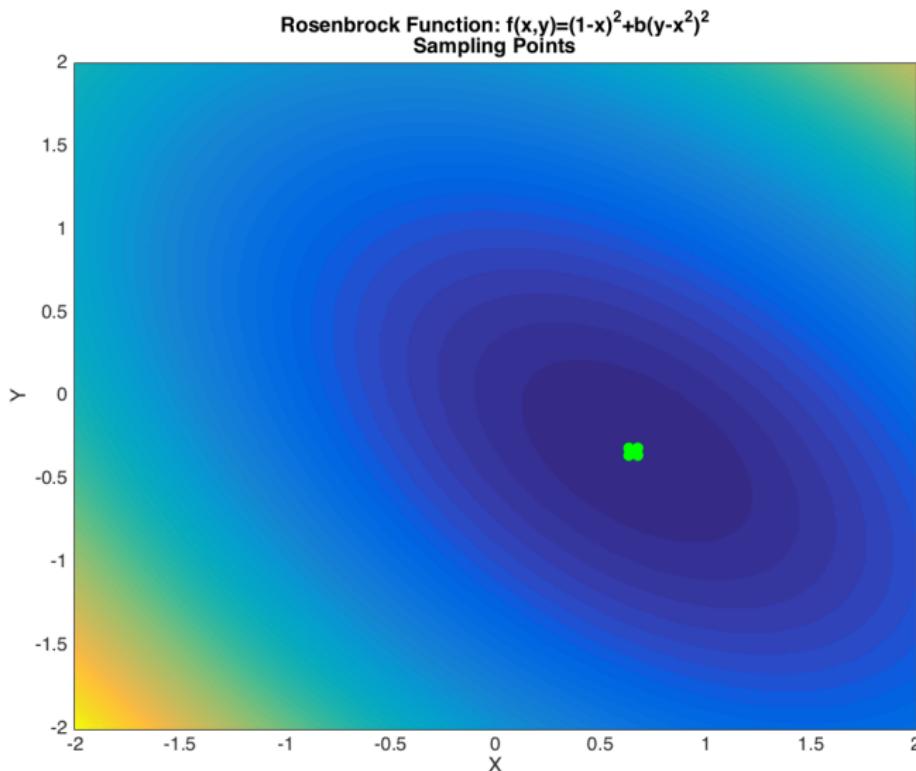
# PATTERN-SEARCH: NON-CARTESIAN BASIS



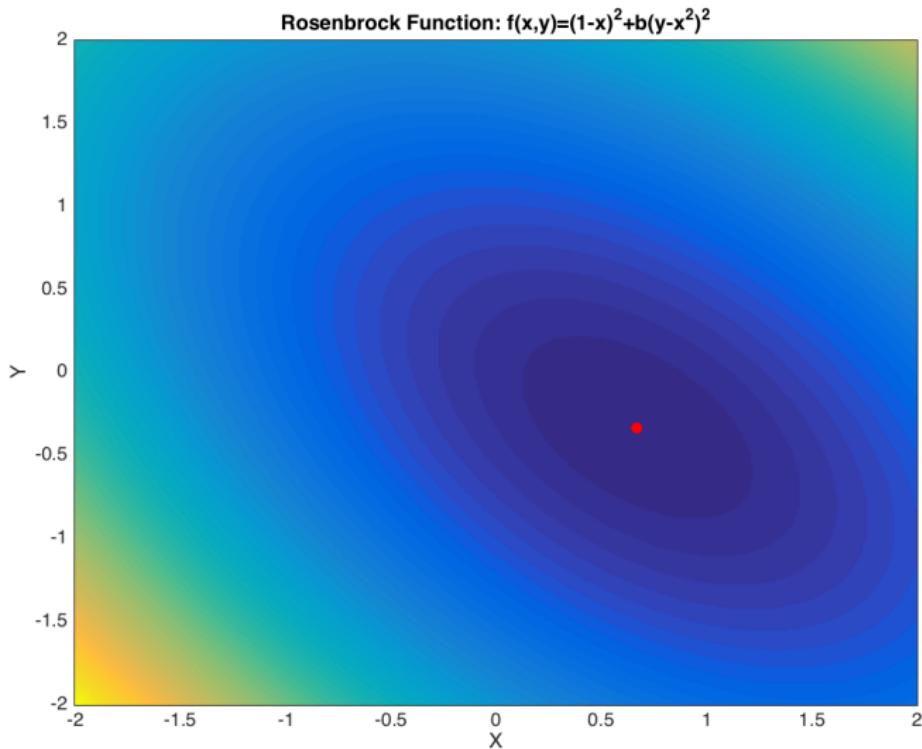
# PATTERN-SEARCH: NON-CARTESIAN BASIS



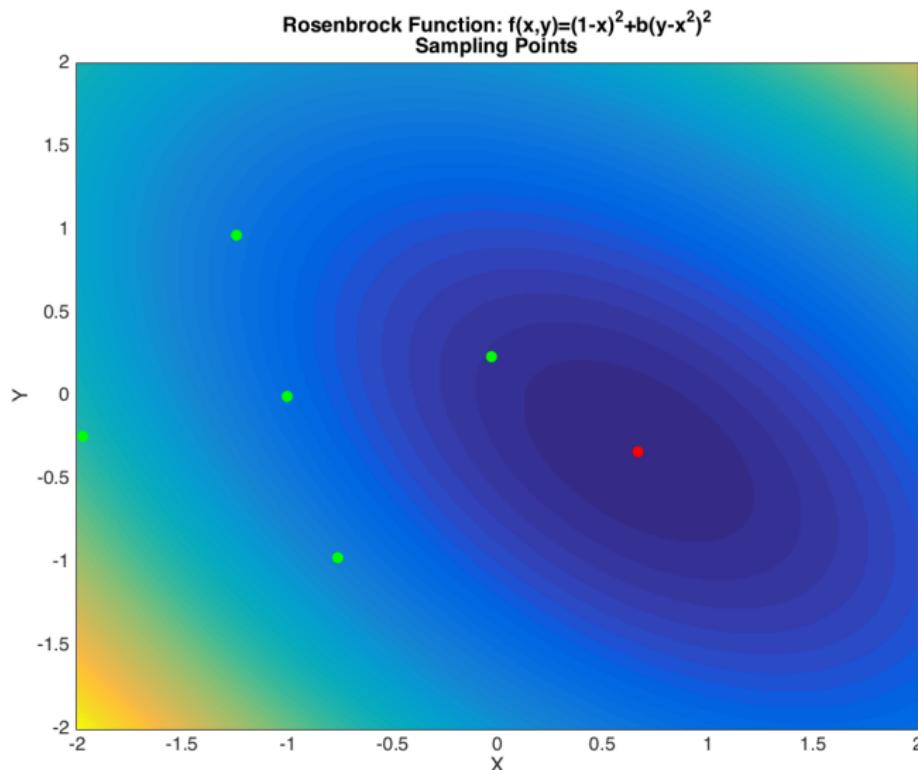
# PATTERN-SEARCH: NON-CARTESIAN BASIS



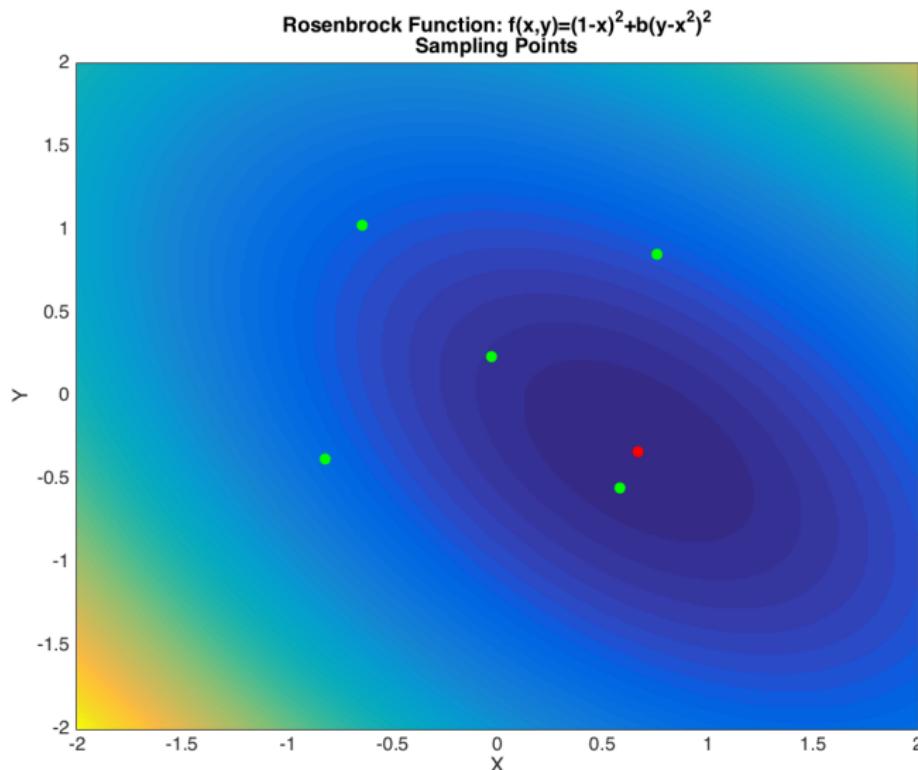
# PATTERN SEARCH: RANDOM BASIS



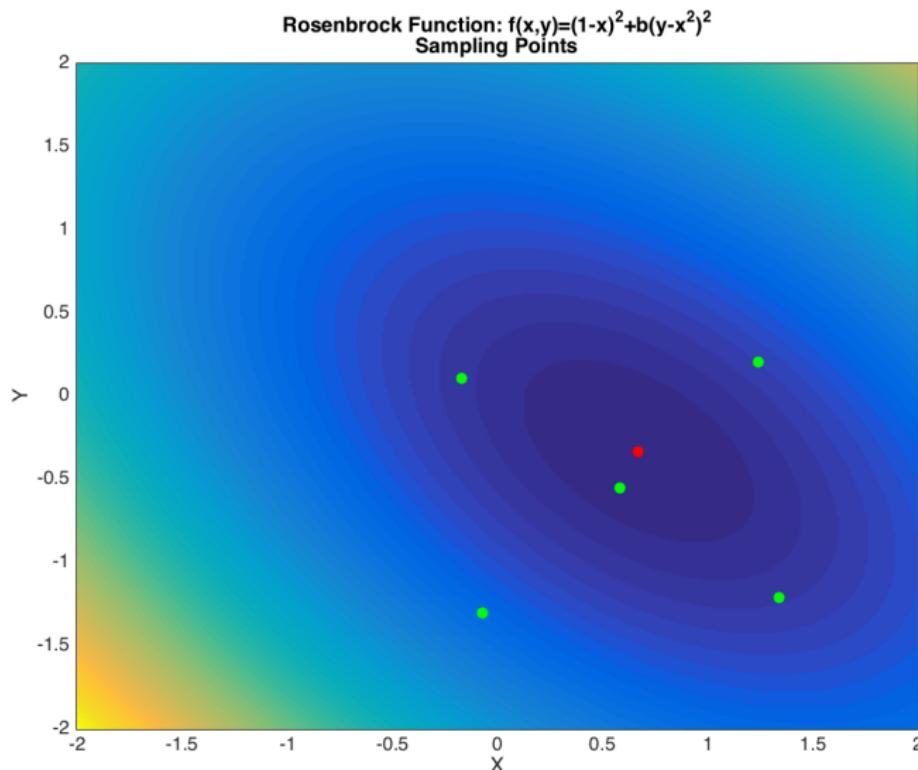
# PATTERN SEARCH: RANDOM BASIS



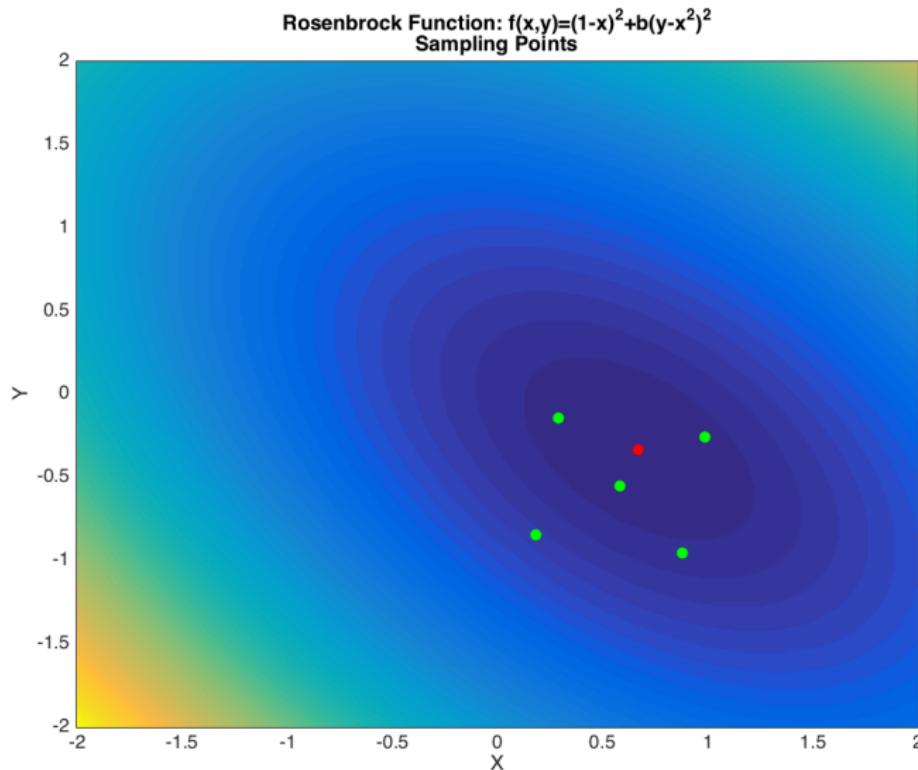
# PATTERN SEARCH: RANDOM BASIS



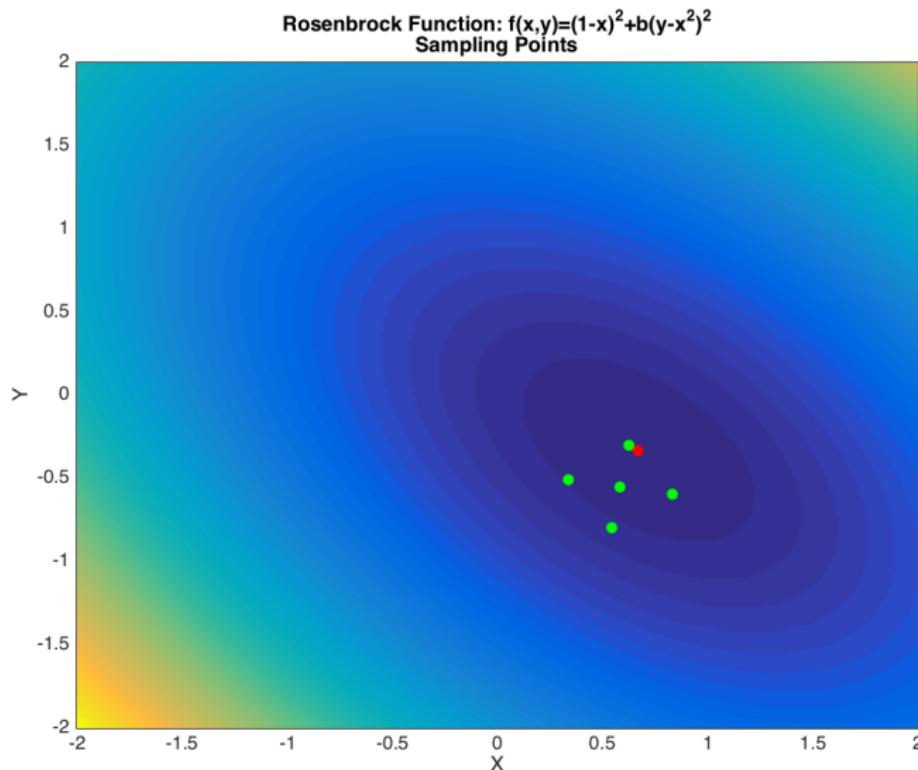
# PATTERN SEARCH: RANDOM BASIS



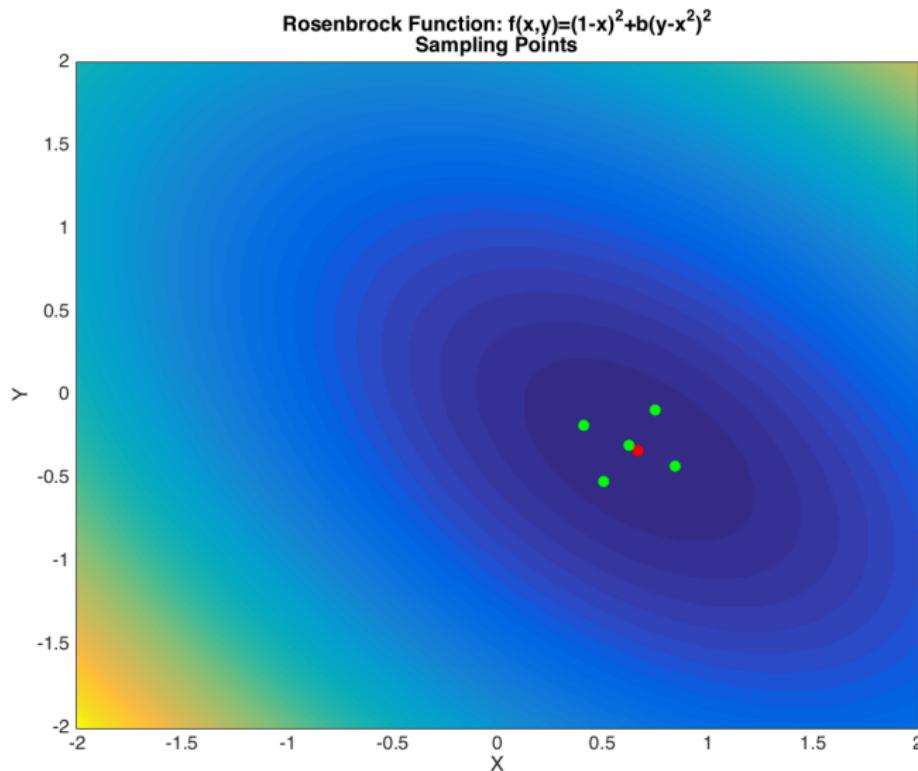
# PATTERN SEARCH: RANDOM BASIS



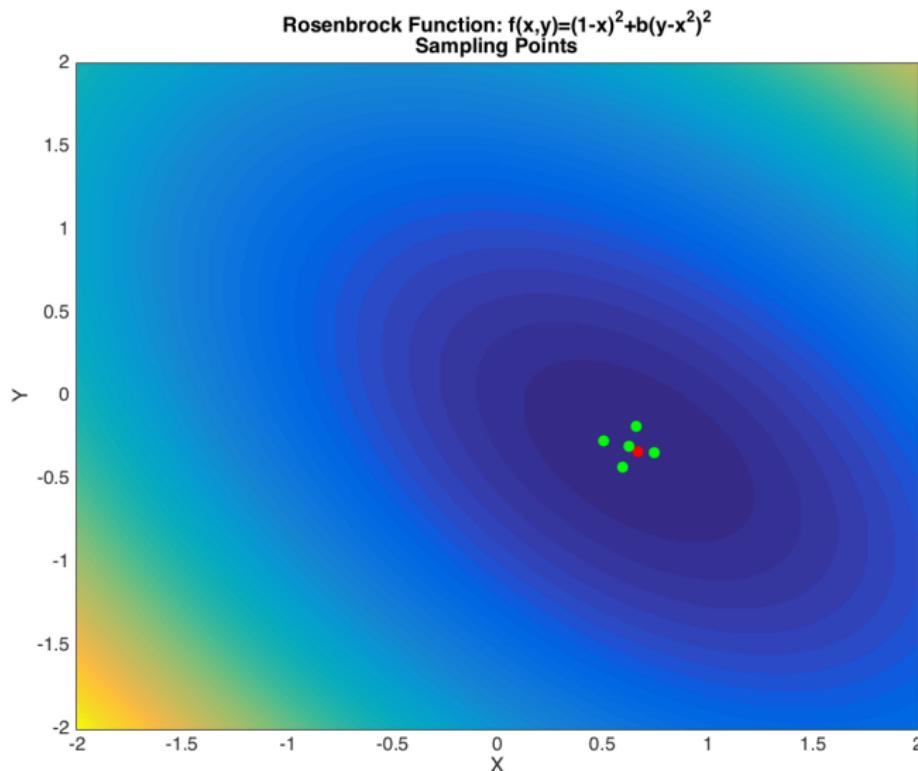
# PATTERN SEARCH: RANDOM BASIS



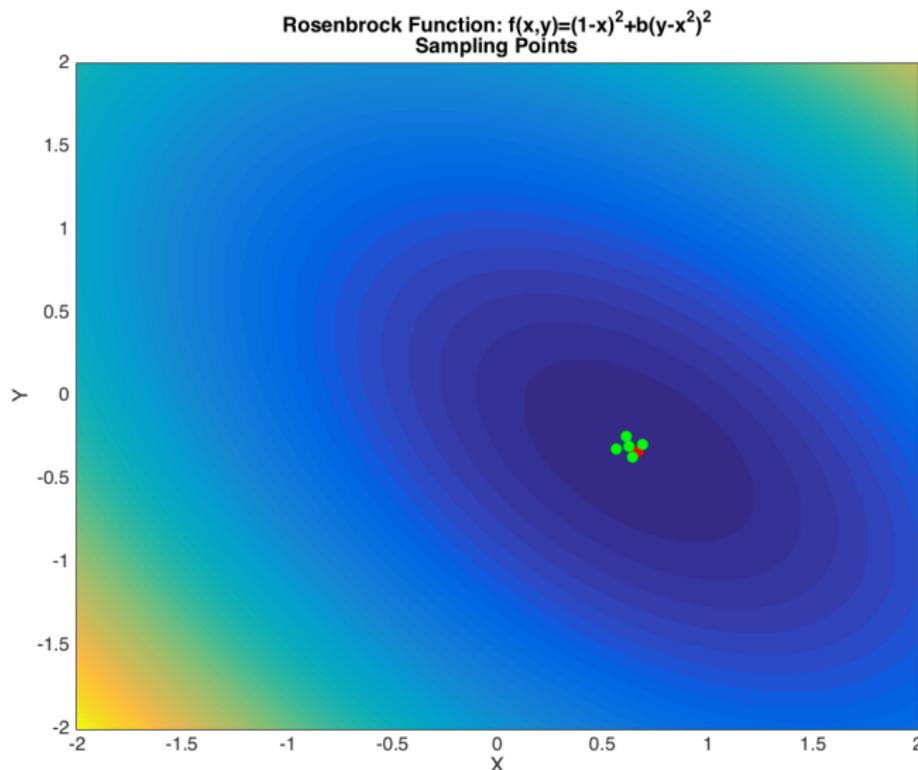
# PATTERN SEARCH: RANDOM BASIS



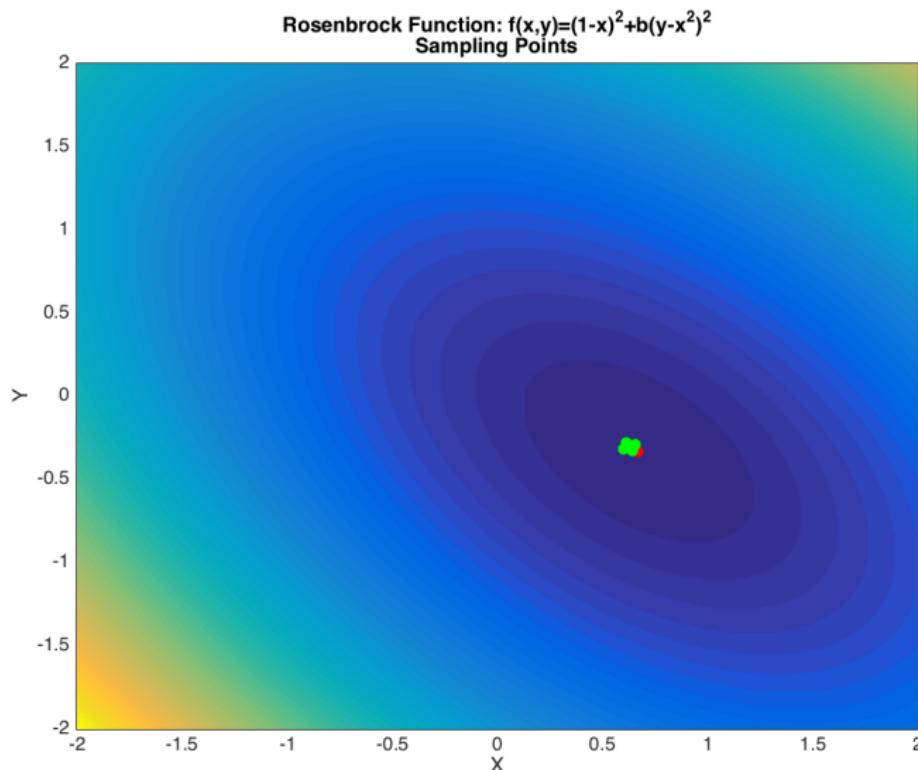
# PATTERN SEARCH: RANDOM BASIS



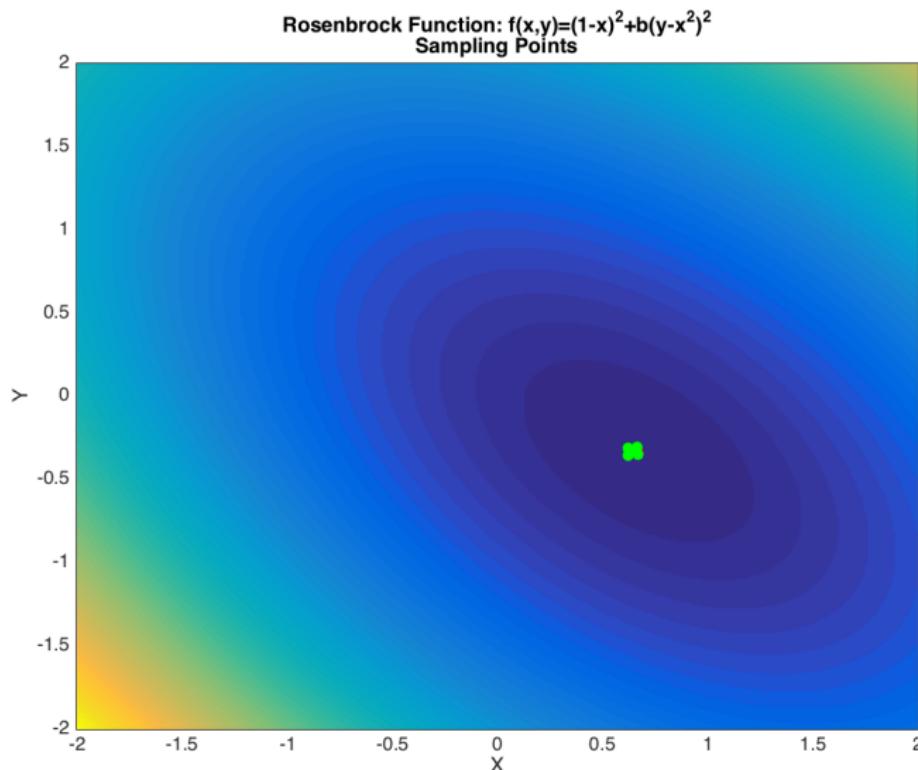
# PATTERN SEARCH: RANDOM BASIS



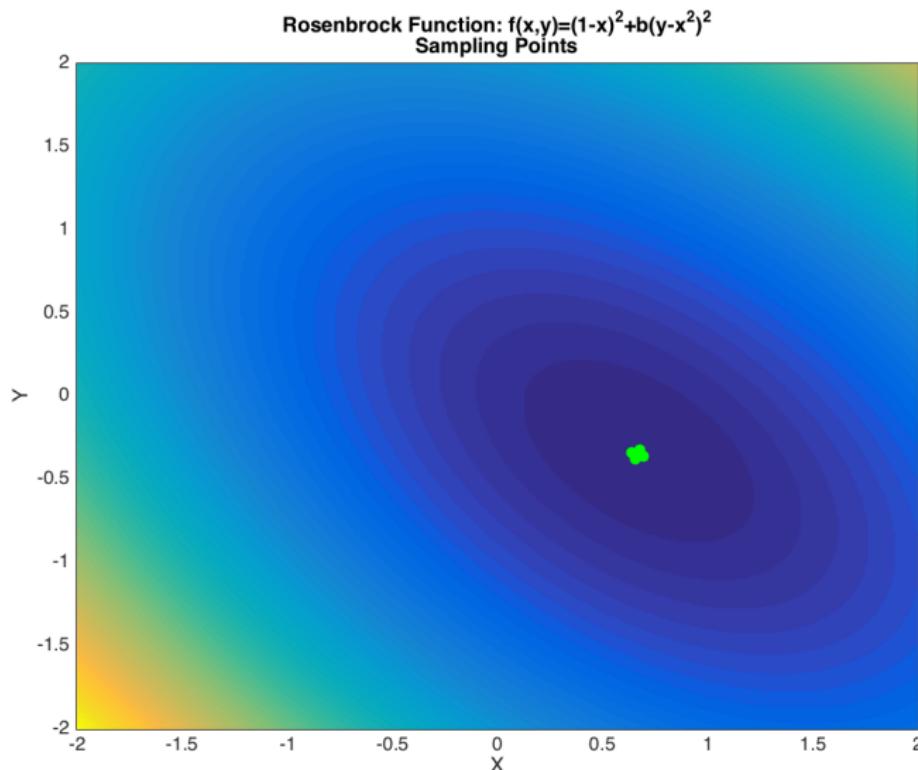
# PATTERN SEARCH: RANDOM BASIS



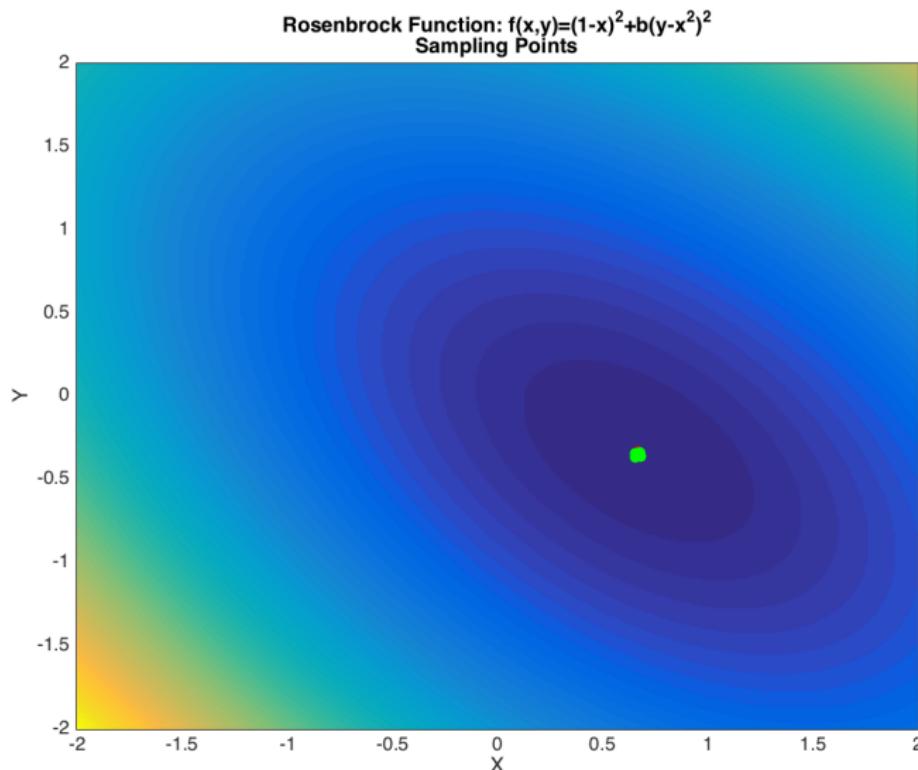
# PATTERN SEARCH: RANDOM BASIS



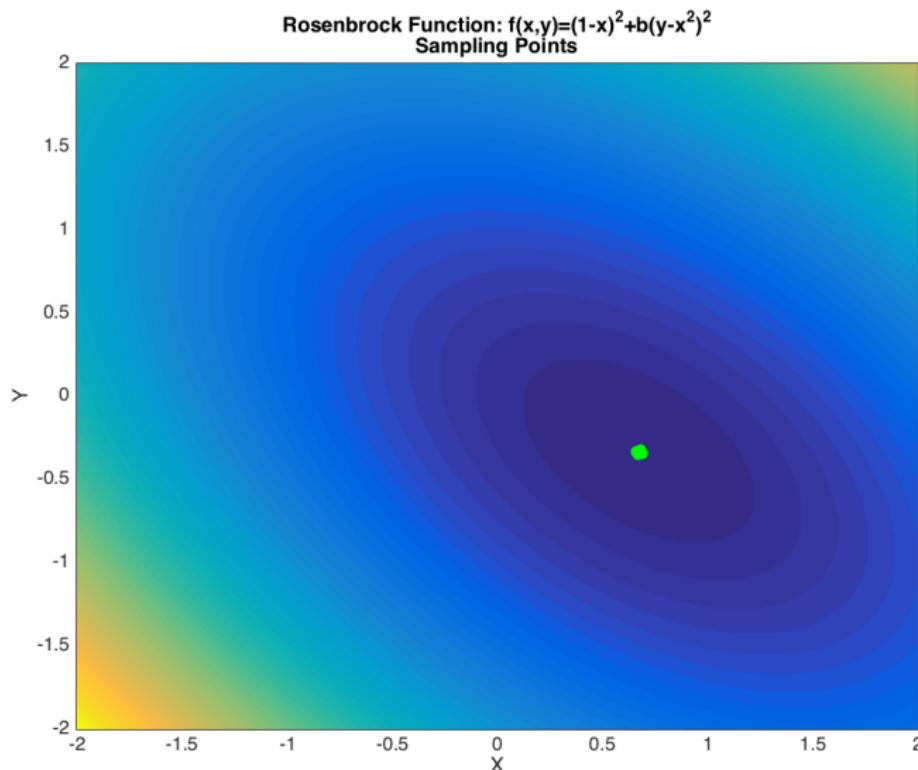
# PATTERN SEARCH: RANDOM BASIS



# PATTERN SEARCH: RANDOM BASIS



# PATTERN SEARCH: RANDOM BASIS



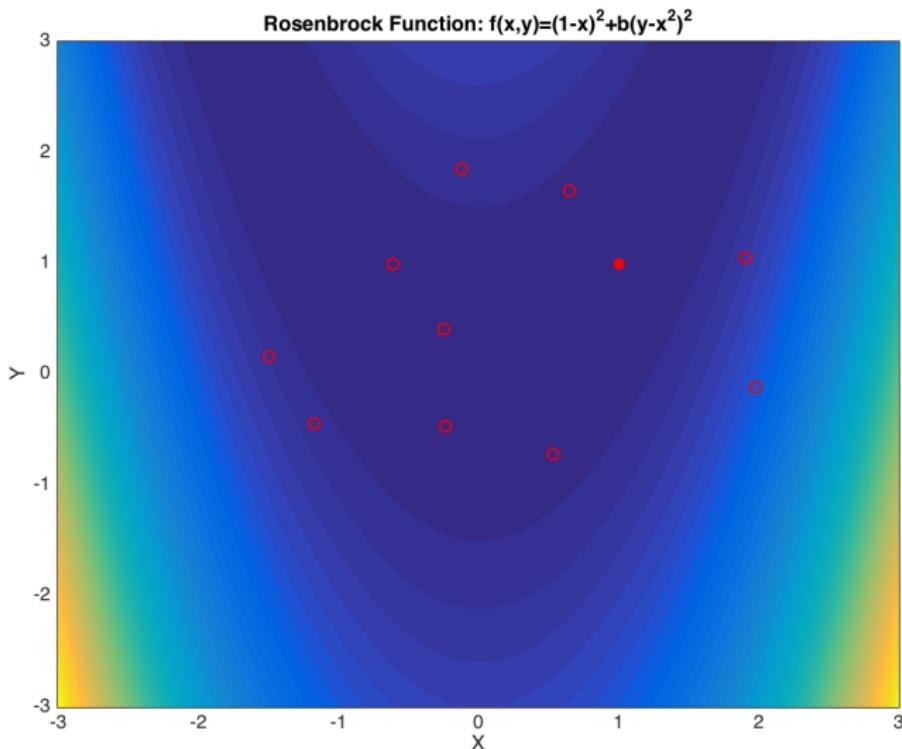
## HEURISTICS

- ▶ Do Nelder Mead or Pattern Search before you go to anything genetic/simulated!
- ▶ There's a natural time trap when something takes a long time to optimize
- ▶ It's rarely *actually* necessary

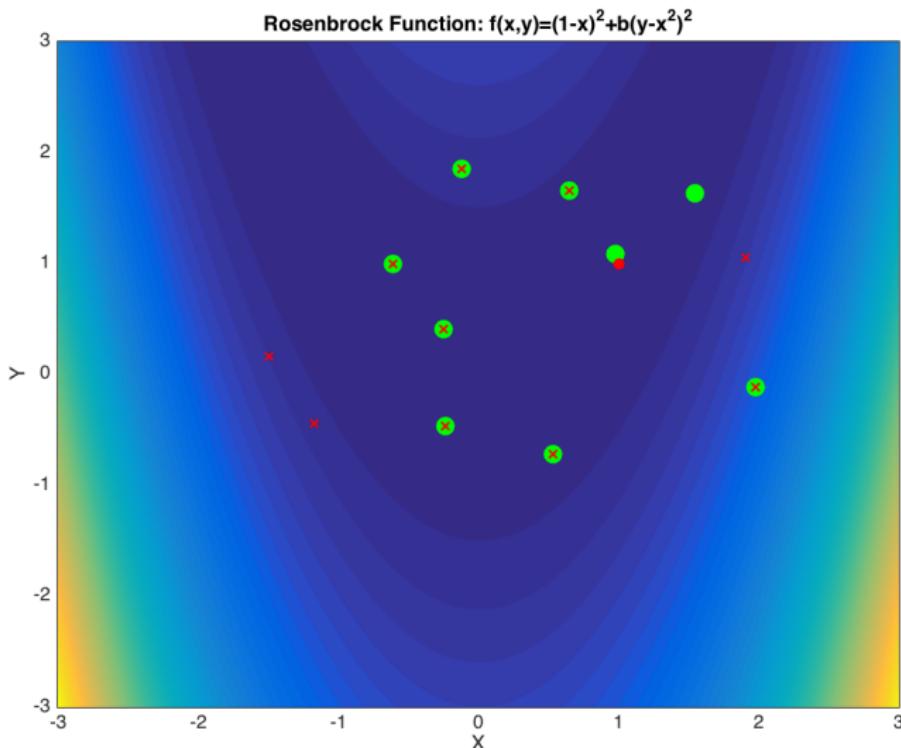
## DIFFERENTIAL EVOLUTION

- ▶ Start with cloud of points
- ▶ For each point, create a competitor by:
  - ▶ Taking two other points and defining their difference as a new direction vector
  - ▶ Take a third point and add that direction vector to define competitor point
- ▶ If the competitor is better, take it, otherwise, keep the old point
- ▶ Repeat until you haven't accepted any new competitors for a long time, or you've reached a maximum level

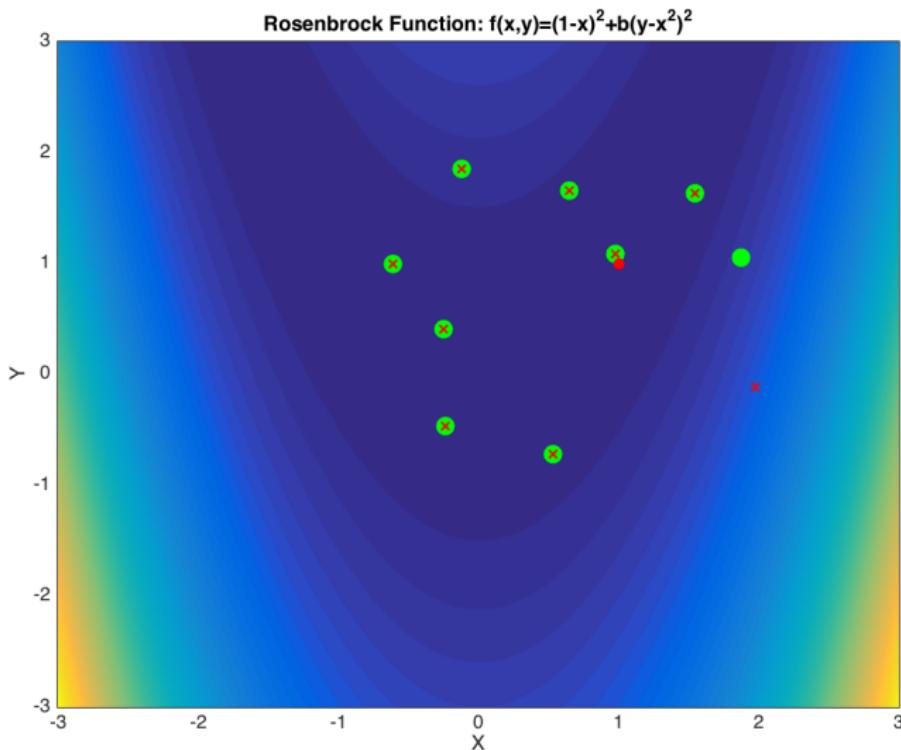
# DIFFERENTIAL EVOLUTION



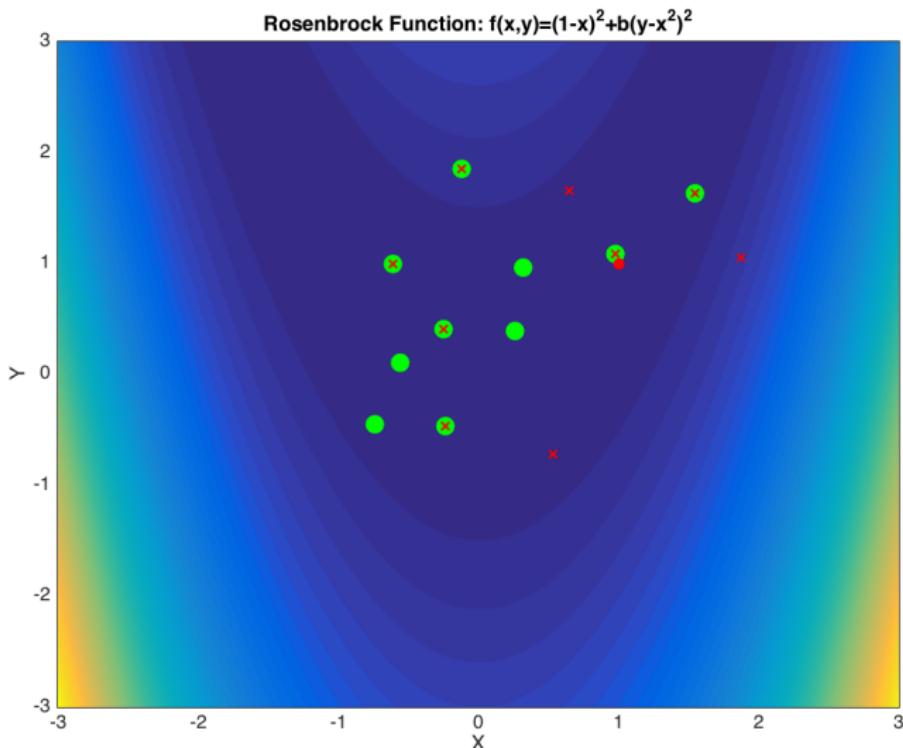
# DIFFERENTIAL EVOLUTION



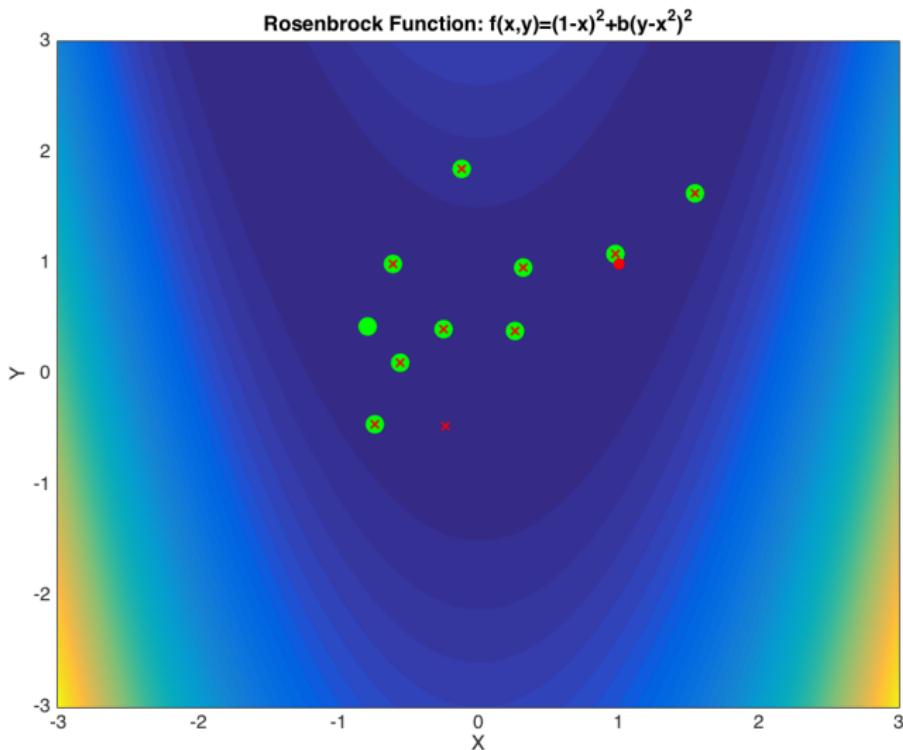
# DIFFERENTIAL EVOLUTION



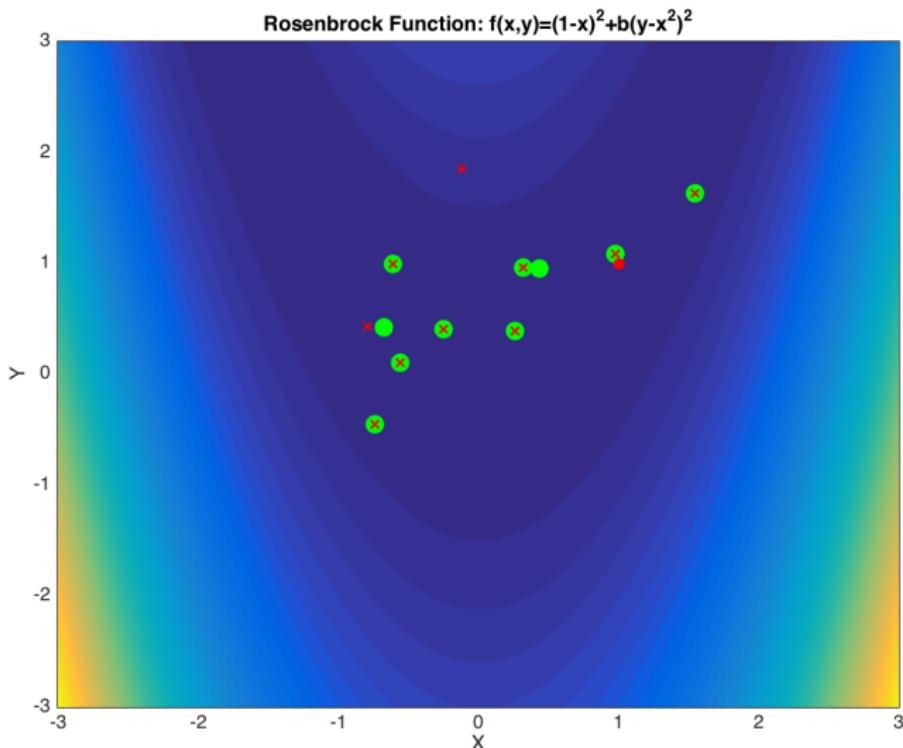
# DIFFERENTIAL EVOLUTION



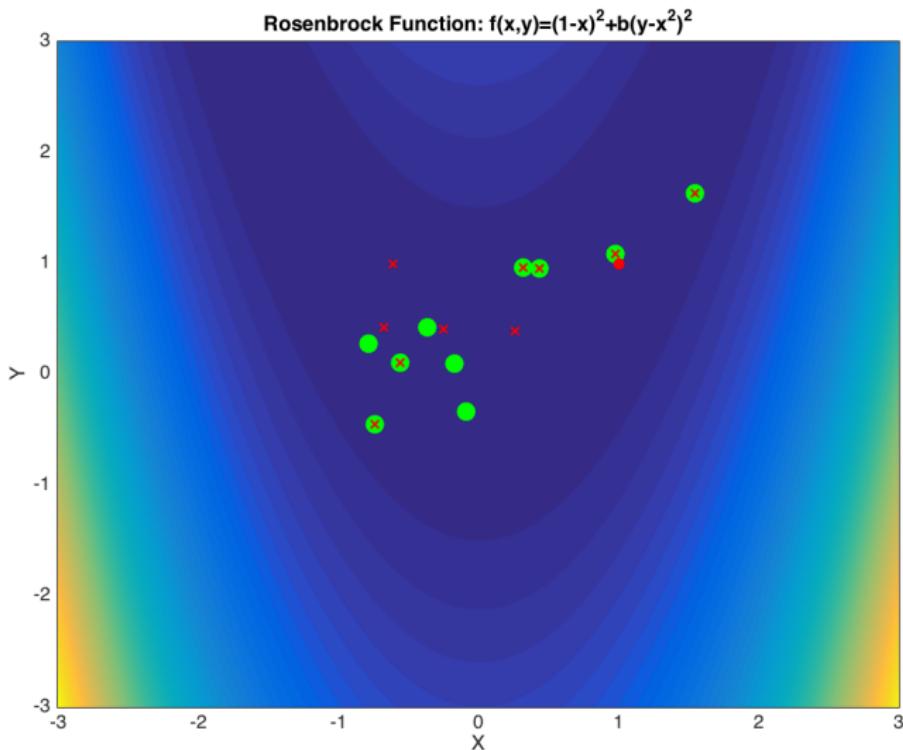
# DIFFERENTIAL EVOLUTION



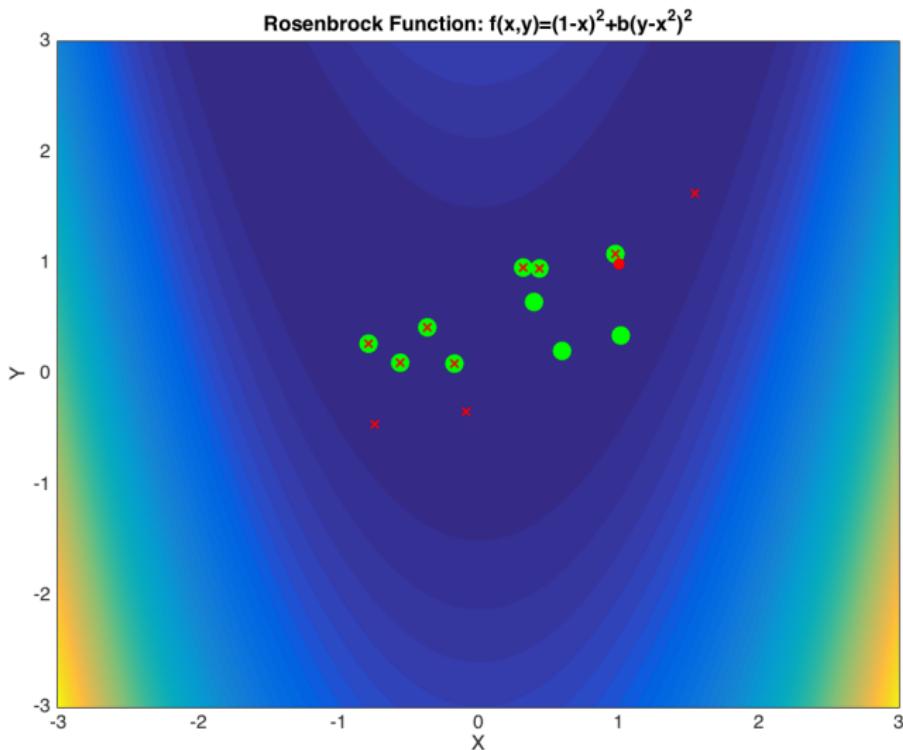
# DIFFERENTIAL EVOLUTION



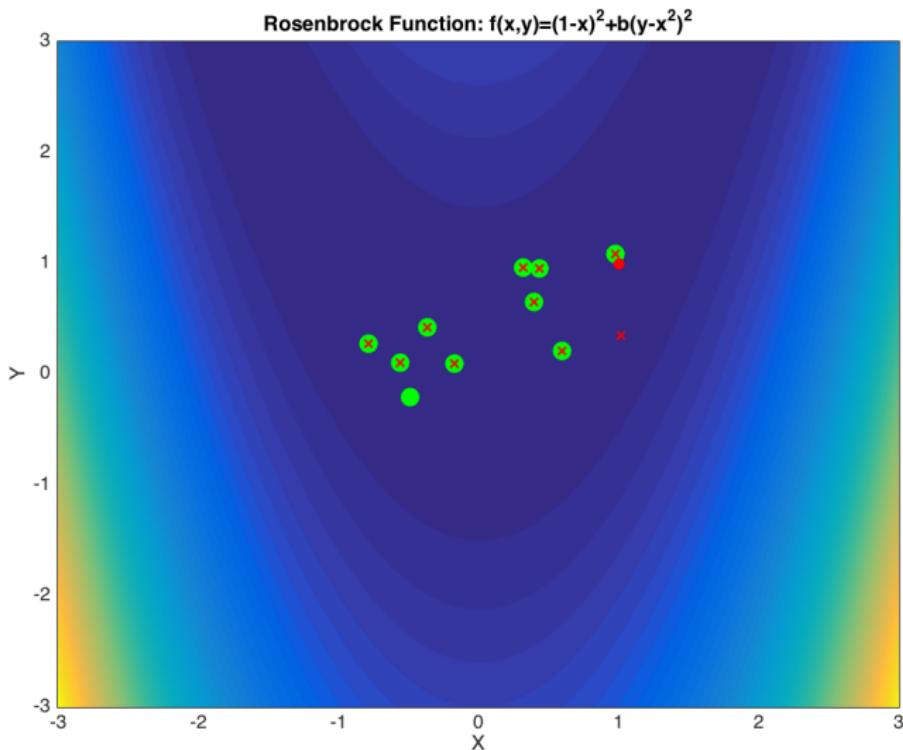
# DIFFERENTIAL EVOLUTION



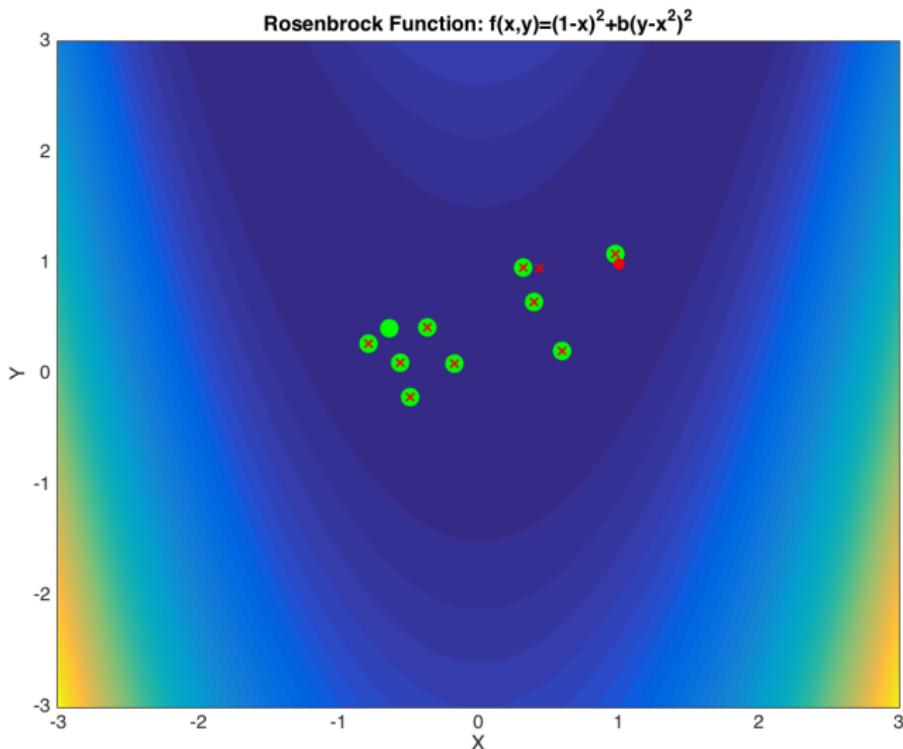
# DIFFERENTIAL EVOLUTION



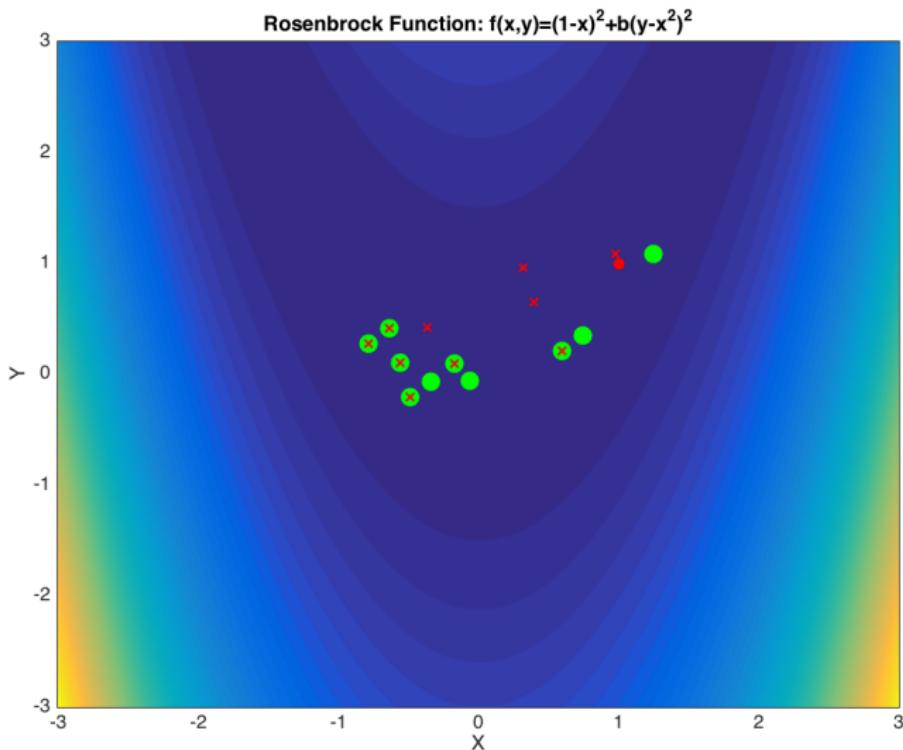
# DIFFERENTIAL EVOLUTION



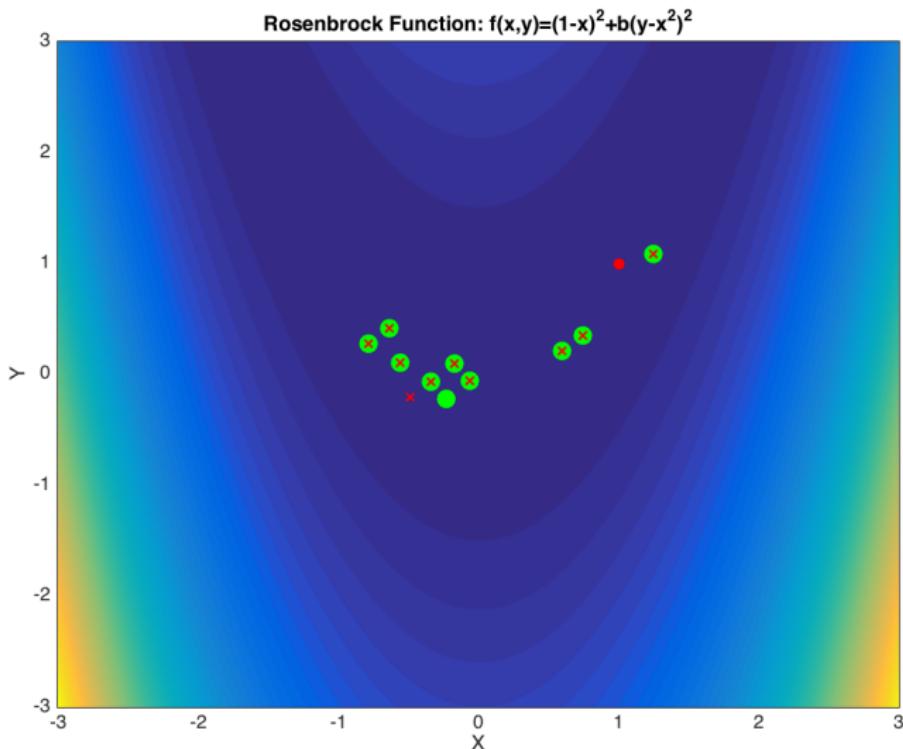
# DIFFERENTIAL EVOLUTION



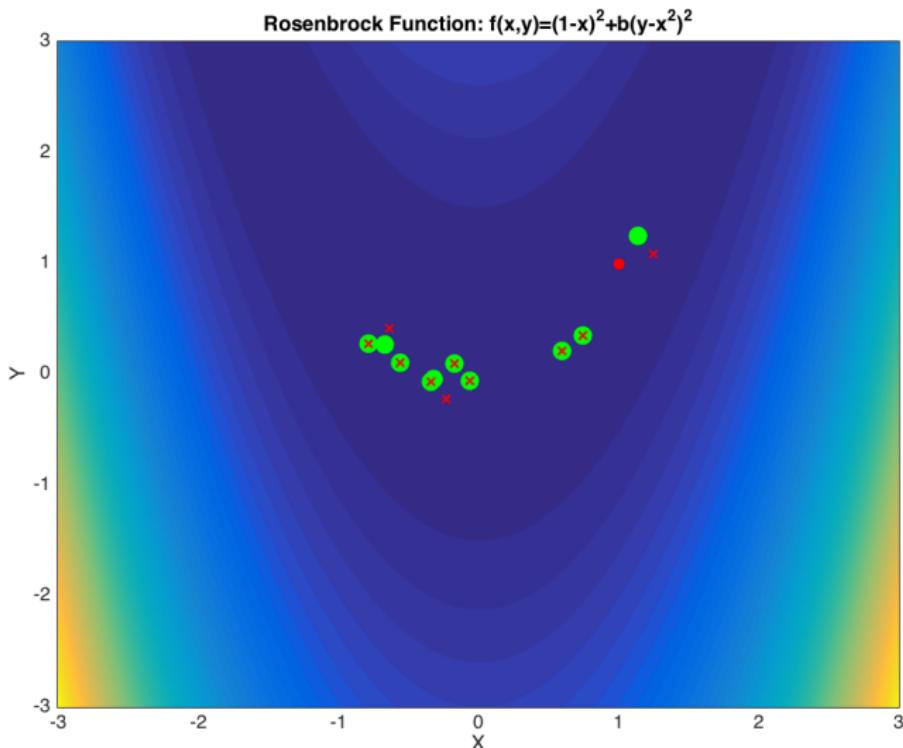
# DIFFERENTIAL EVOLUTION



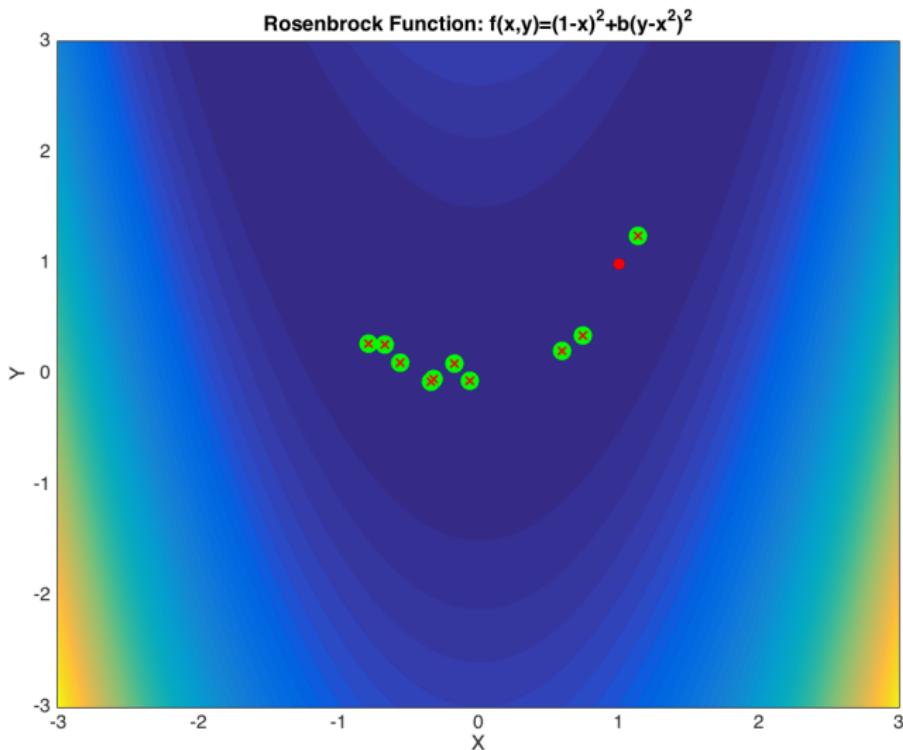
# DIFFERENTIAL EVOLUTION



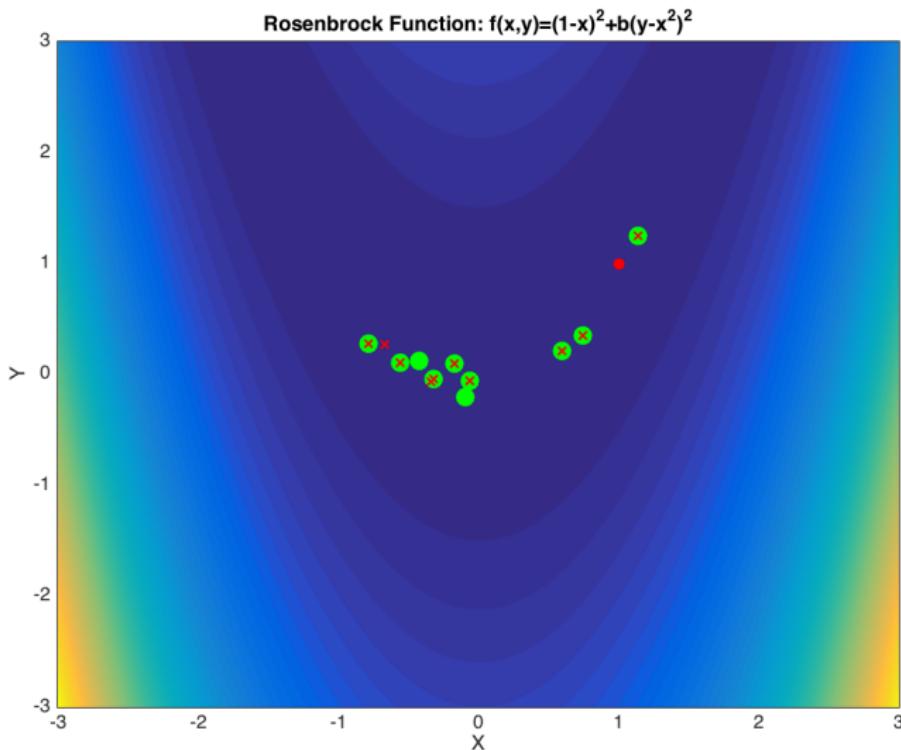
# DIFFERENTIAL EVOLUTION



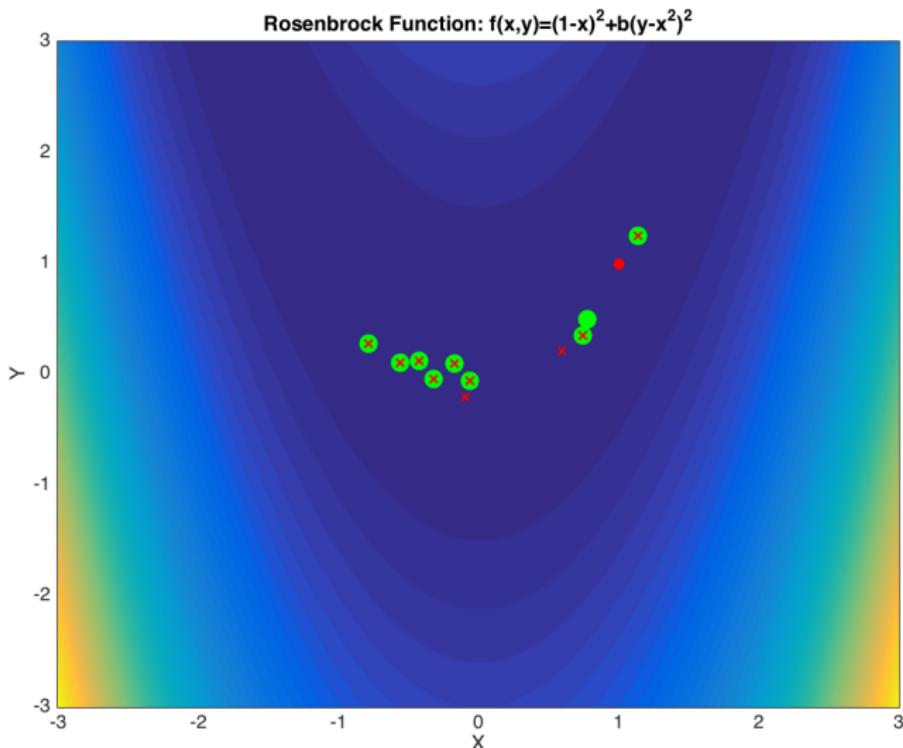
# DIFFERENTIAL EVOLUTION



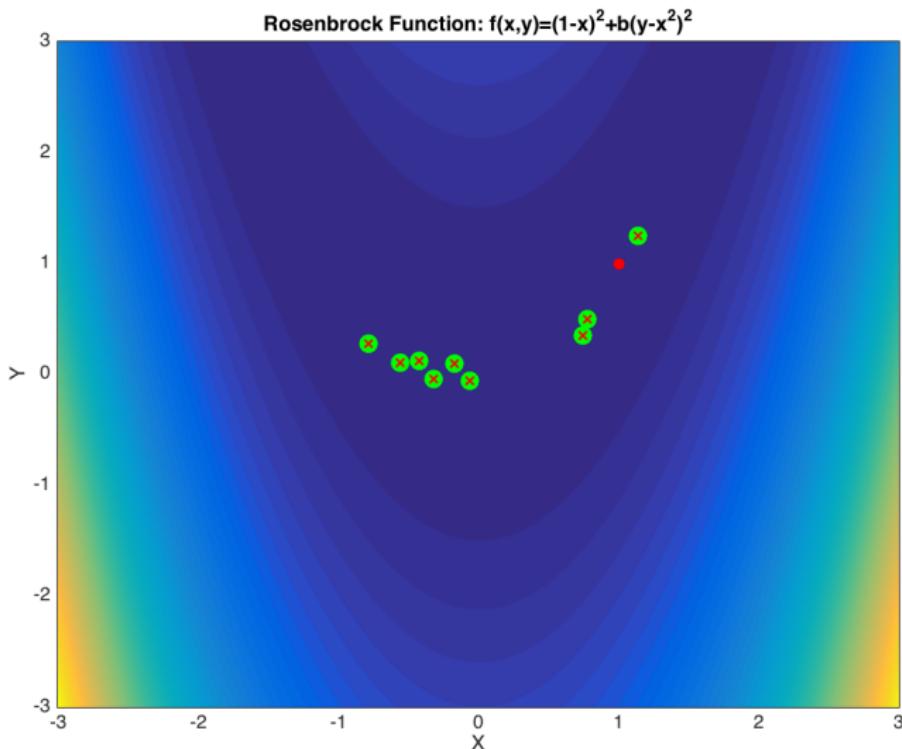
# DIFFERENTIAL EVOLUTION



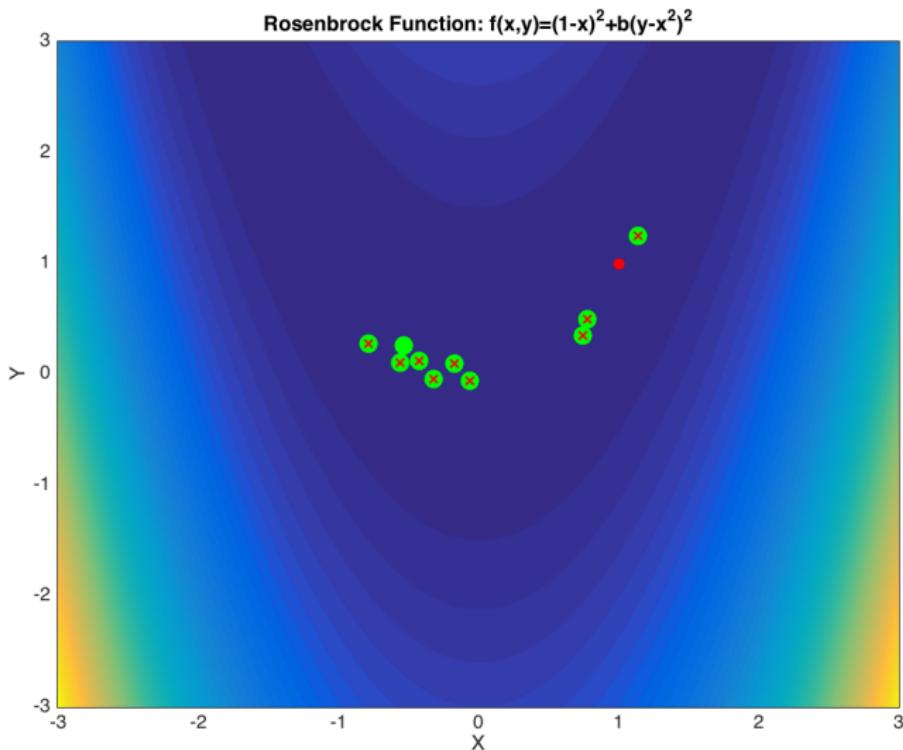
# DIFFERENTIAL EVOLUTION



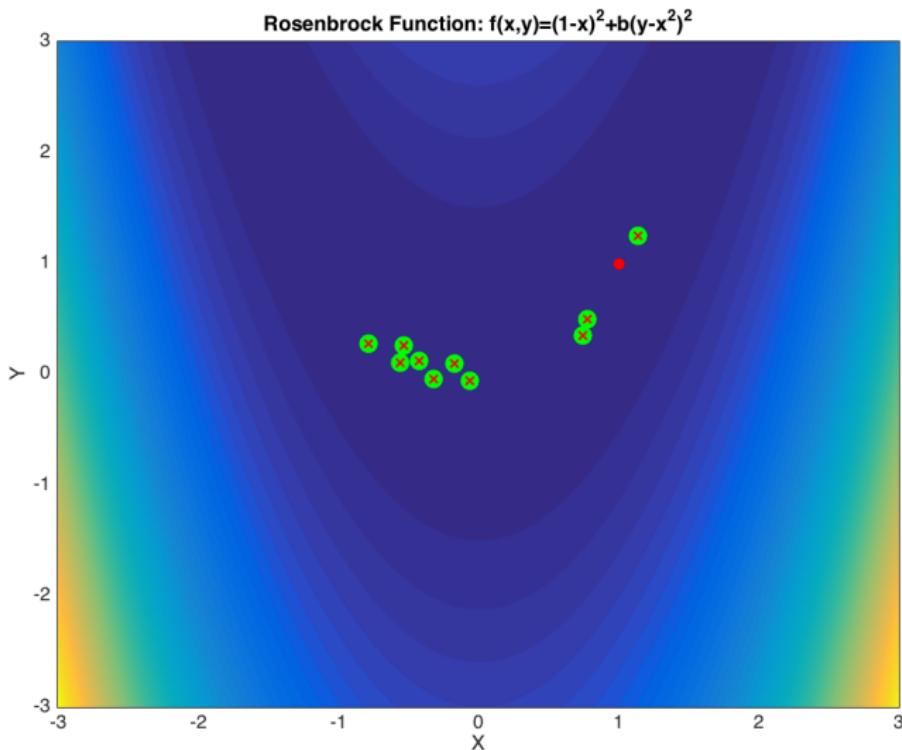
# DIFFERENTIAL EVOLUTION



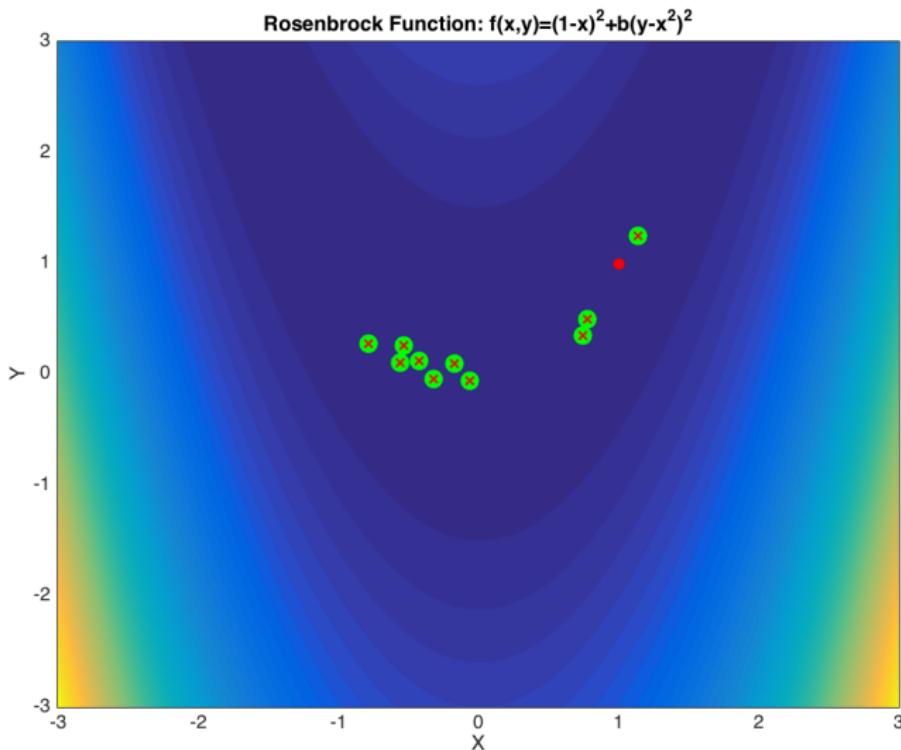
# DIFFERENTIAL EVOLUTION



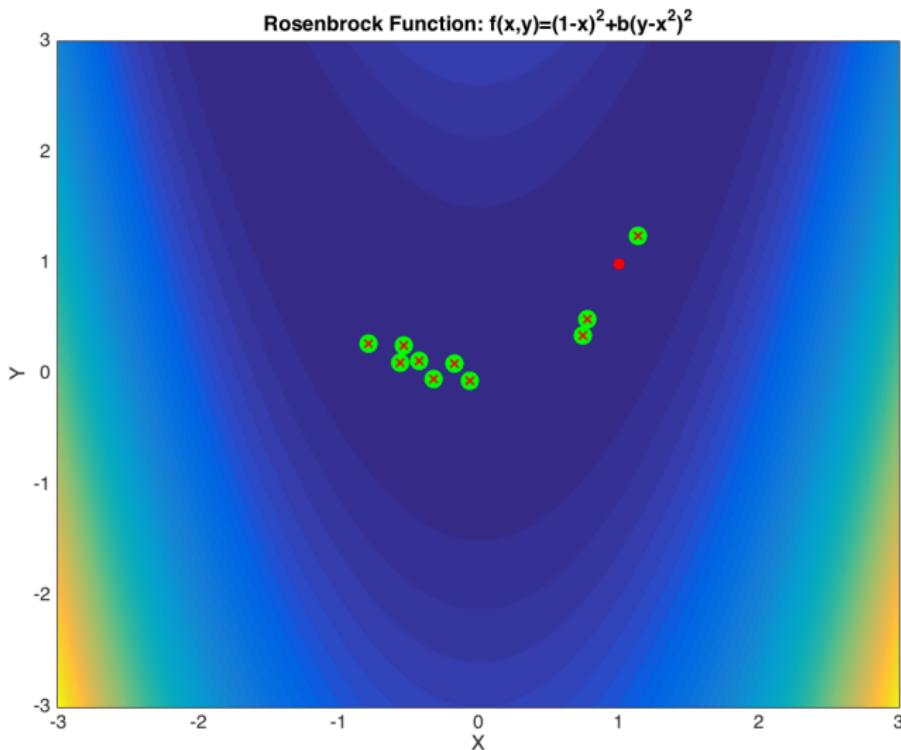
# DIFFERENTIAL EVOLUTION



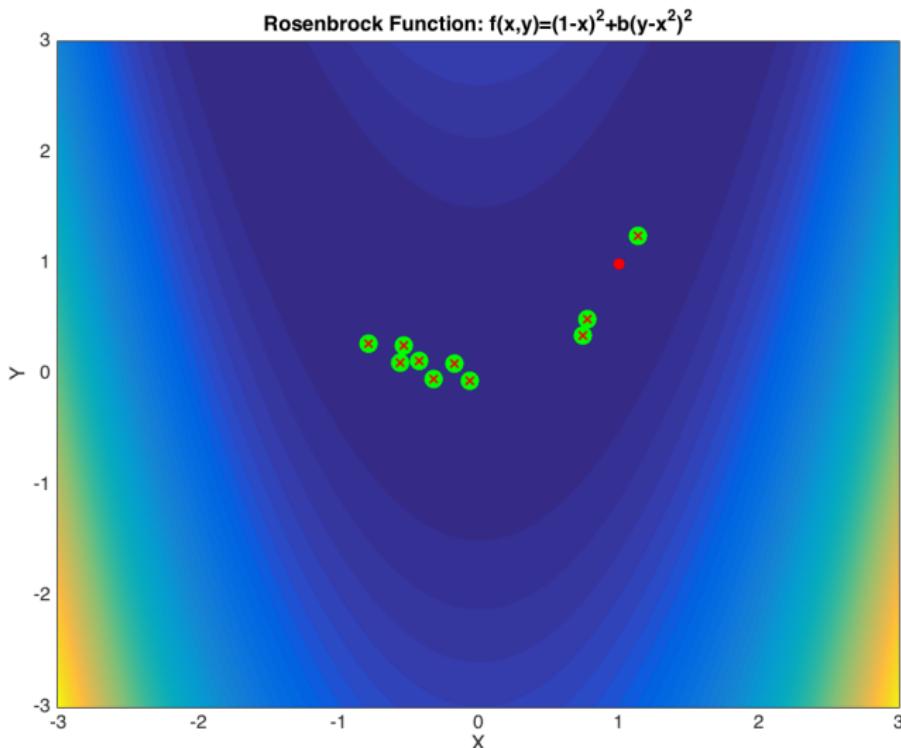
# DIFFERENTIAL EVOLUTION



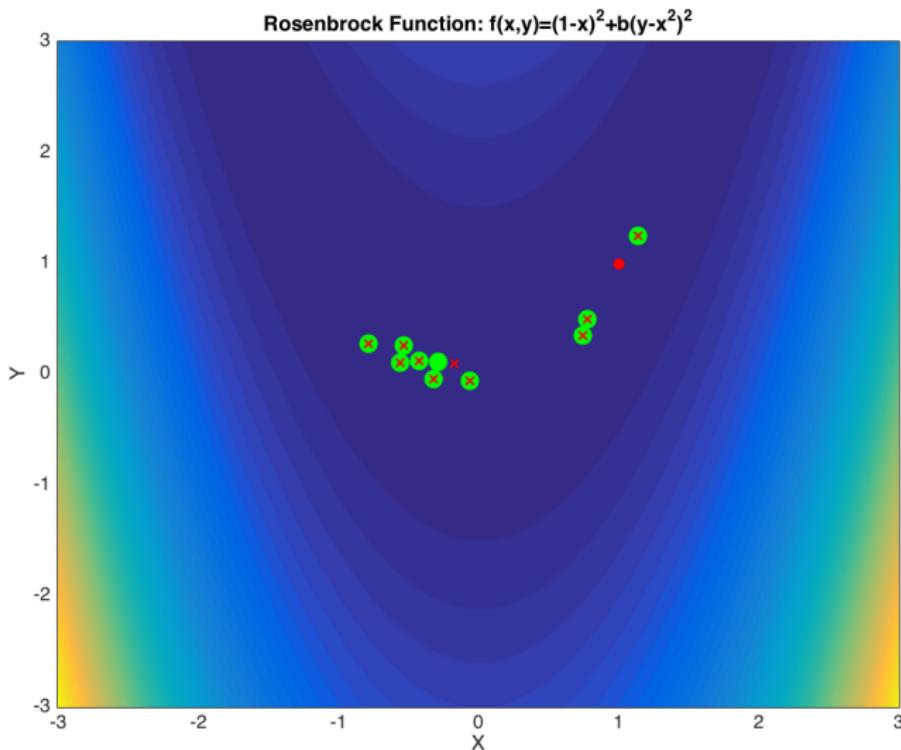
# DIFFERENTIAL EVOLUTION



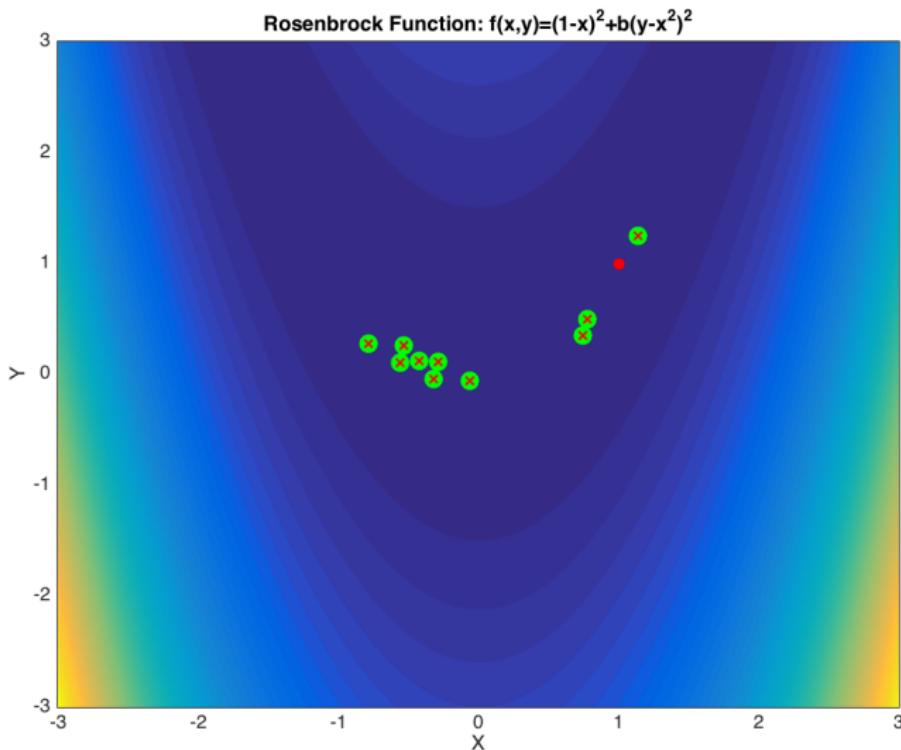
# DIFFERENTIAL EVOLUTION



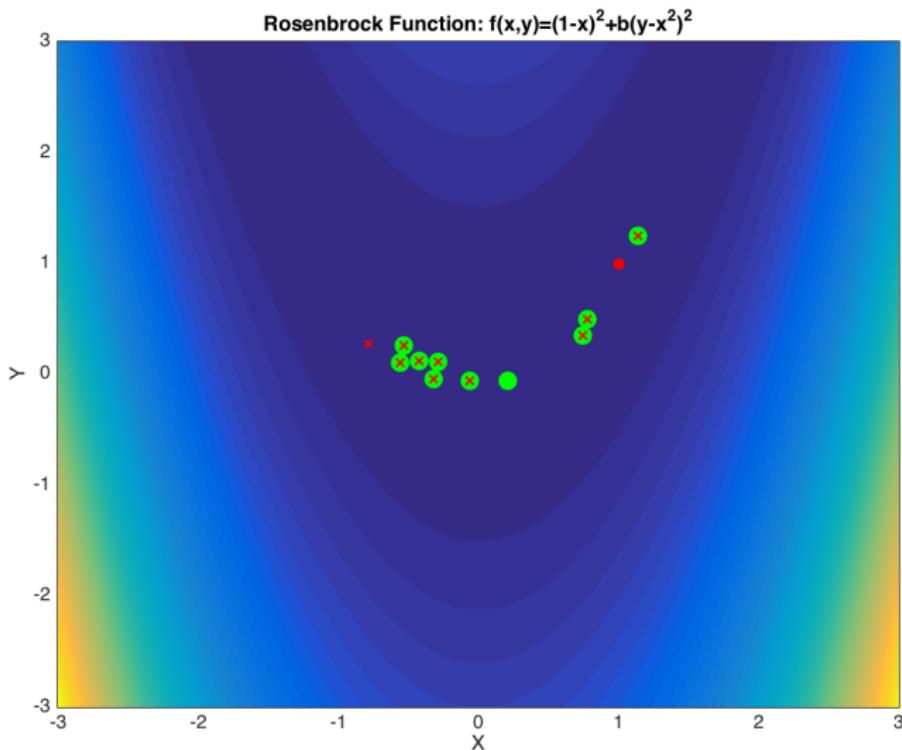
# DIFFERENTIAL EVOLUTION



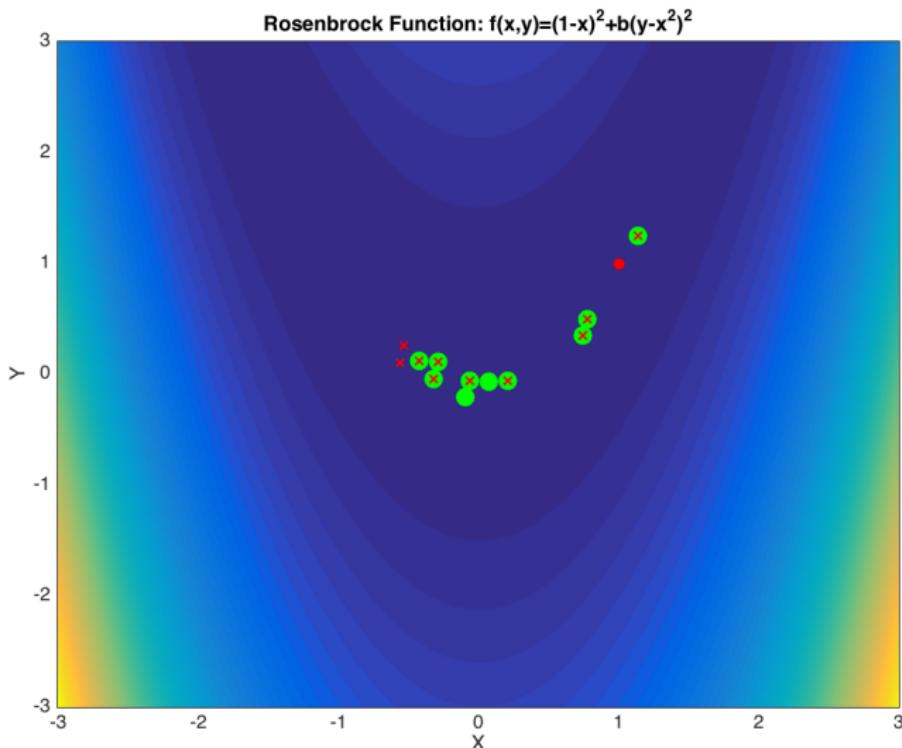
# DIFFERENTIAL EVOLUTION



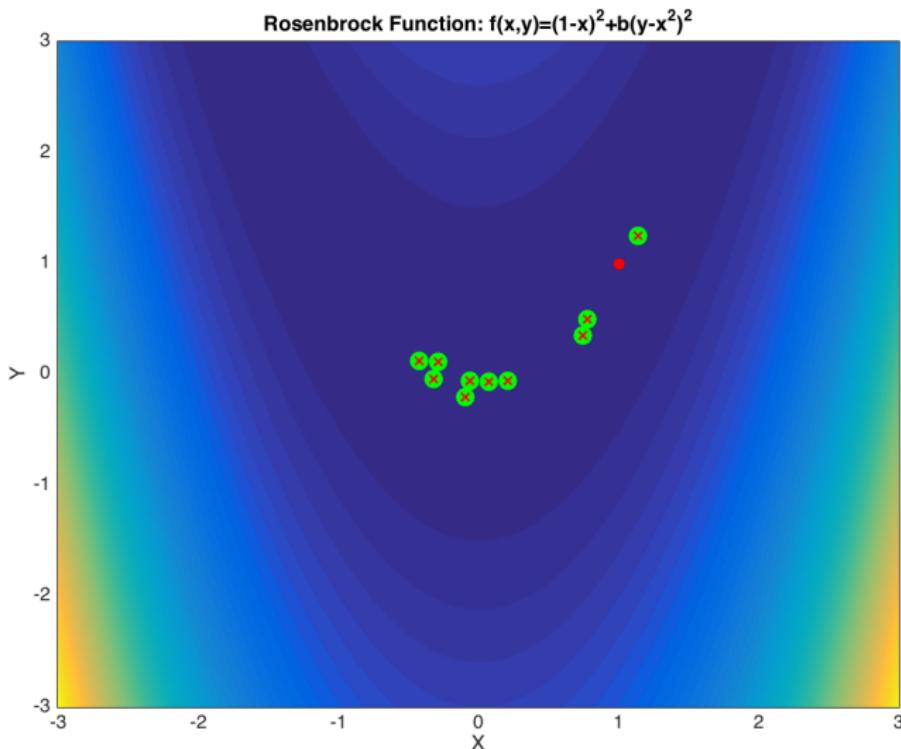
# DIFFERENTIAL EVOLUTION



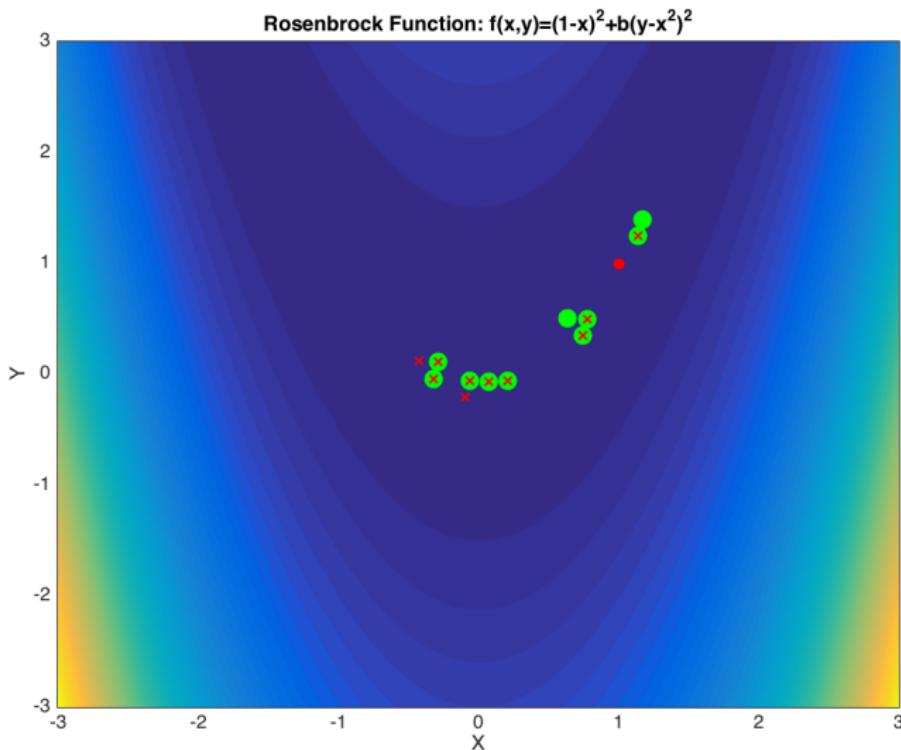
# DIFFERENTIAL EVOLUTION



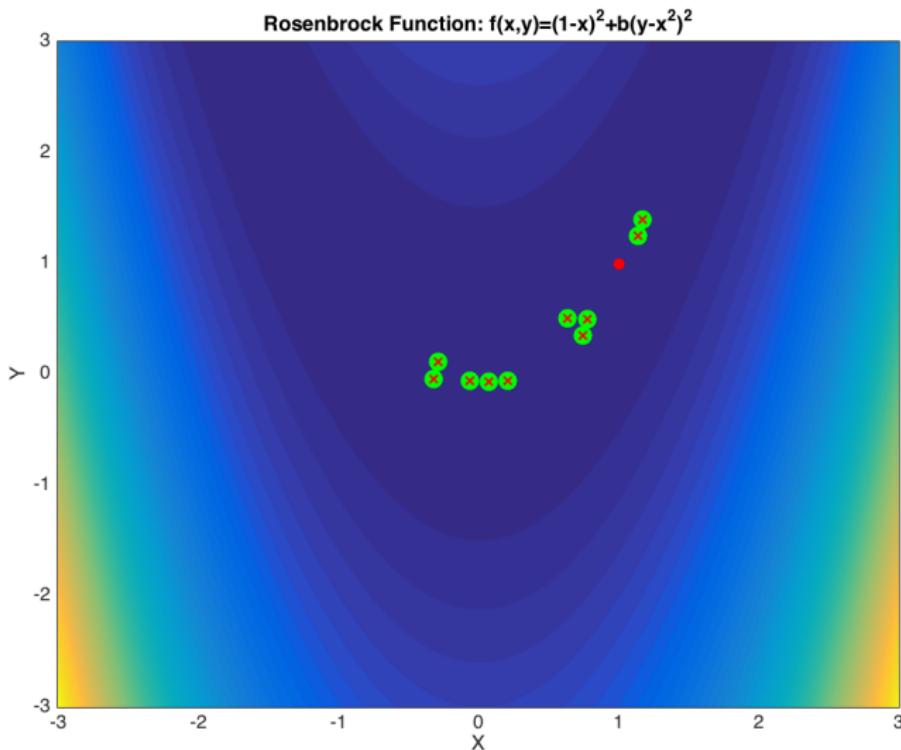
# DIFFERENTIAL EVOLUTION



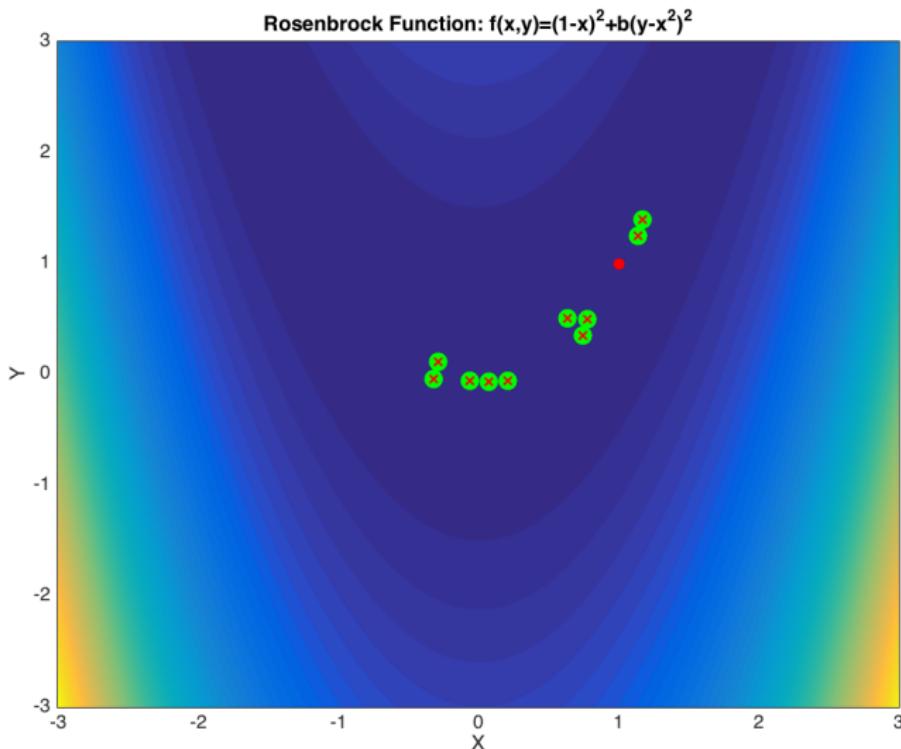
# DIFFERENTIAL EVOLUTION



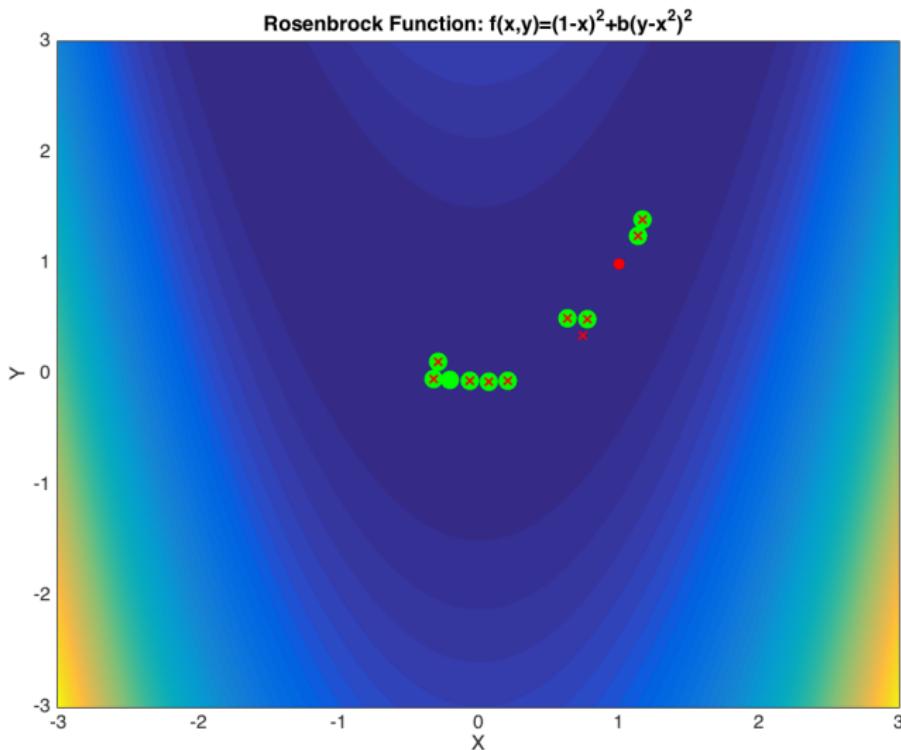
# DIFFERENTIAL EVOLUTION



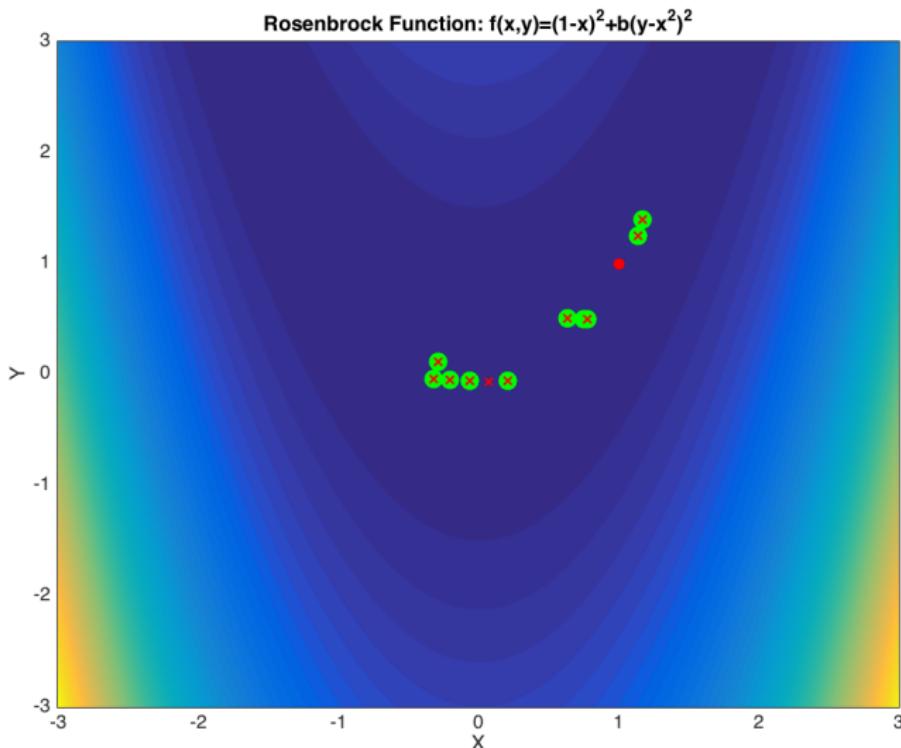
# DIFFERENTIAL EVOLUTION



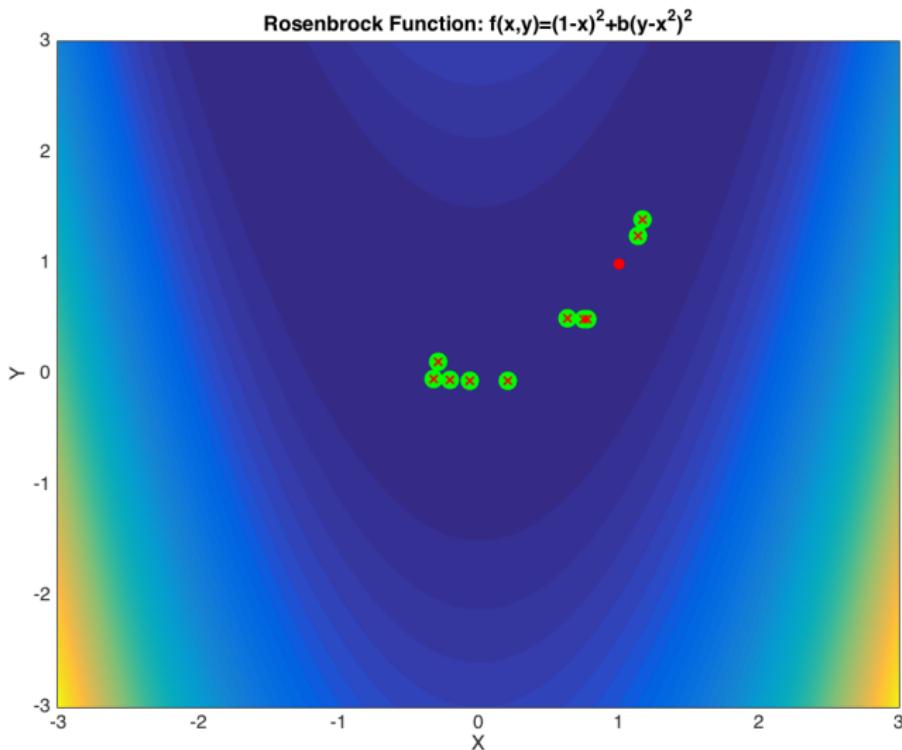
# DIFFERENTIAL EVOLUTION



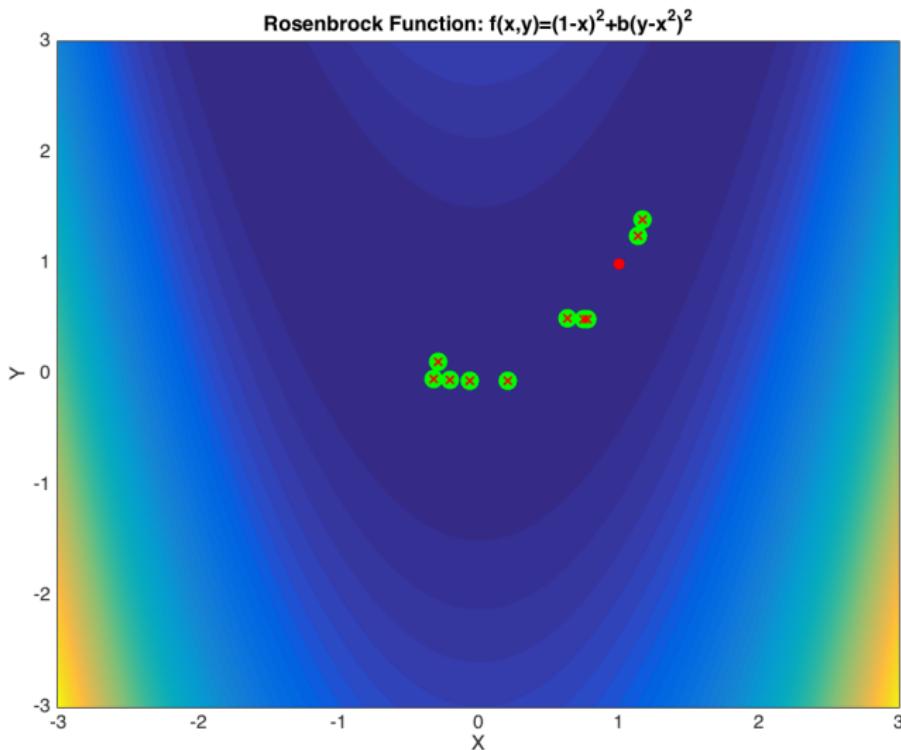
# DIFFERENTIAL EVOLUTION



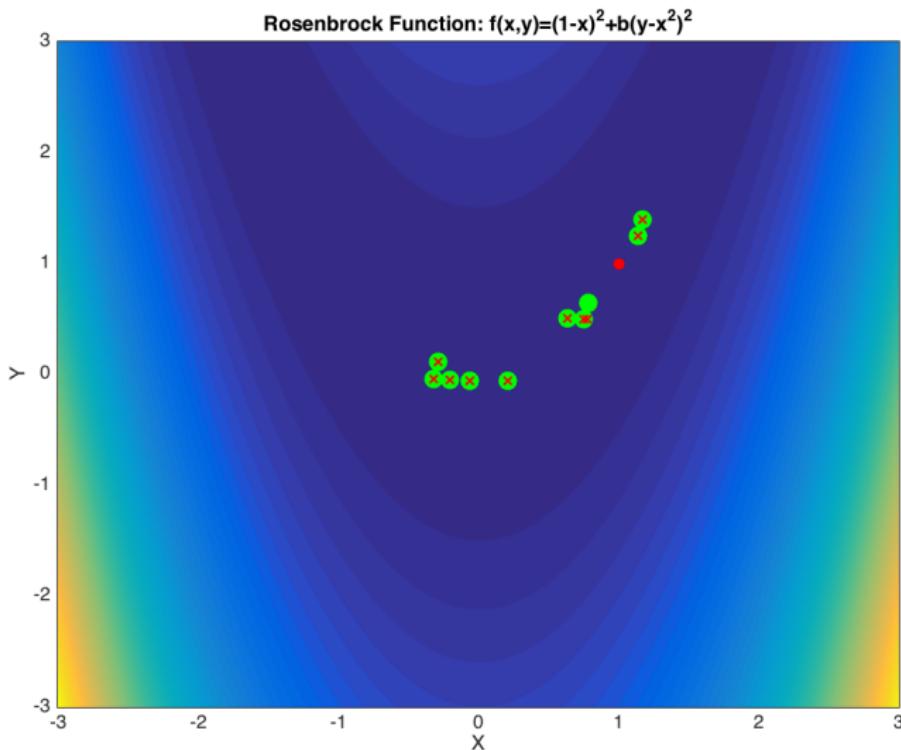
# DIFFERENTIAL EVOLUTION



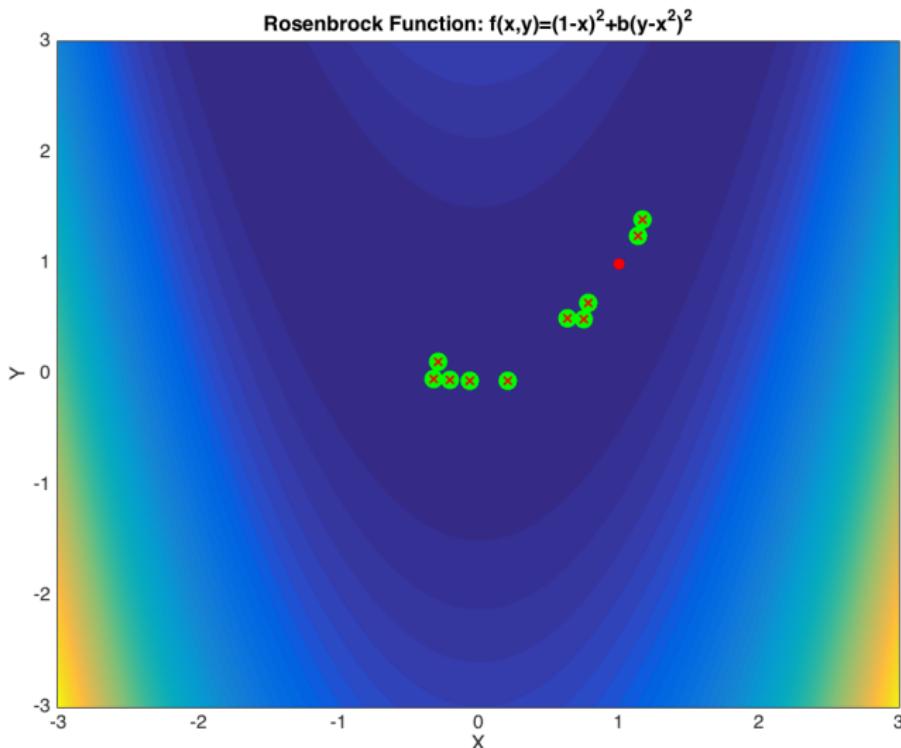
# DIFFERENTIAL EVOLUTION



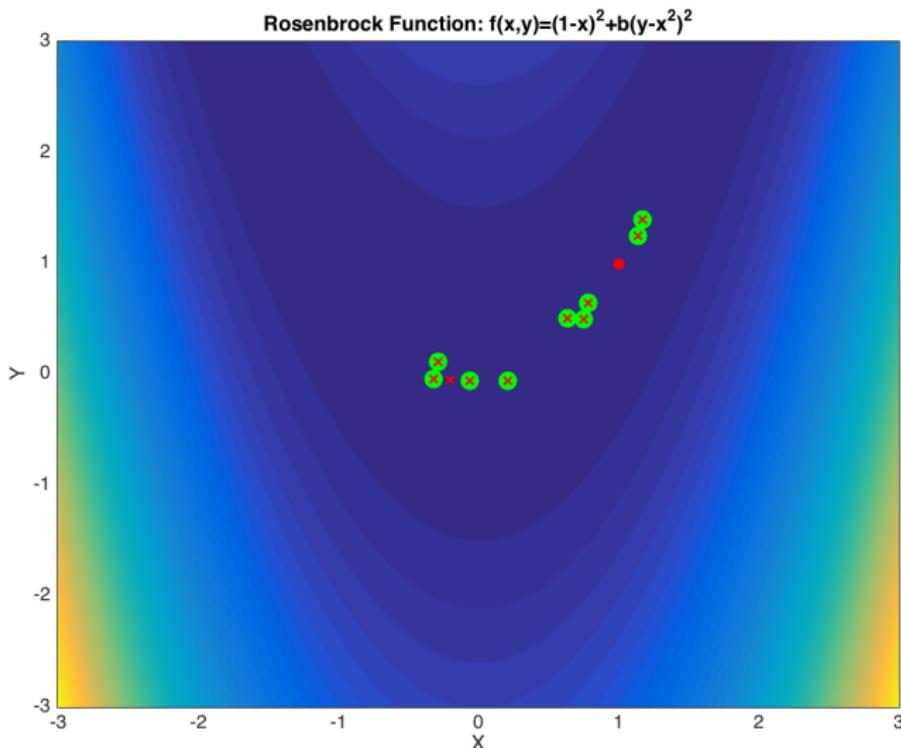
# DIFFERENTIAL EVOLUTION



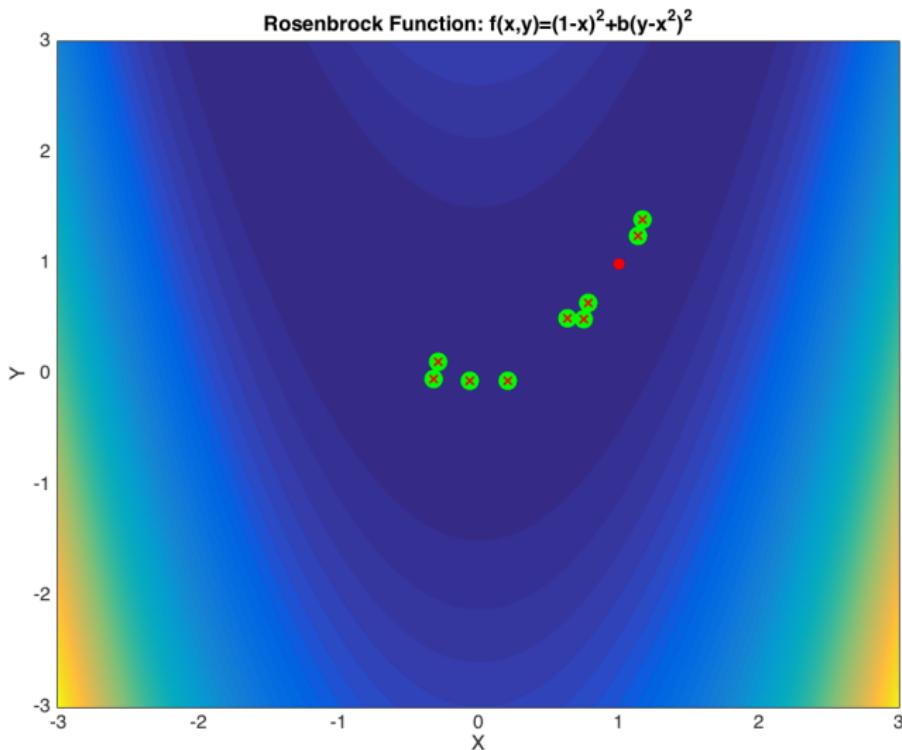
# DIFFERENTIAL EVOLUTION



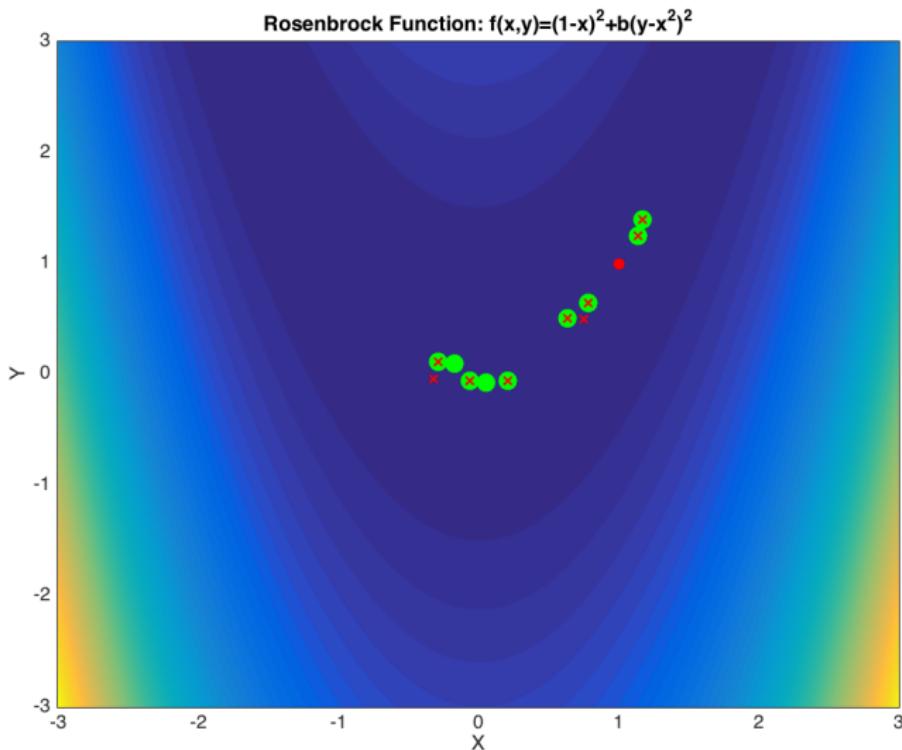
# DIFFERENTIAL EVOLUTION



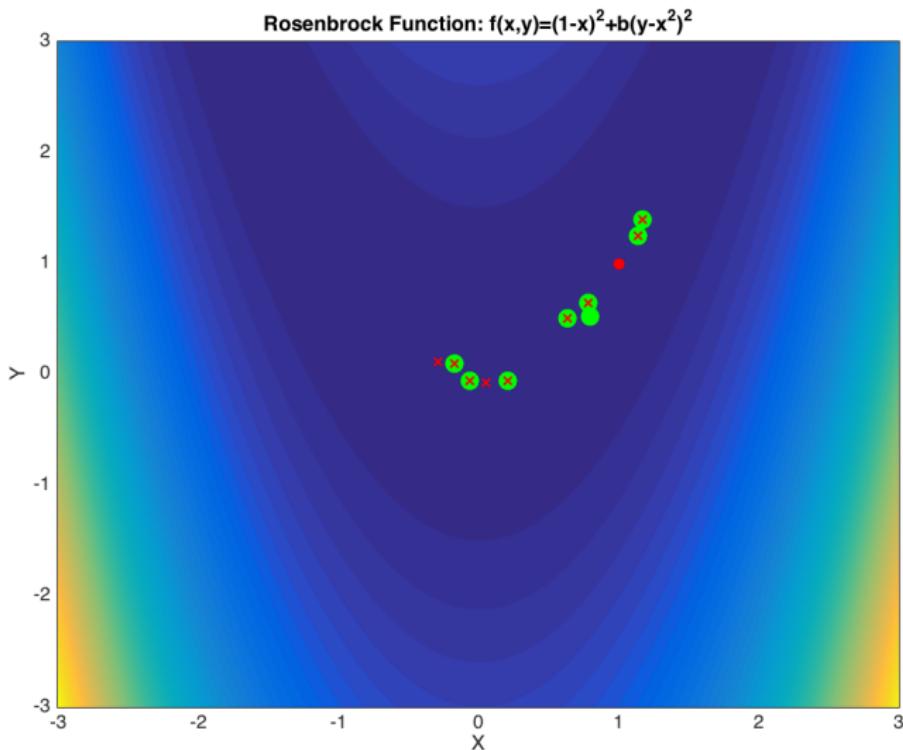
# DIFFERENTIAL EVOLUTION



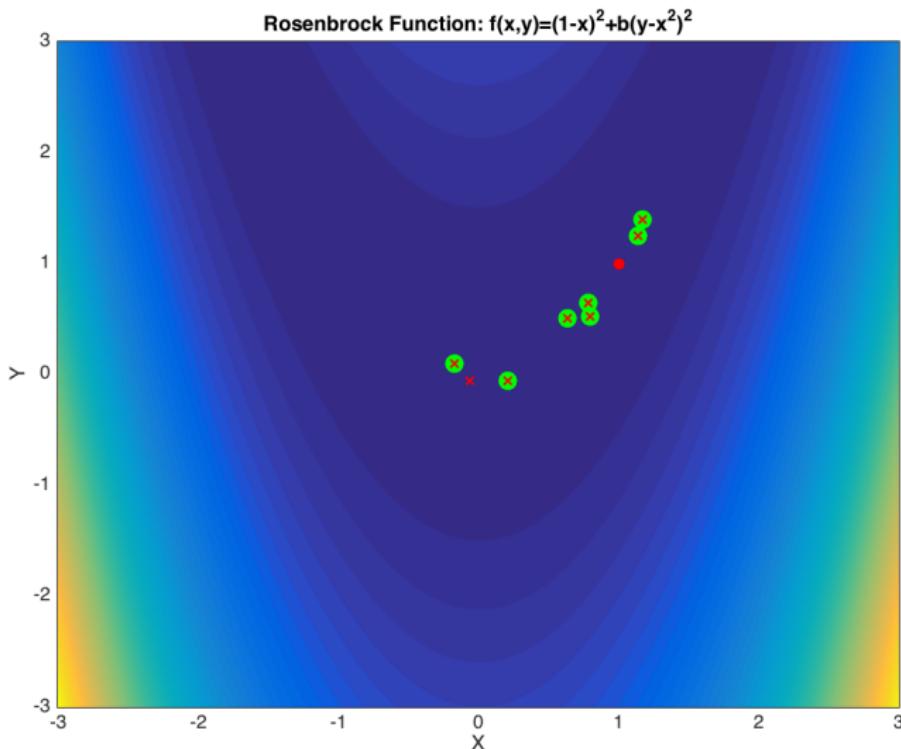
# DIFFERENTIAL EVOLUTION



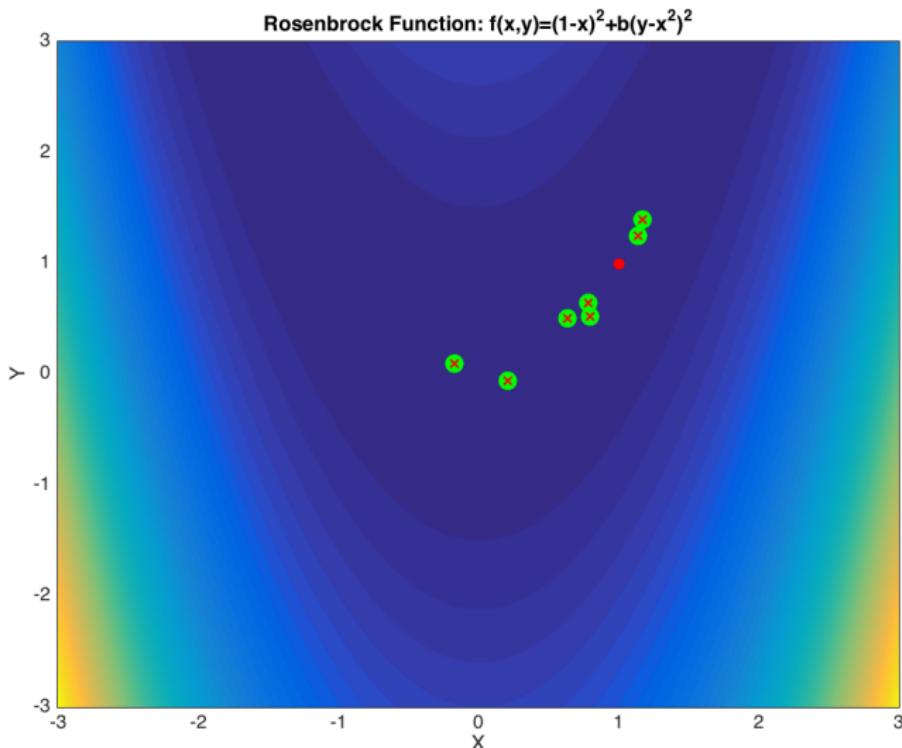
# DIFFERENTIAL EVOLUTION



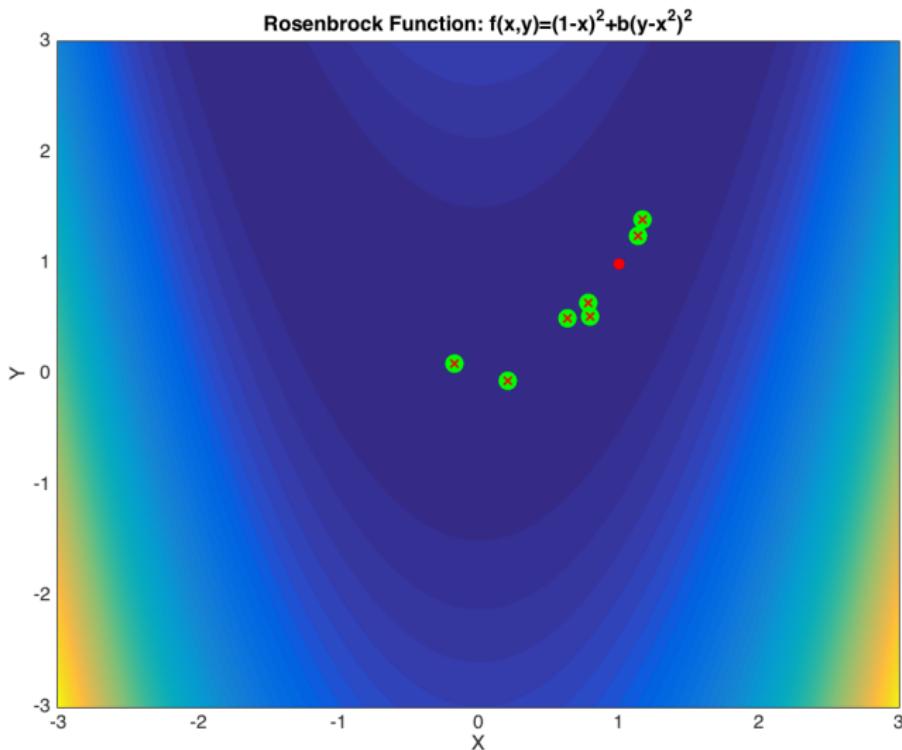
# DIFFERENTIAL EVOLUTION



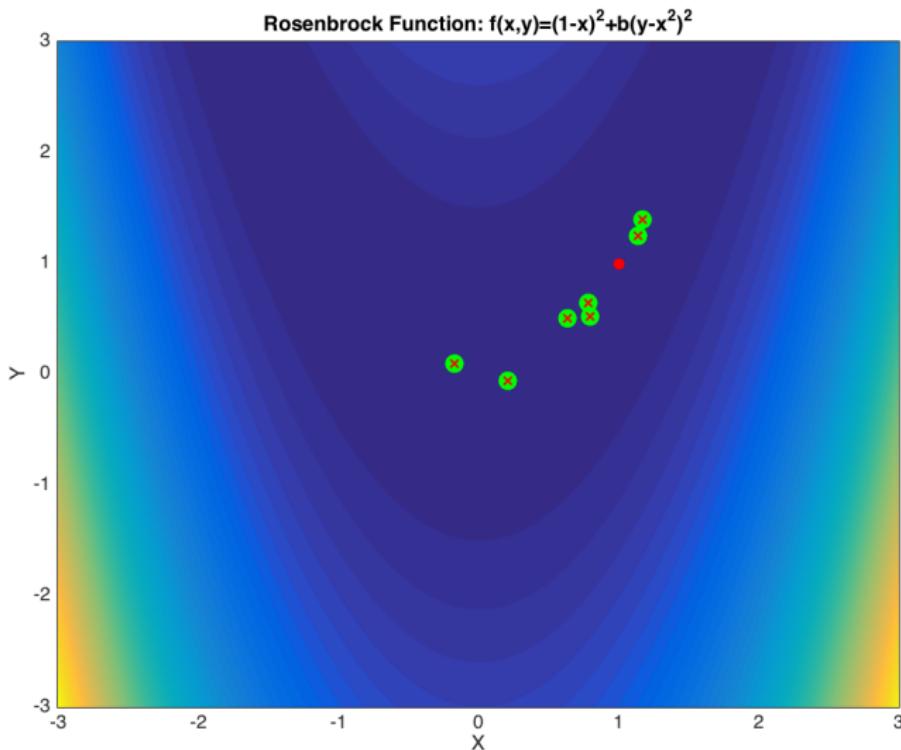
# DIFFERENTIAL EVOLUTION



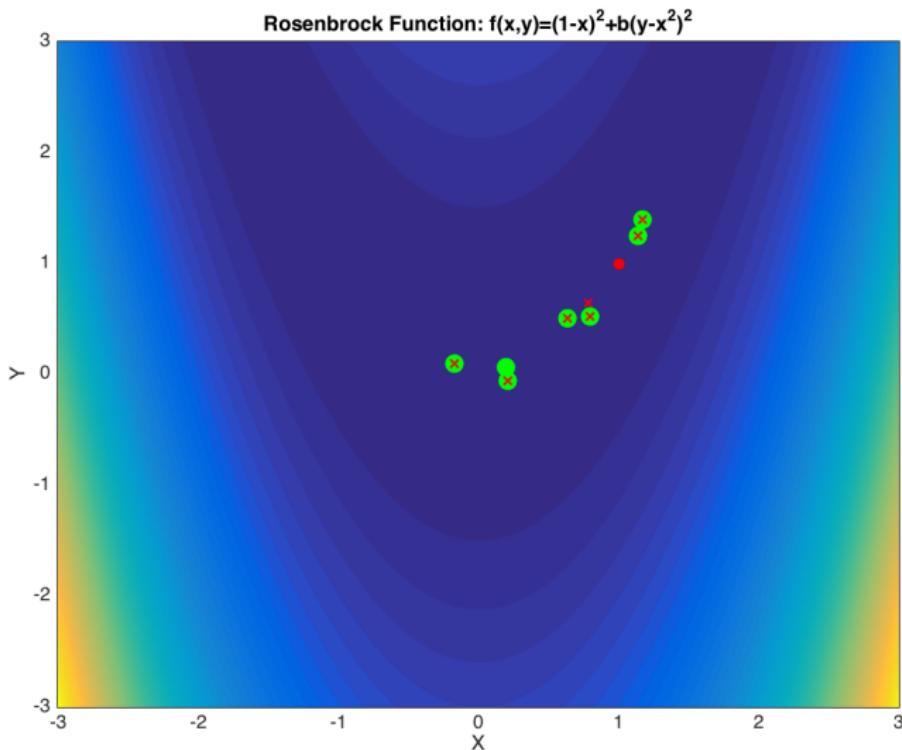
# DIFFERENTIAL EVOLUTION



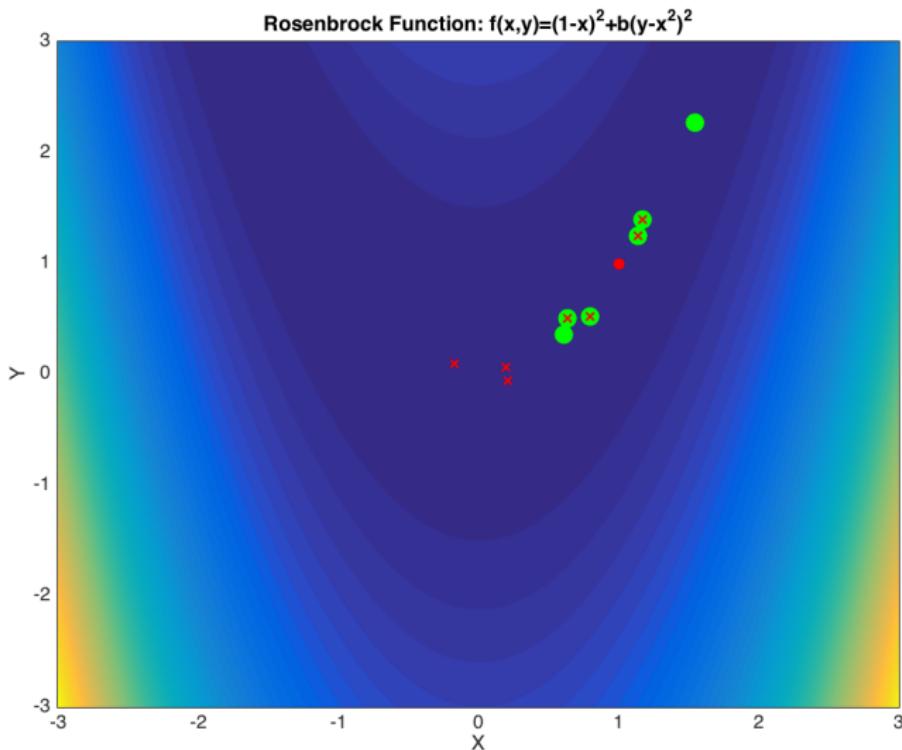
# DIFFERENTIAL EVOLUTION



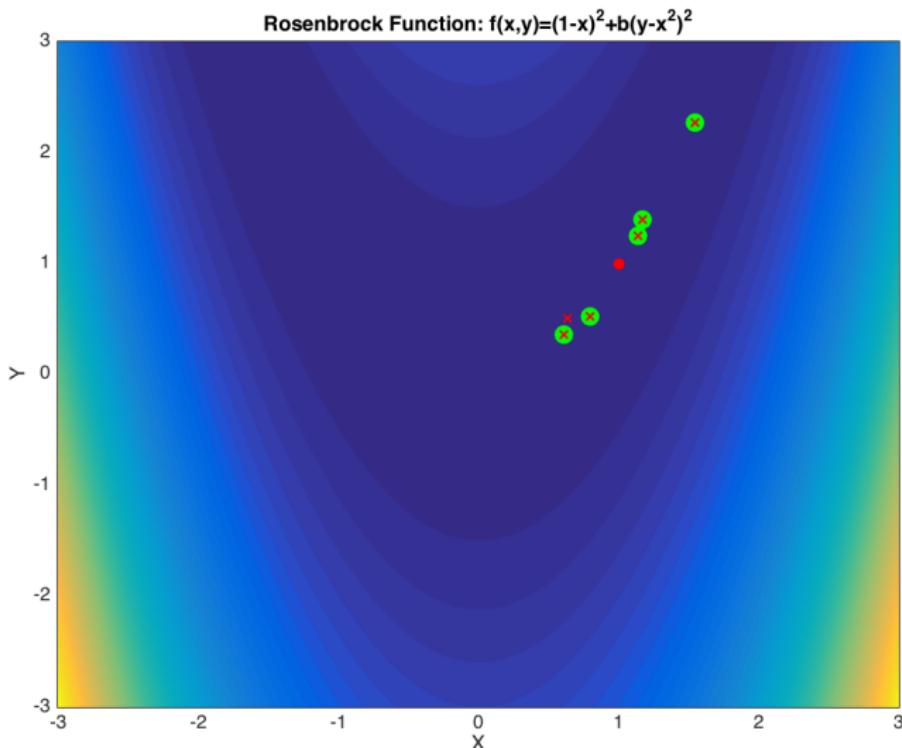
# DIFFERENTIAL EVOLUTION



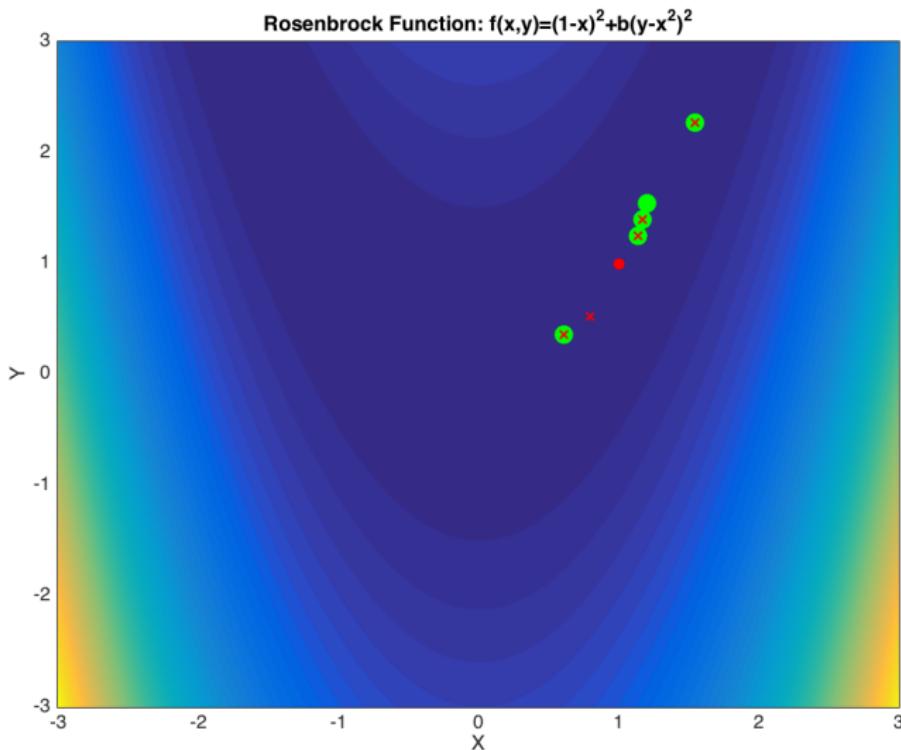
# DIFFERENTIAL EVOLUTION



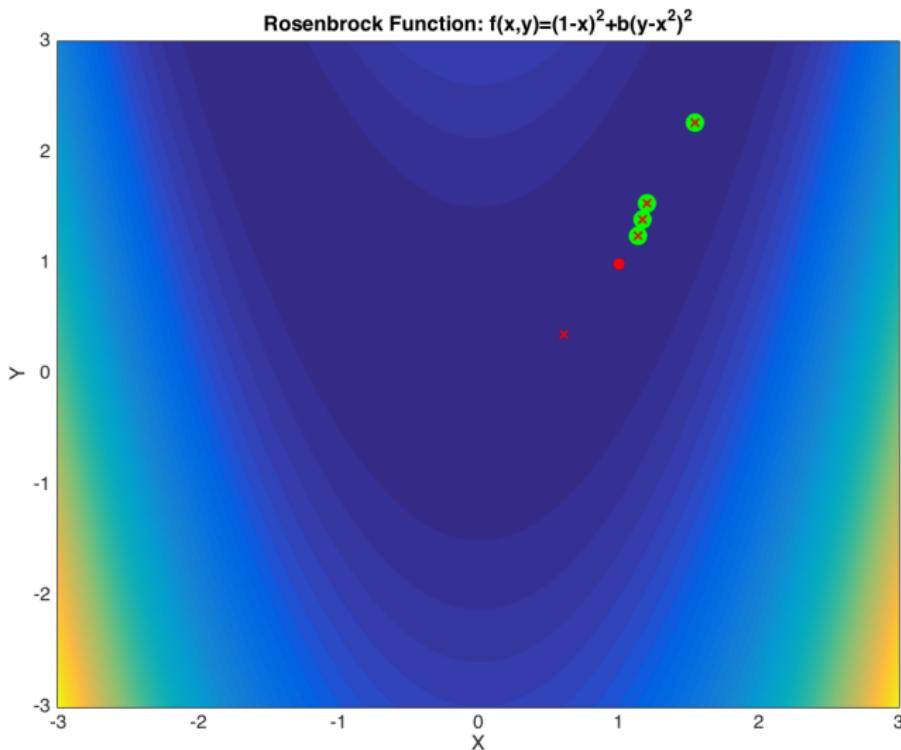
# DIFFERENTIAL EVOLUTION



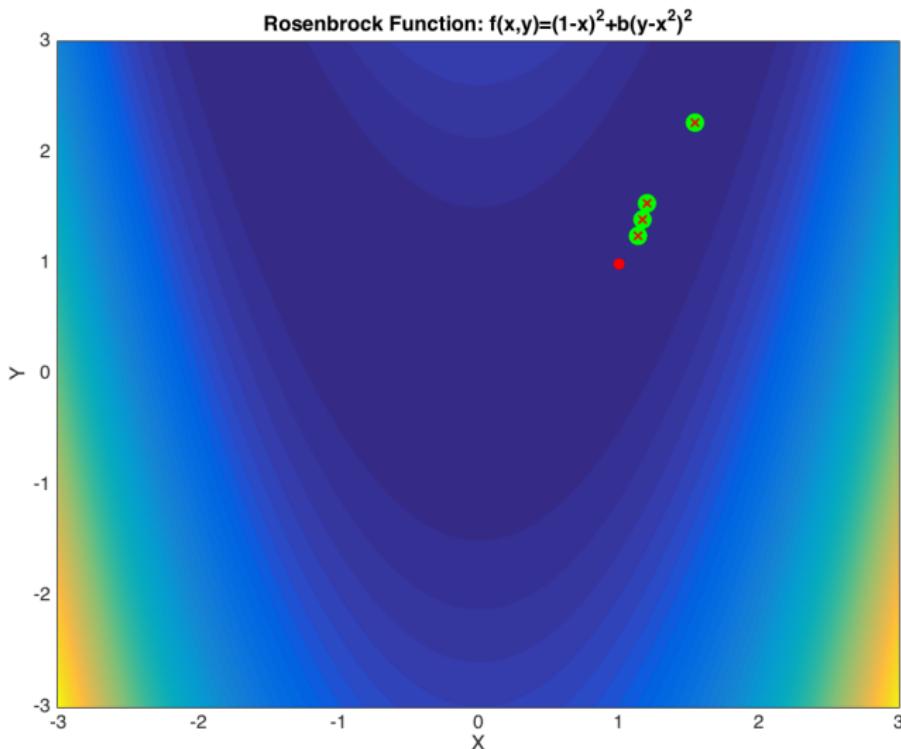
# DIFFERENTIAL EVOLUTION



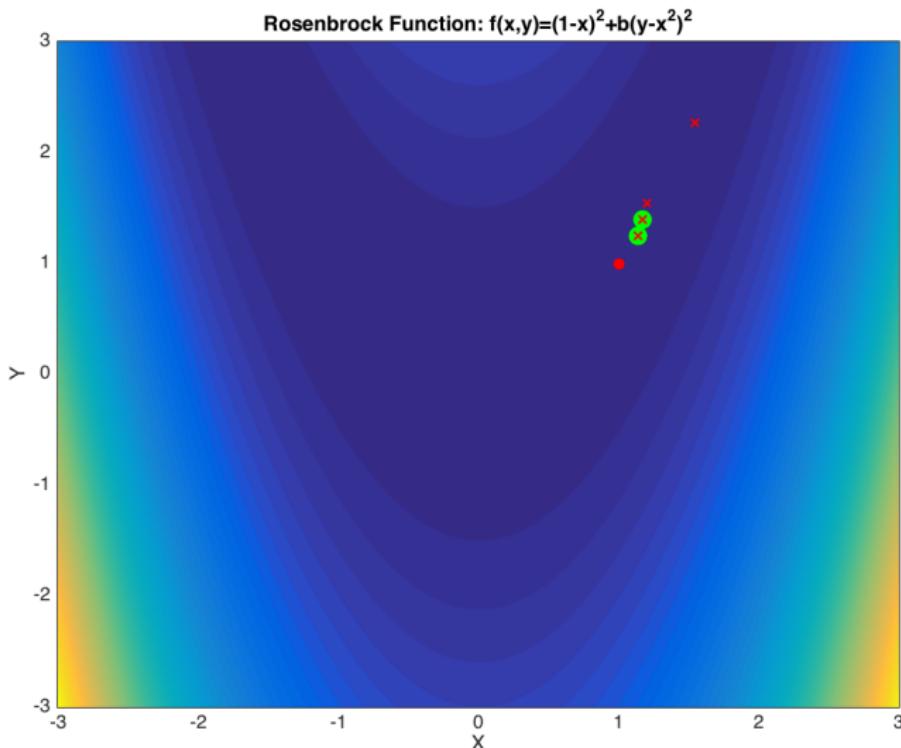
# DIFFERENTIAL EVOLUTION



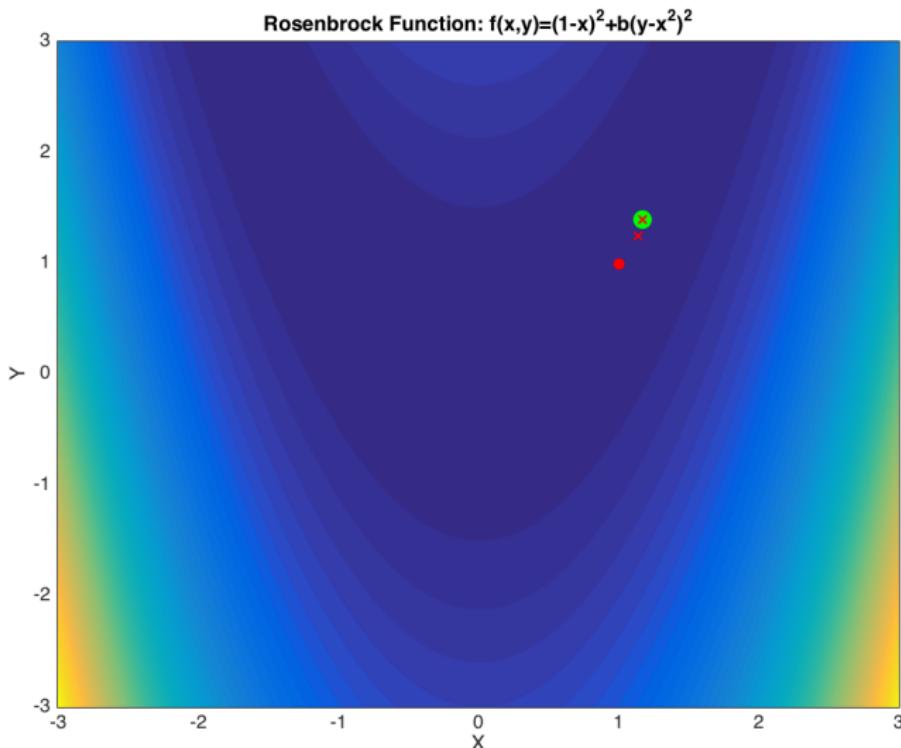
# DIFFERENTIAL EVOLUTION



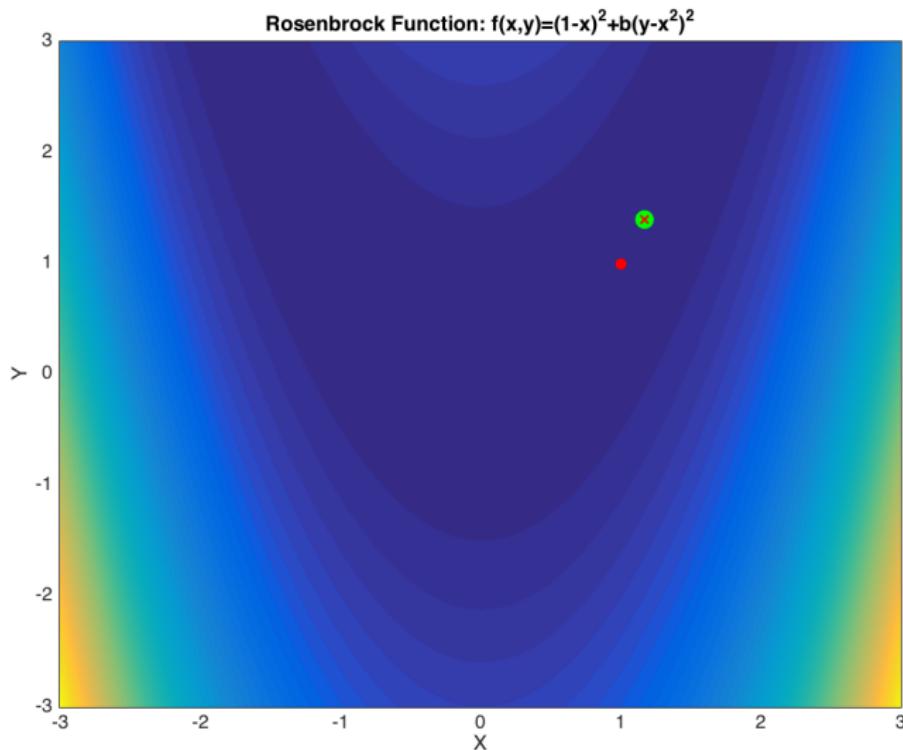
# DIFFERENTIAL EVOLUTION



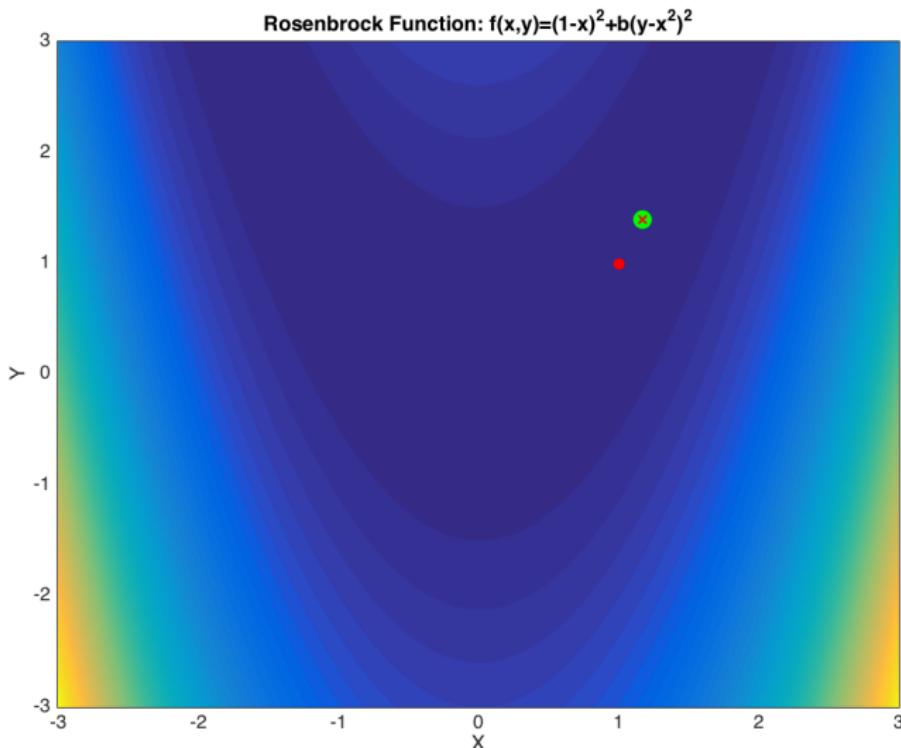
# DIFFERENTIAL EVOLUTION



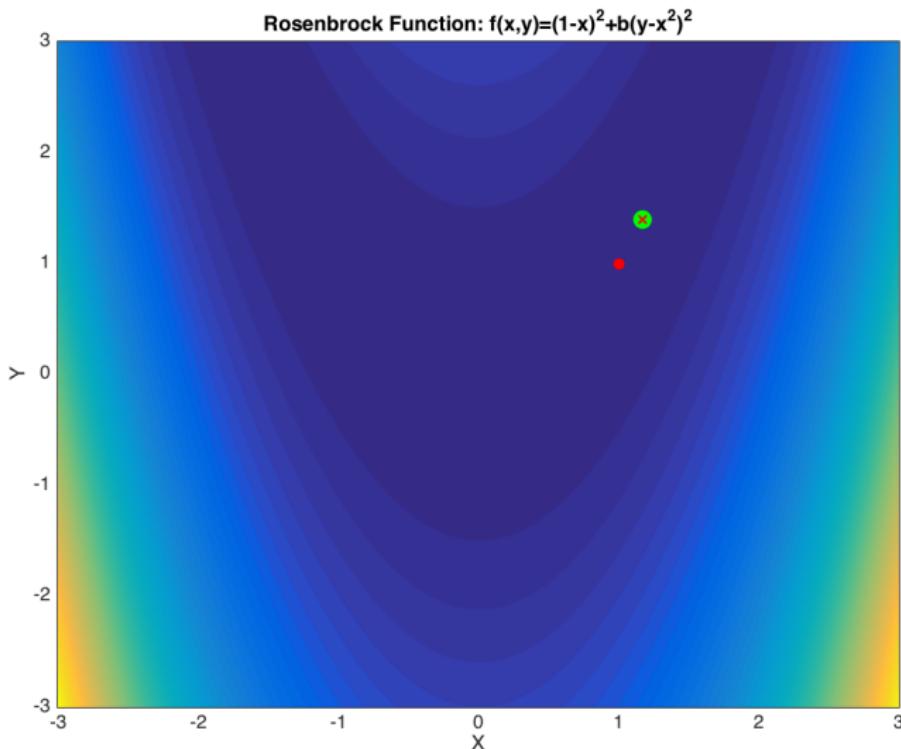
# DIFFERENTIAL EVOLUTION



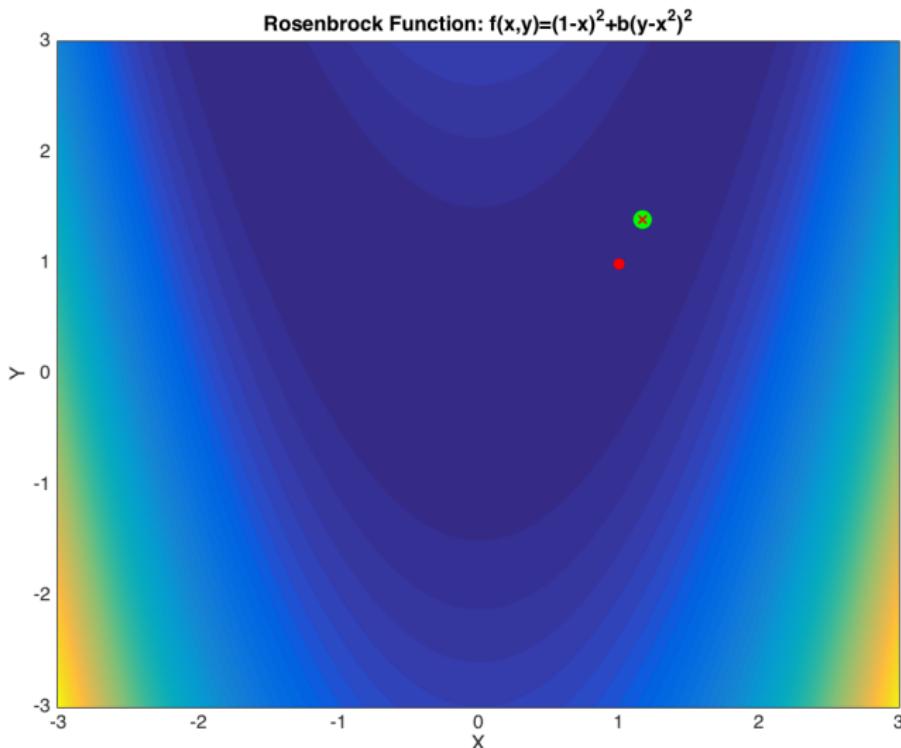
# DIFFERENTIAL EVOLUTION



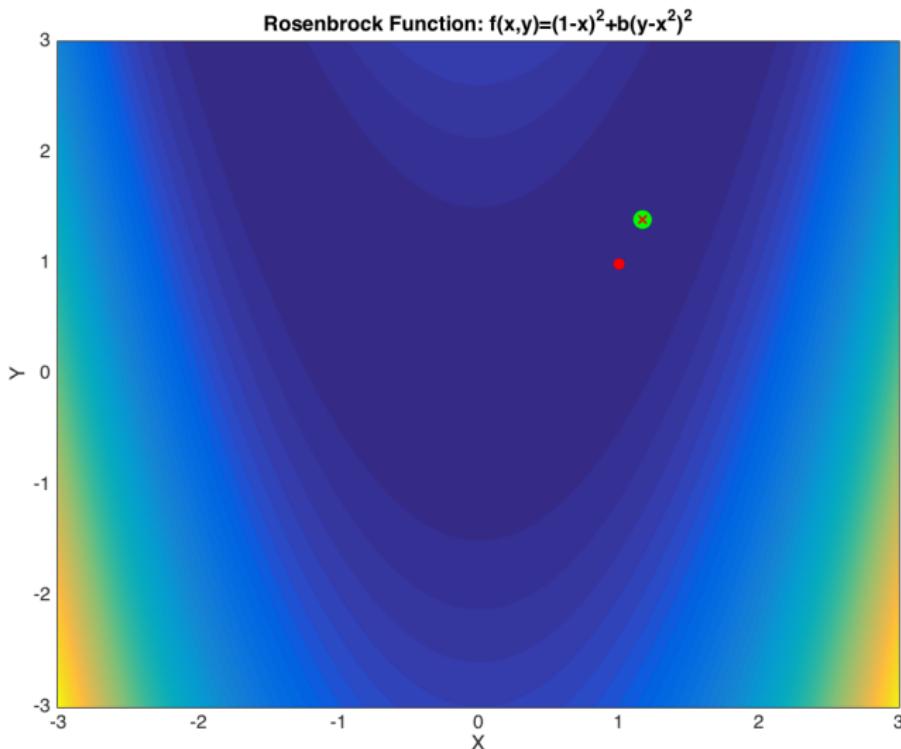
# DIFFERENTIAL EVOLUTION



# DIFFERENTIAL EVOLUTION



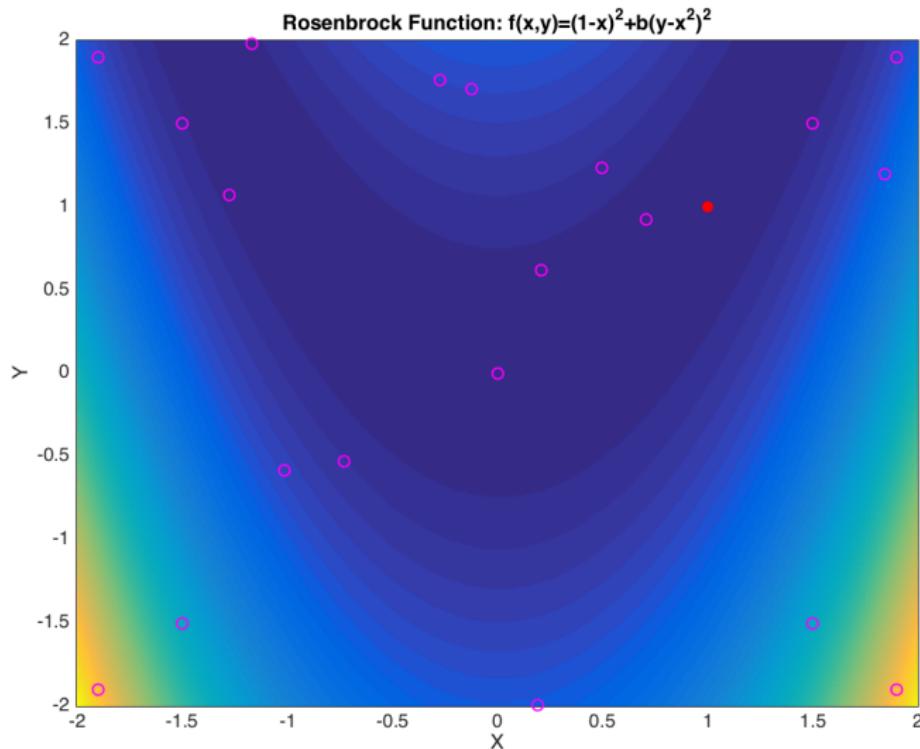
# DIFFERENTIAL EVOLUTION



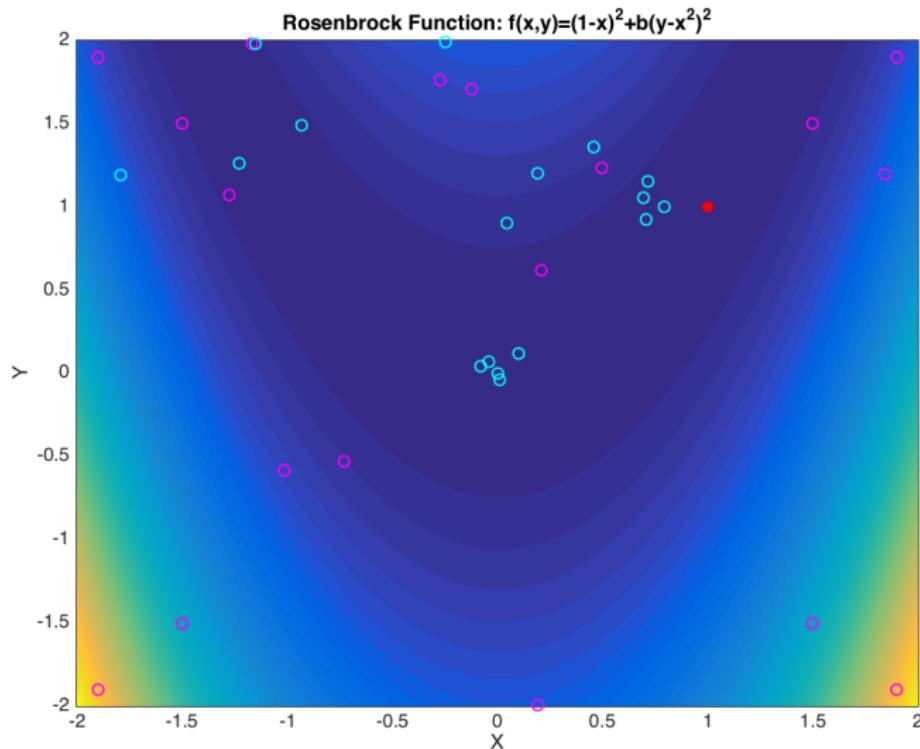
## GENETIC ALGORITHM/EVOLUTIONARY ALGORITHM

- ▶ Start with cloud of points
  - ▶ Take the most fit points and designate them as parents
  - ▶ Take these parents and “breed” them, find their offspring
  - ▶ Take the most fit points of the two populations (one option)
  - ▶ Repeat.
- ▶ Breeding commonly takes one of two steps
  - ▶ Mutation: take parents, and allow their “genes” to mutate
  - ▶ Crossover: take parents, for each vector randomly take  $x\%$  of one and  $(1-x\%)$  of other.

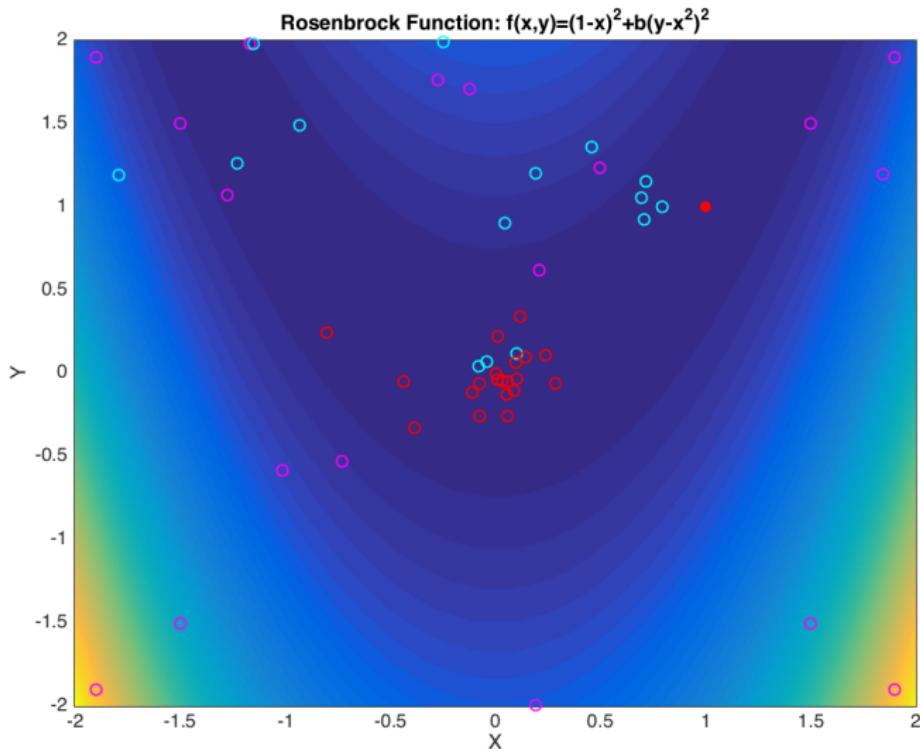
# GENETIC ALGORITHM



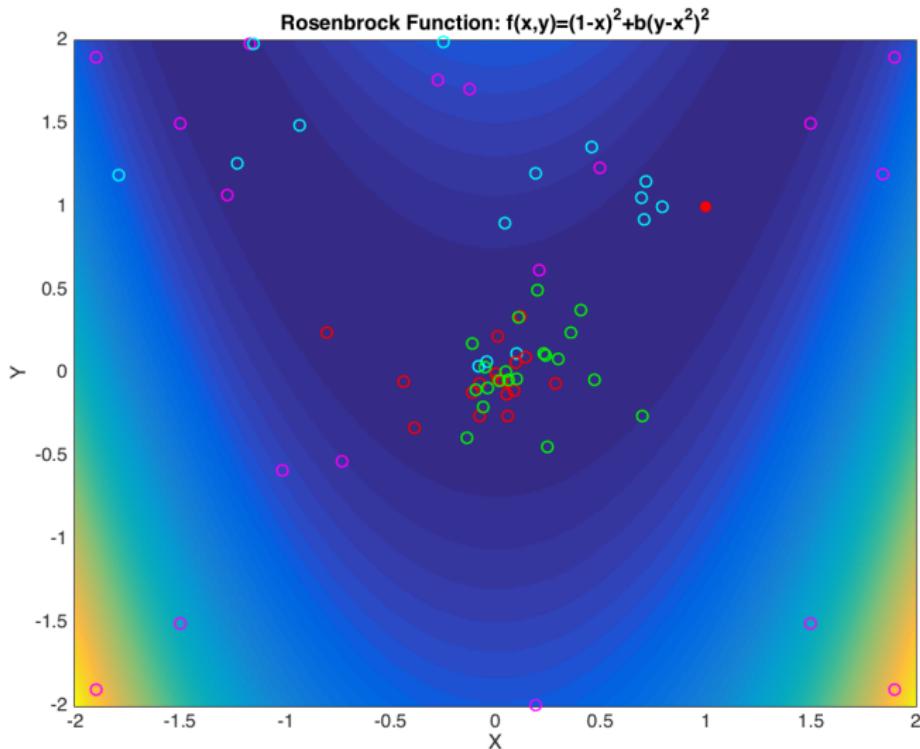
# GENETIC ALGORITHM



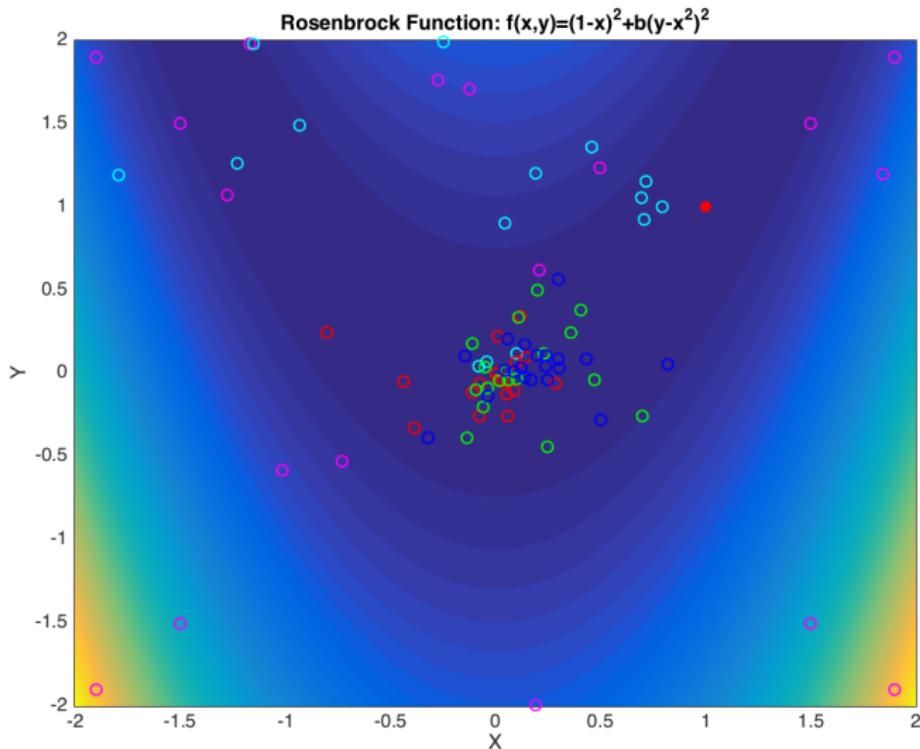
# GENETIC ALGORITHM



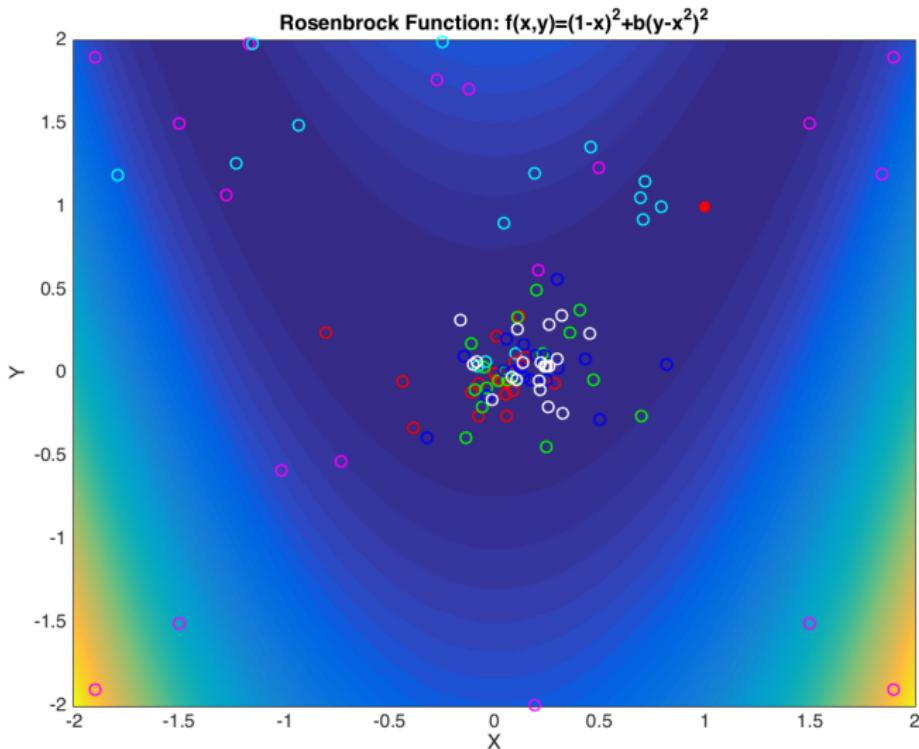
# GENETIC ALGORITHM



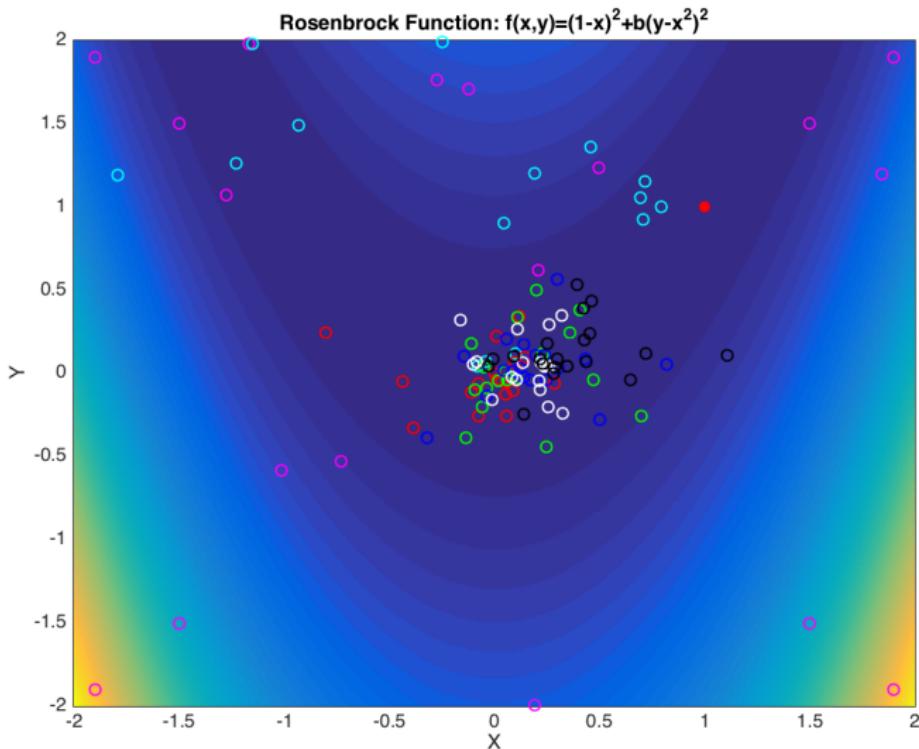
# GENETIC ALGORITHM



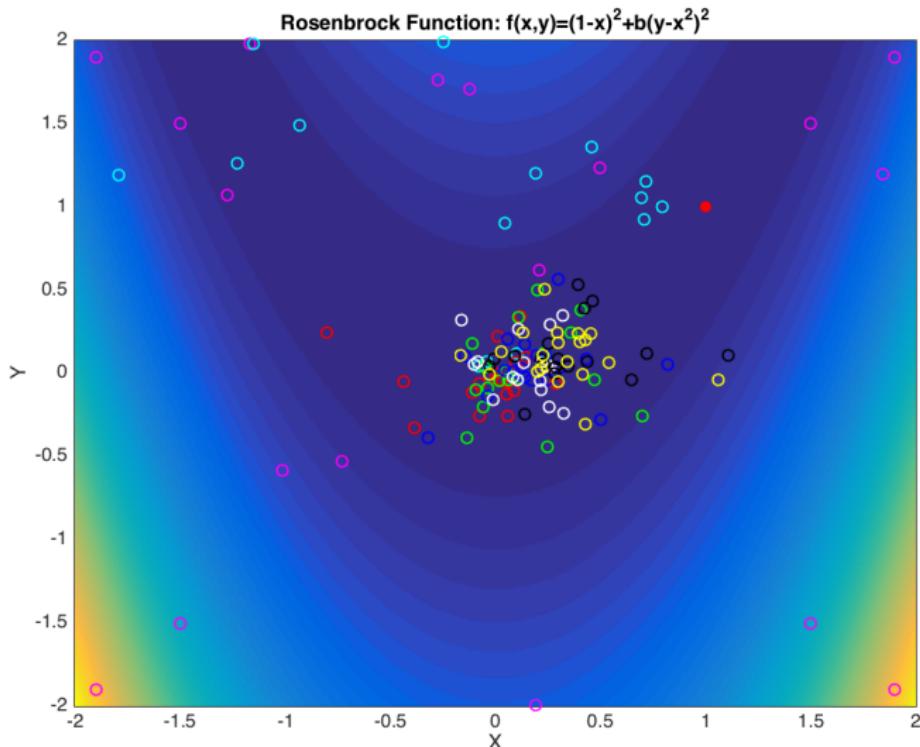
# GENETIC ALGORITHM



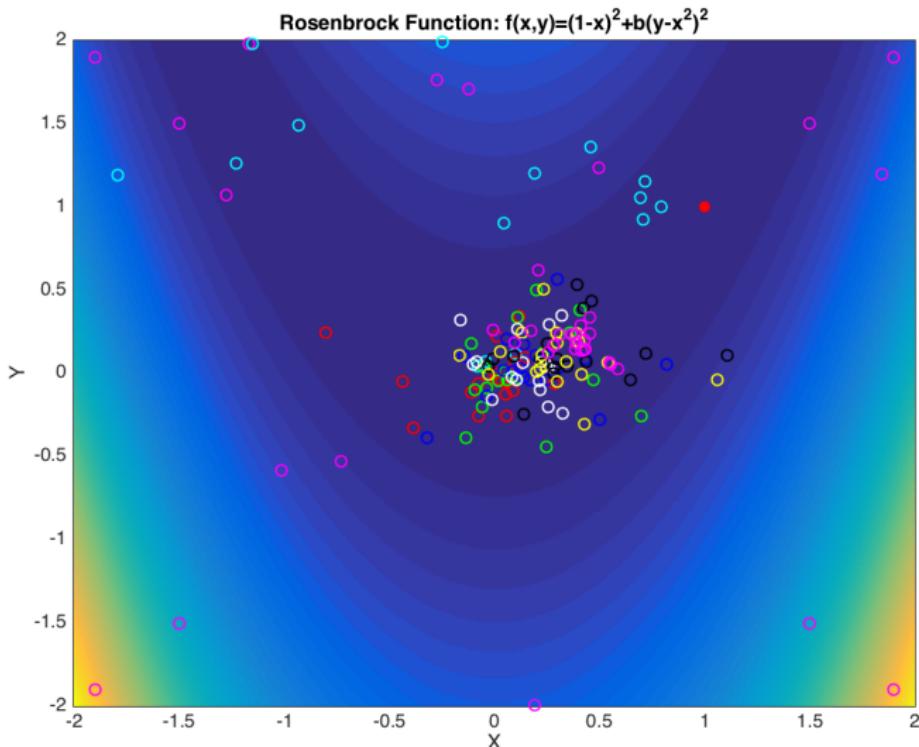
# GENETIC ALGORITHM



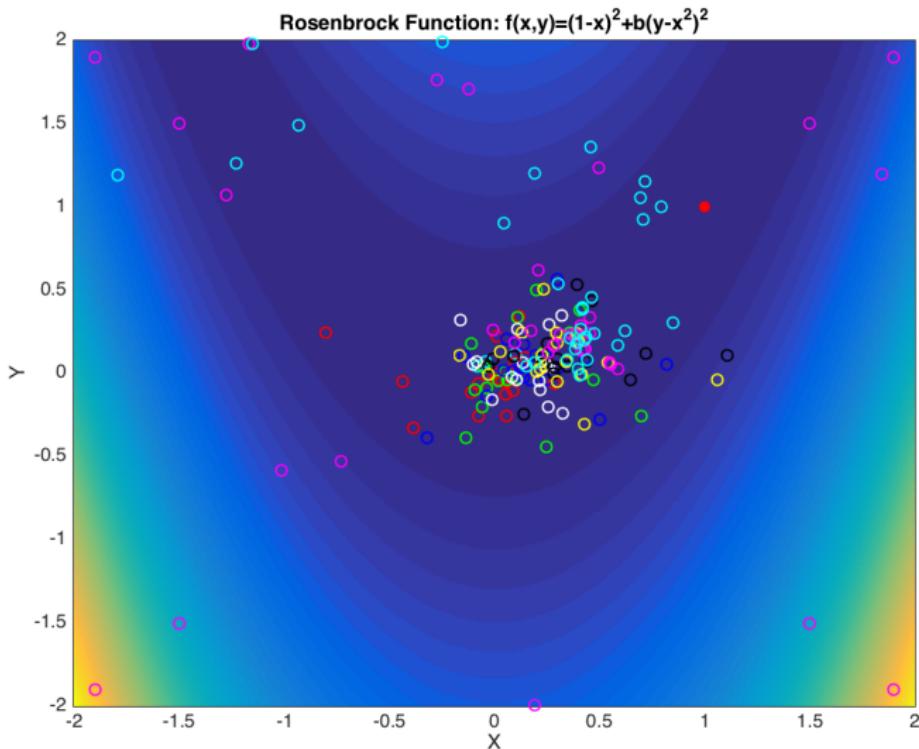
# GENETIC ALGORITHM



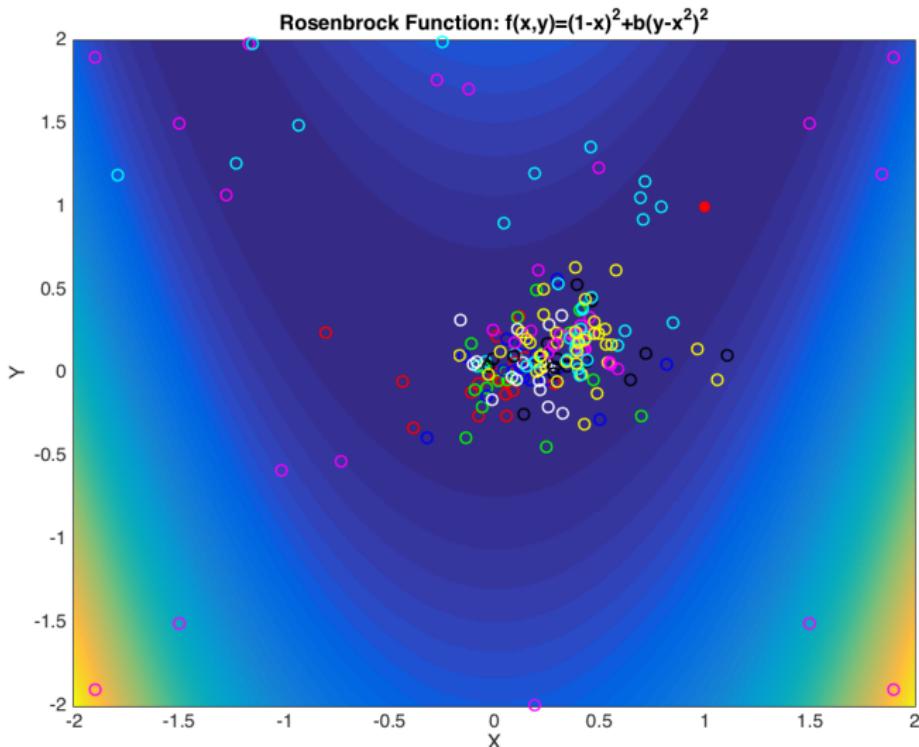
# GENETIC ALGORITHM



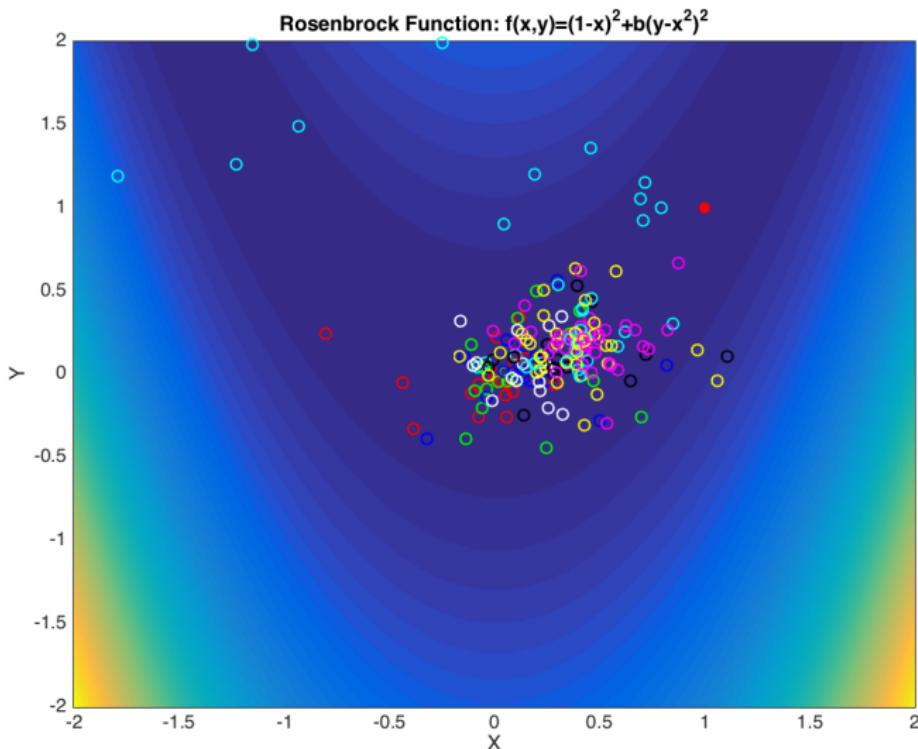
# GENETIC ALGORITHM



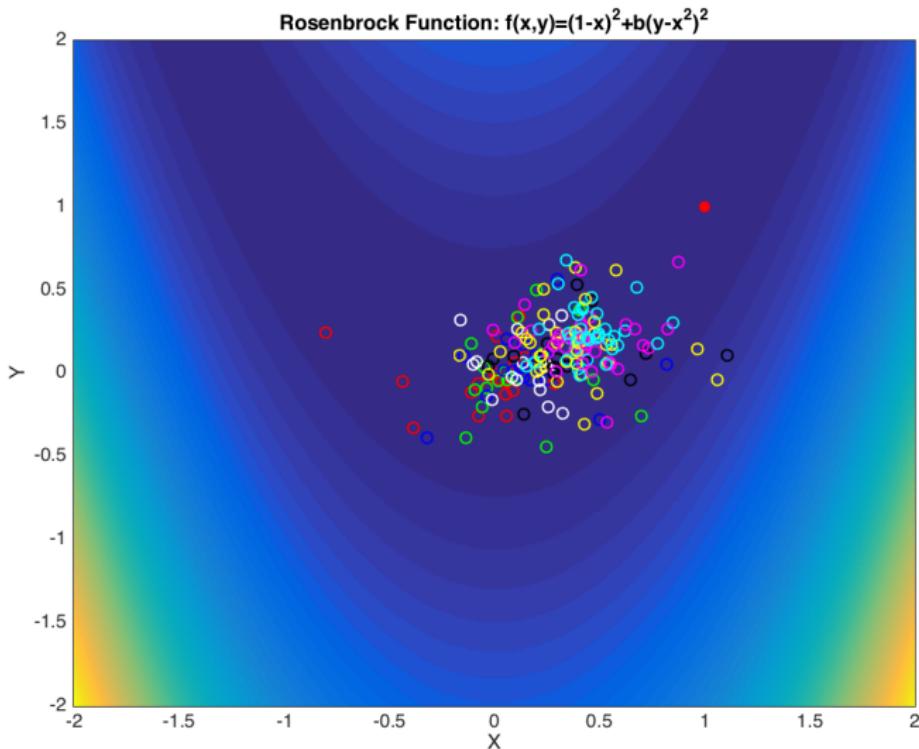
# GENETIC ALGORITHM



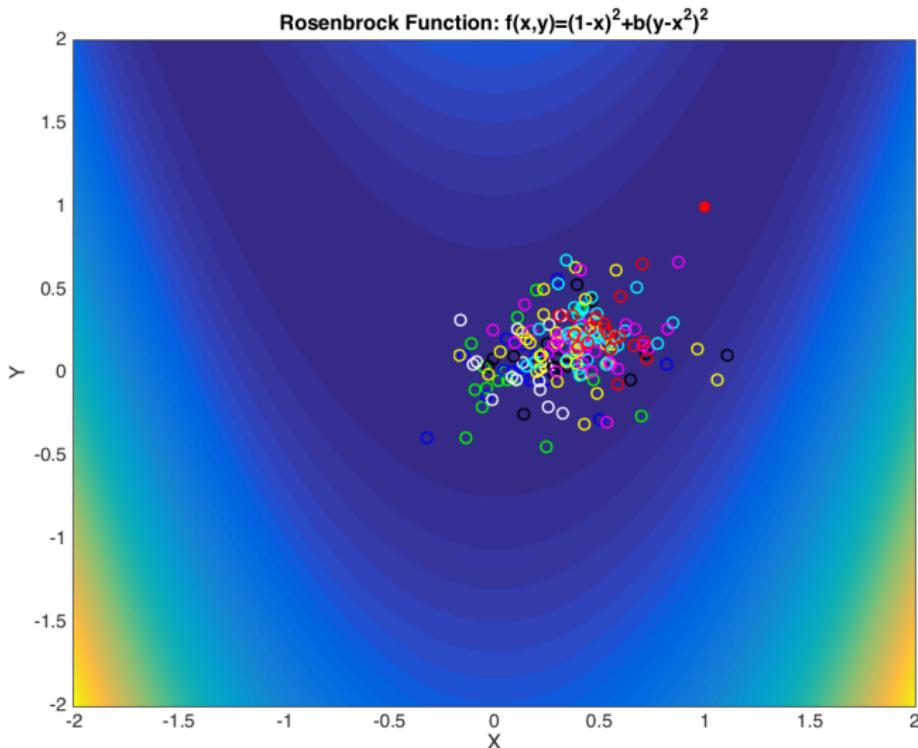
# GENETIC ALGORITHM



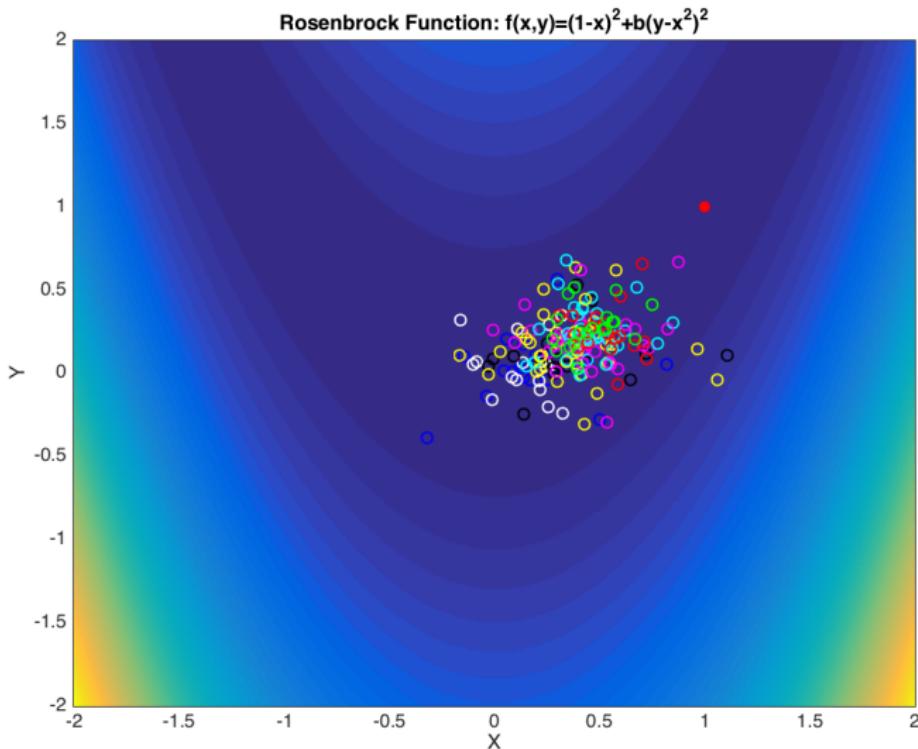
# GENETIC ALGORITHM



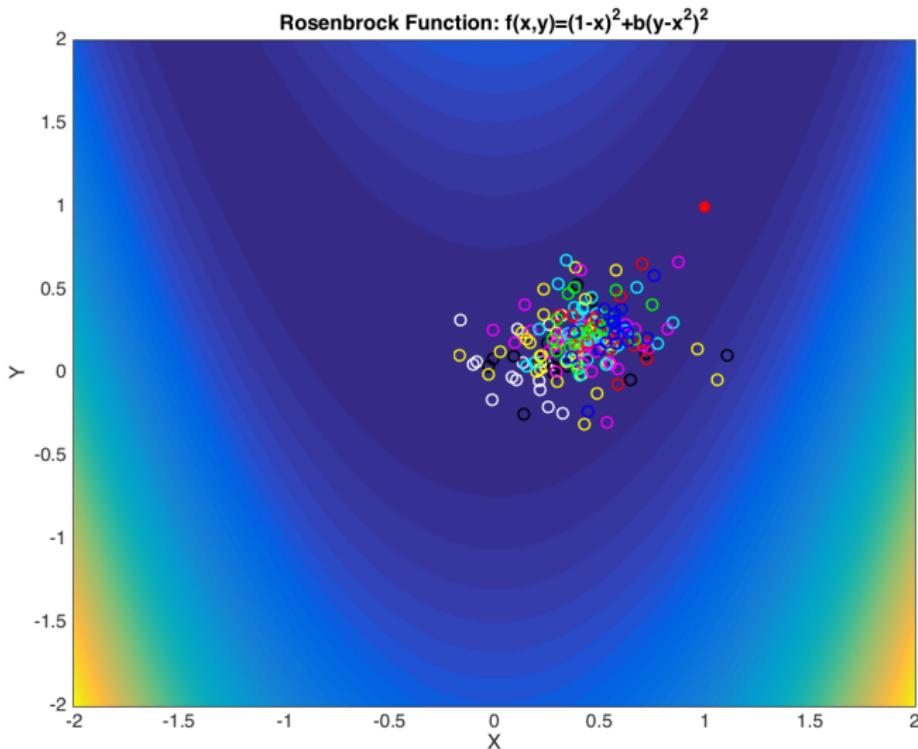
# GENETIC ALGORITHM



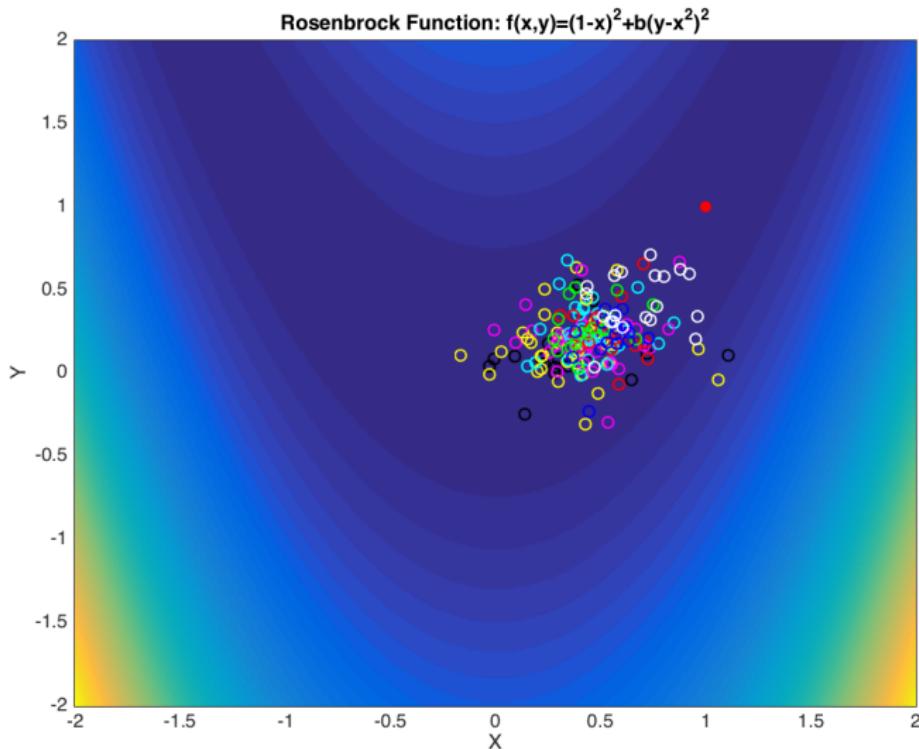
# GENETIC ALGORITHM



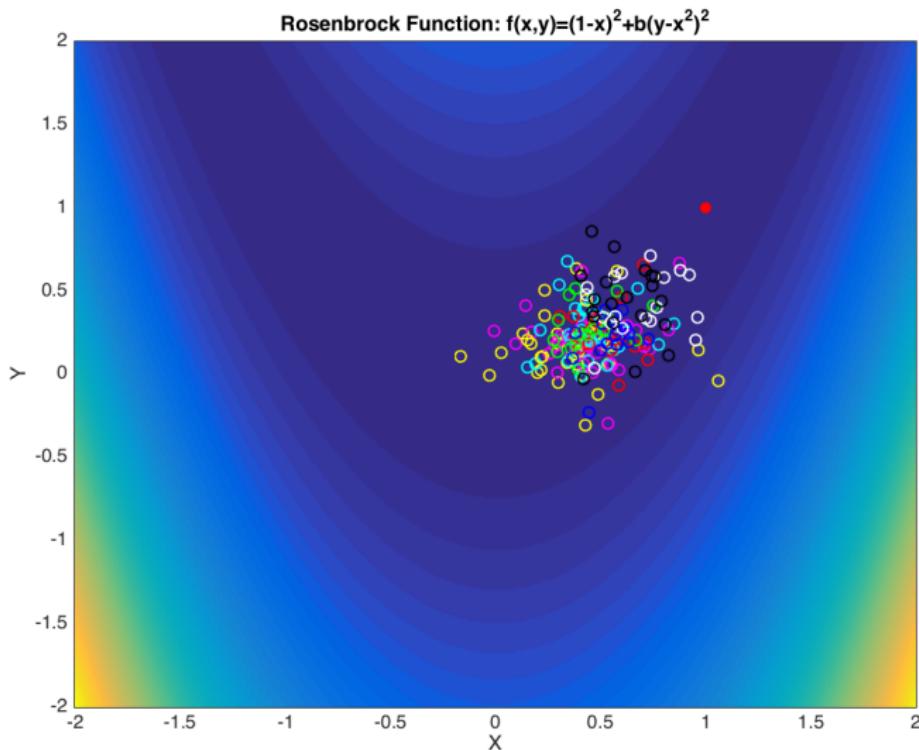
# GENETIC ALGORITHM



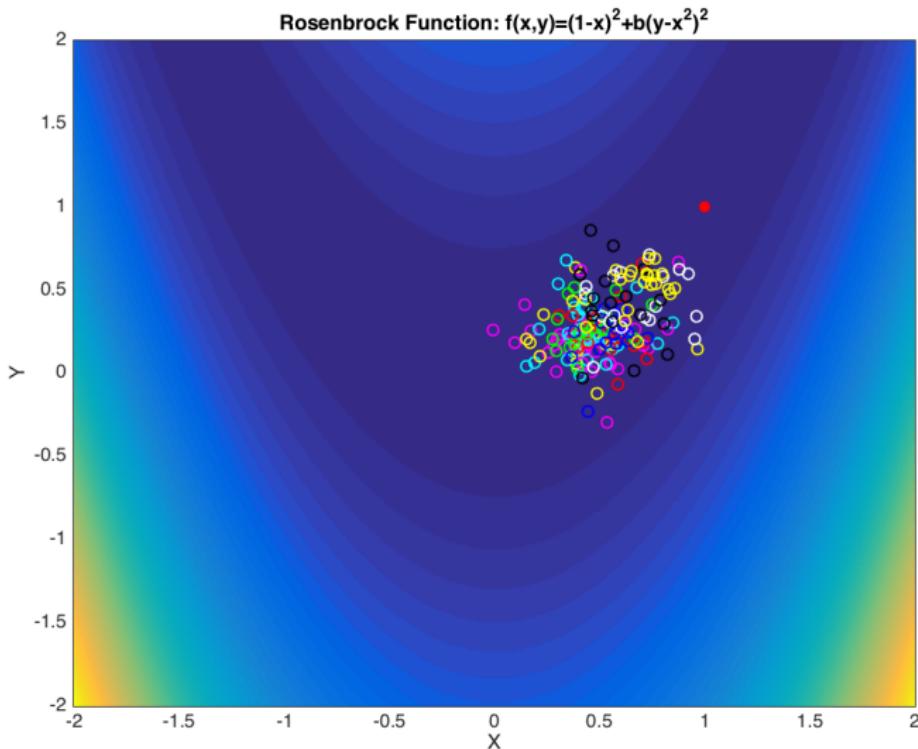
# GENETIC ALGORITHM



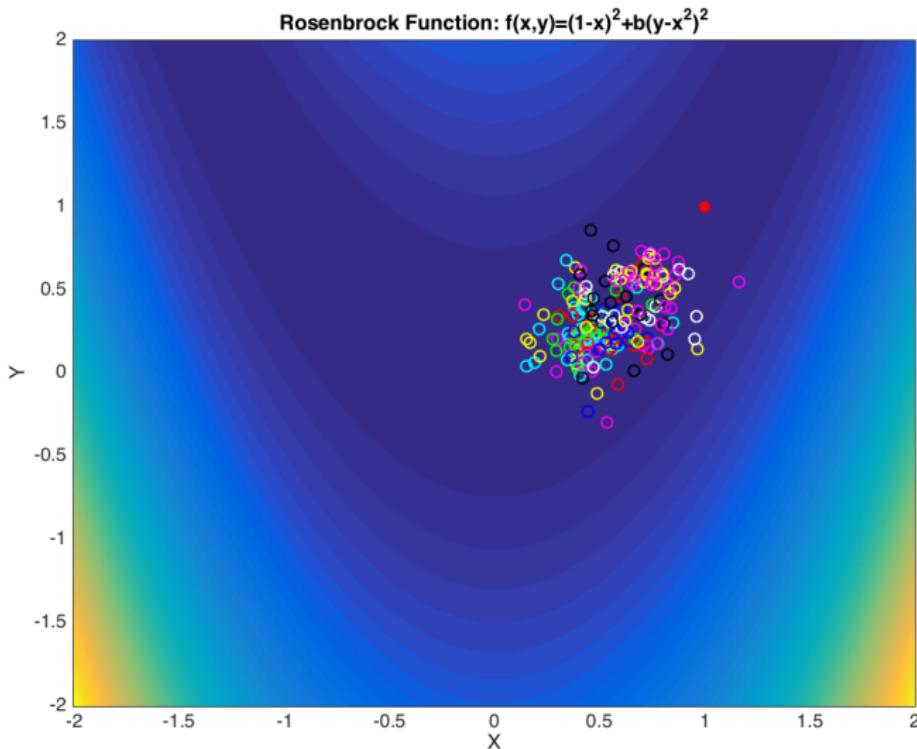
# GENETIC ALGORITHM



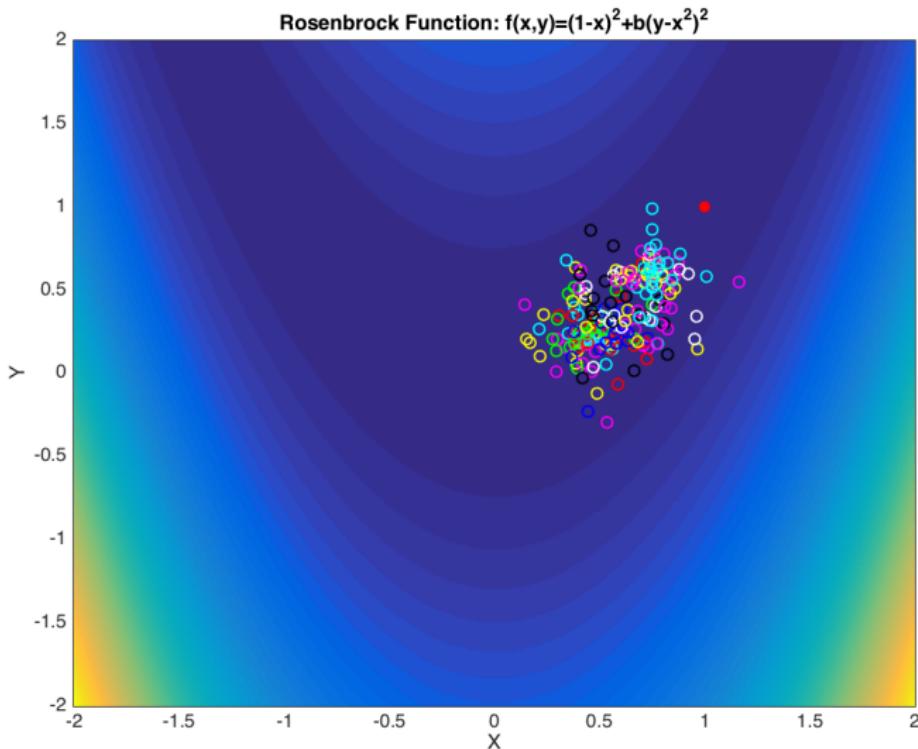
# GENETIC ALGORITHM



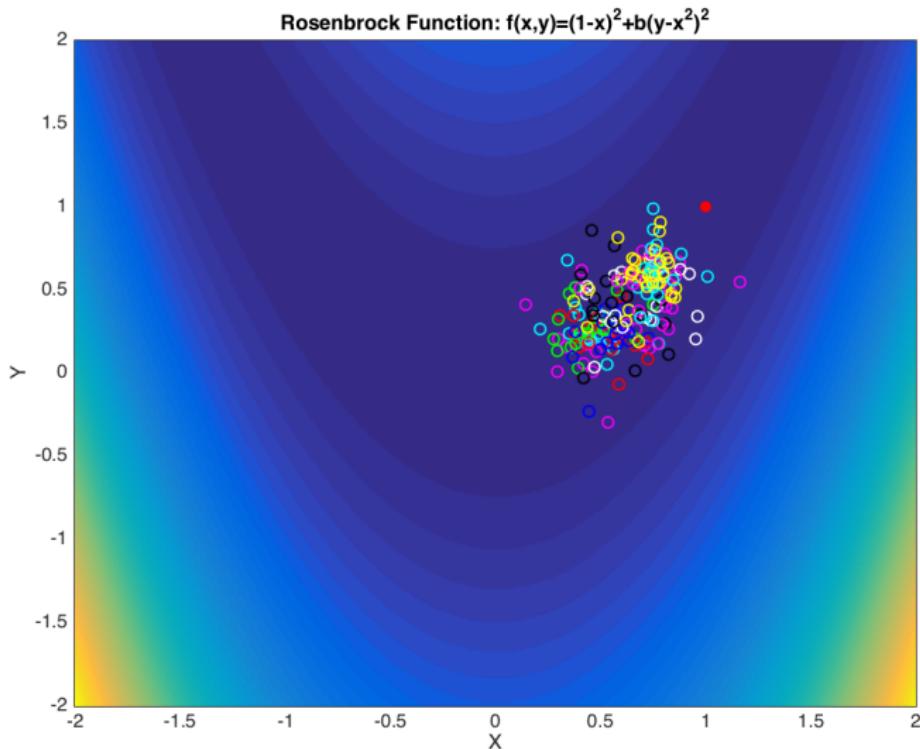
# GENETIC ALGORITHM



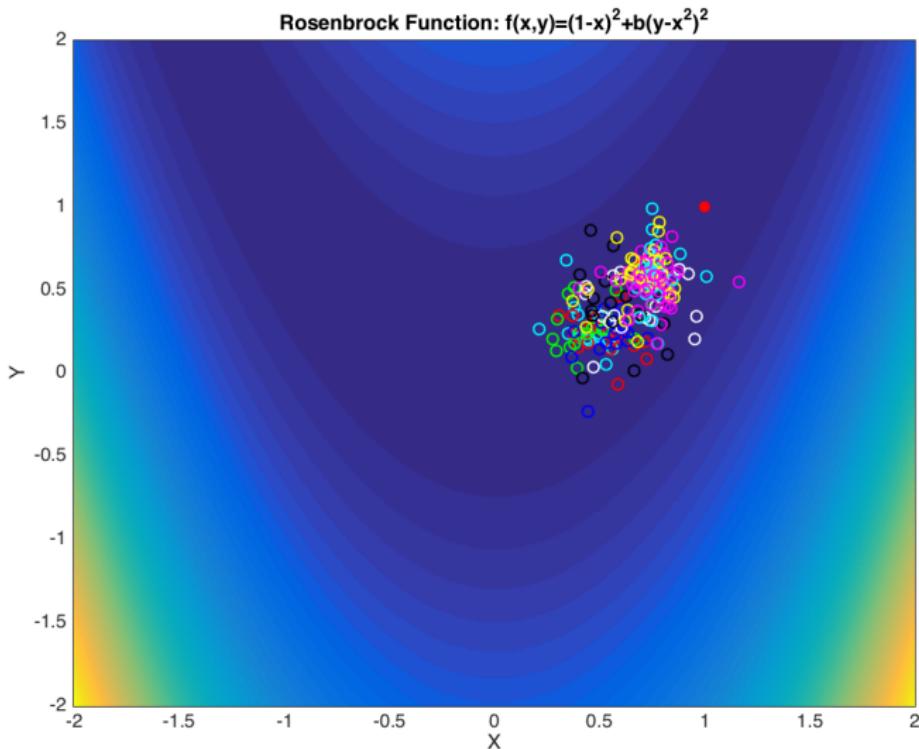
# GENETIC ALGORITHM



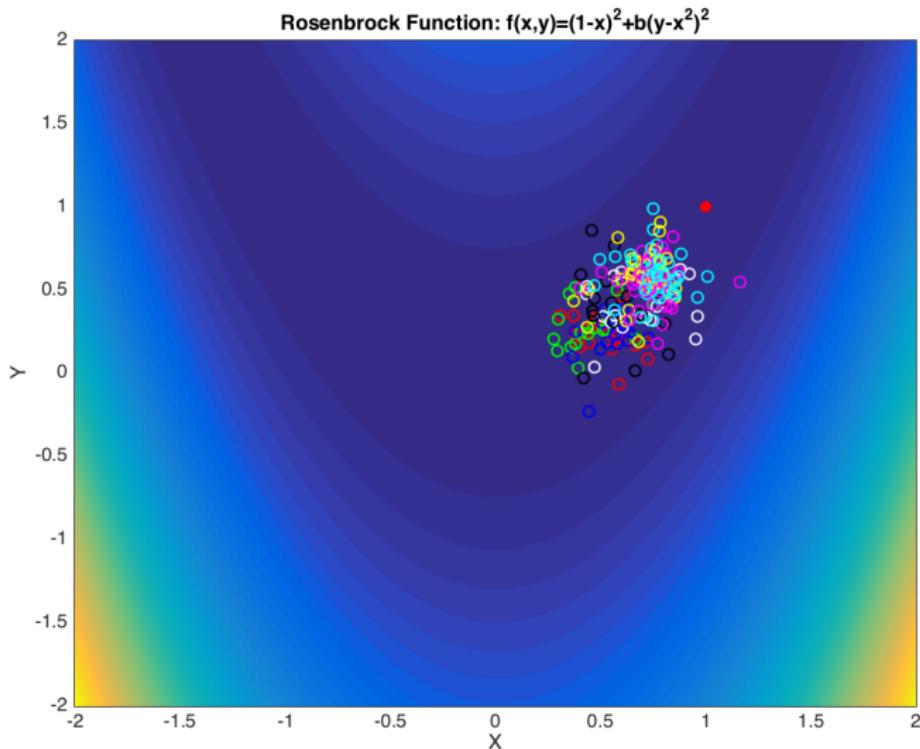
# GENETIC ALGORITHM



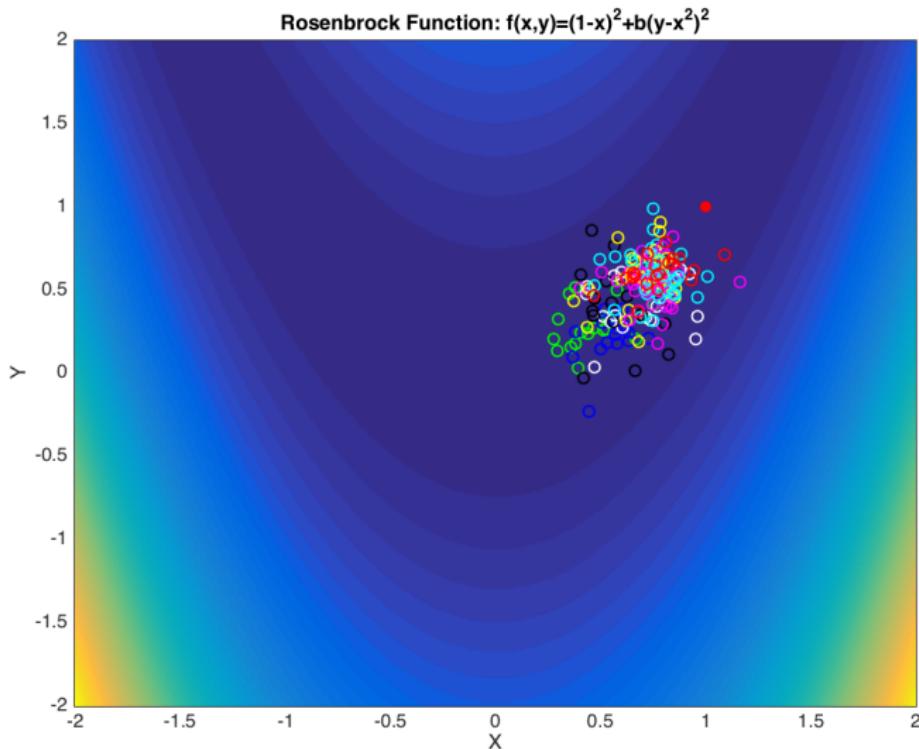
# GENETIC ALGORITHM



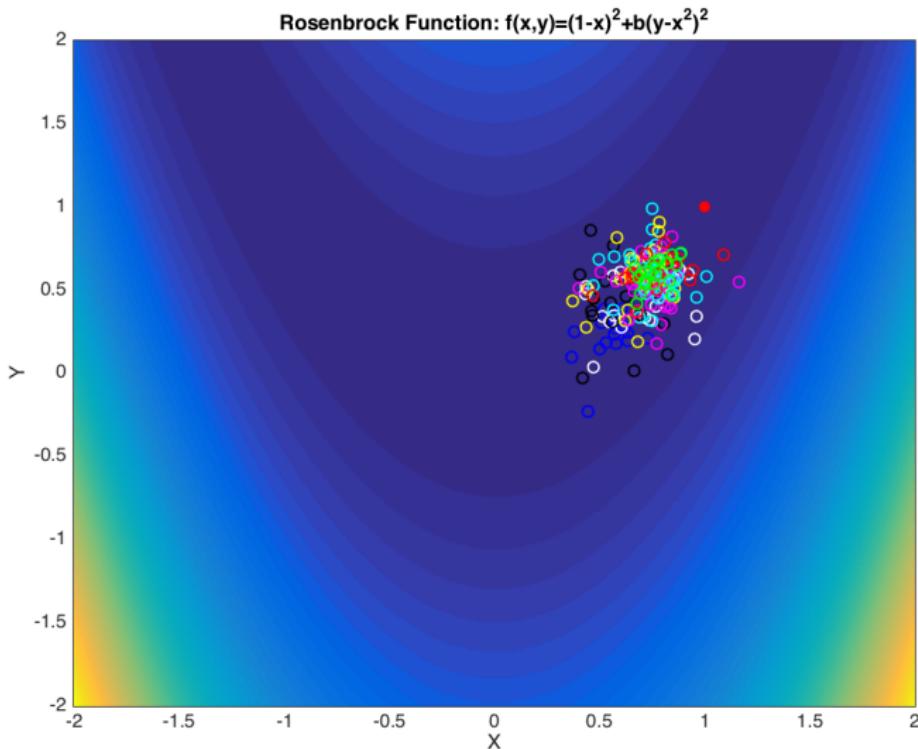
# GENETIC ALGORITHM



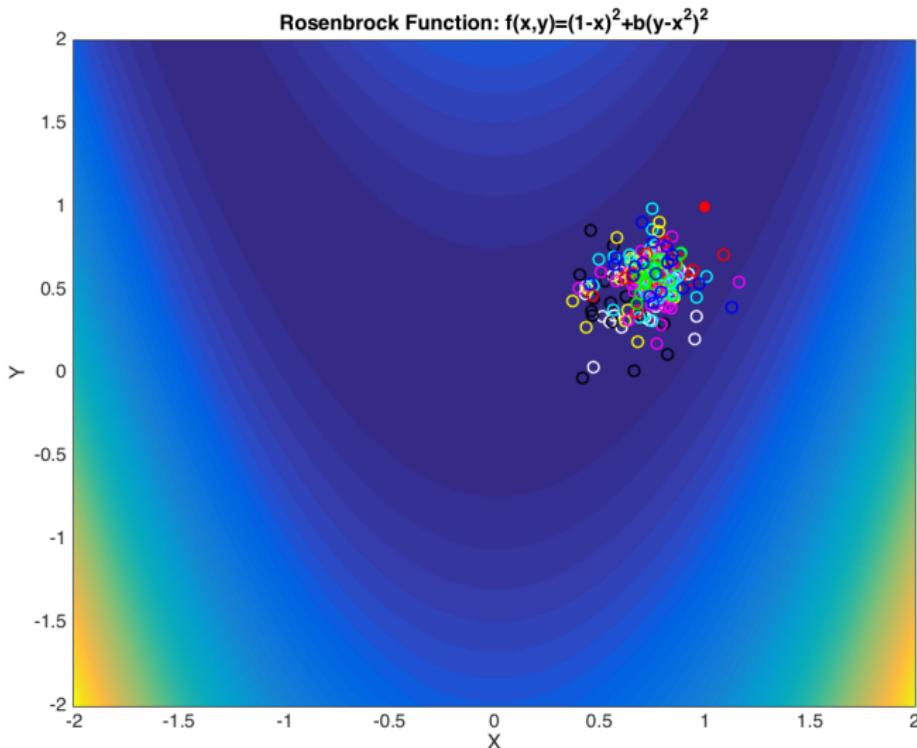
# GENETIC ALGORITHM



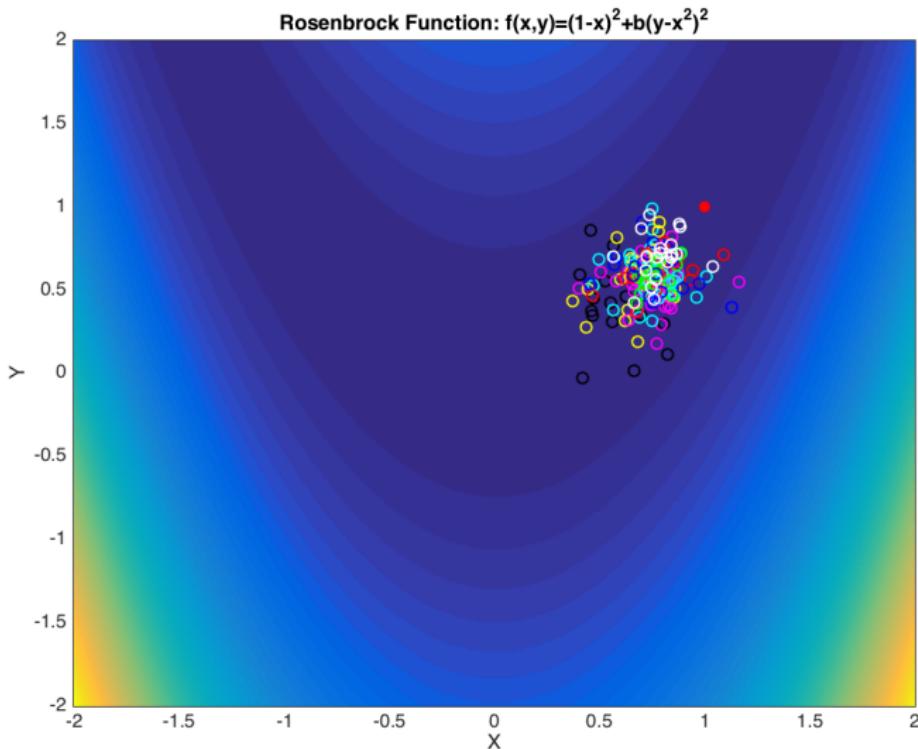
# GENETIC ALGORITHM



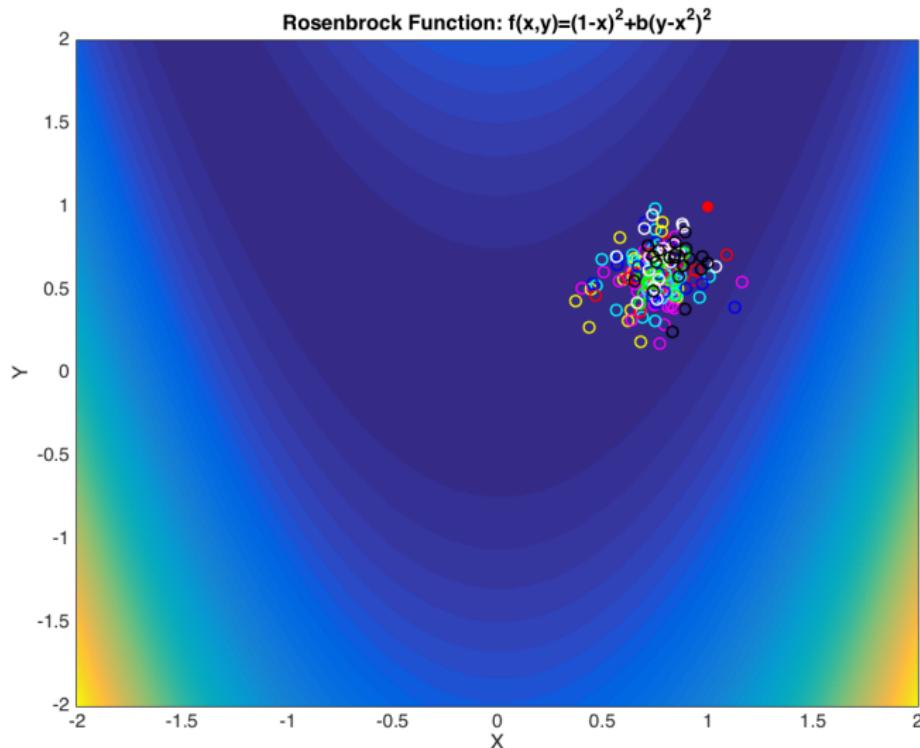
# GENETIC ALGORITHM



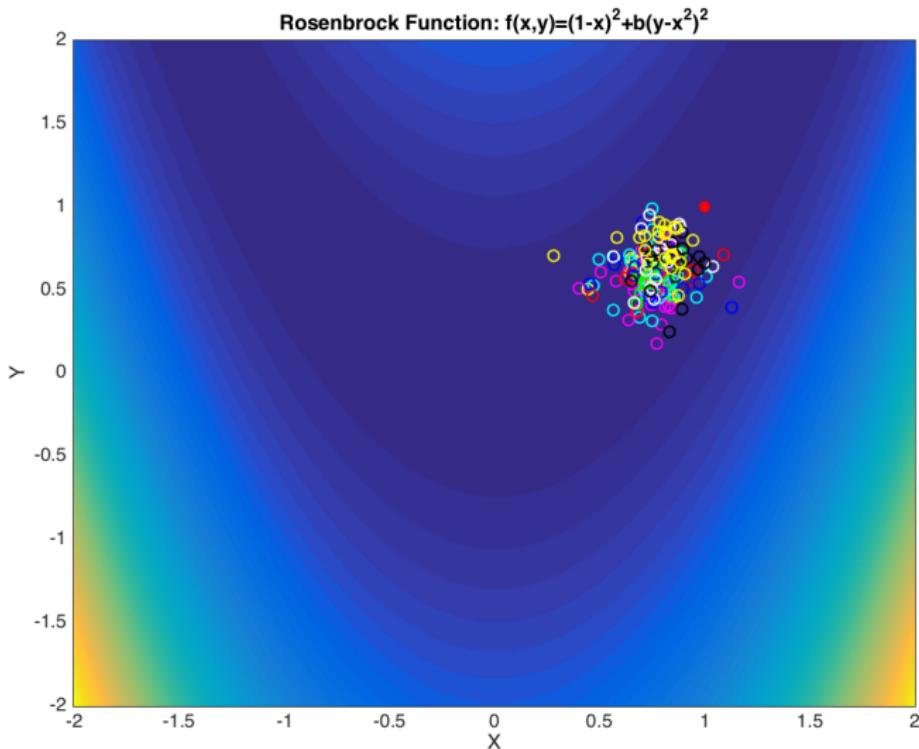
# GENETIC ALGORITHM



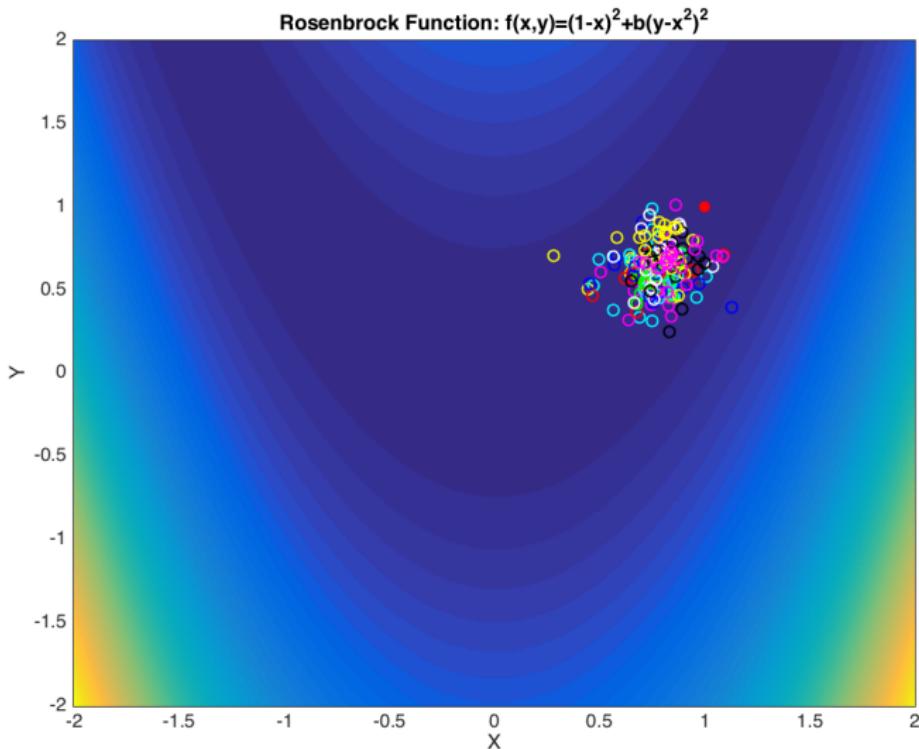
# GENETIC ALGORITHM



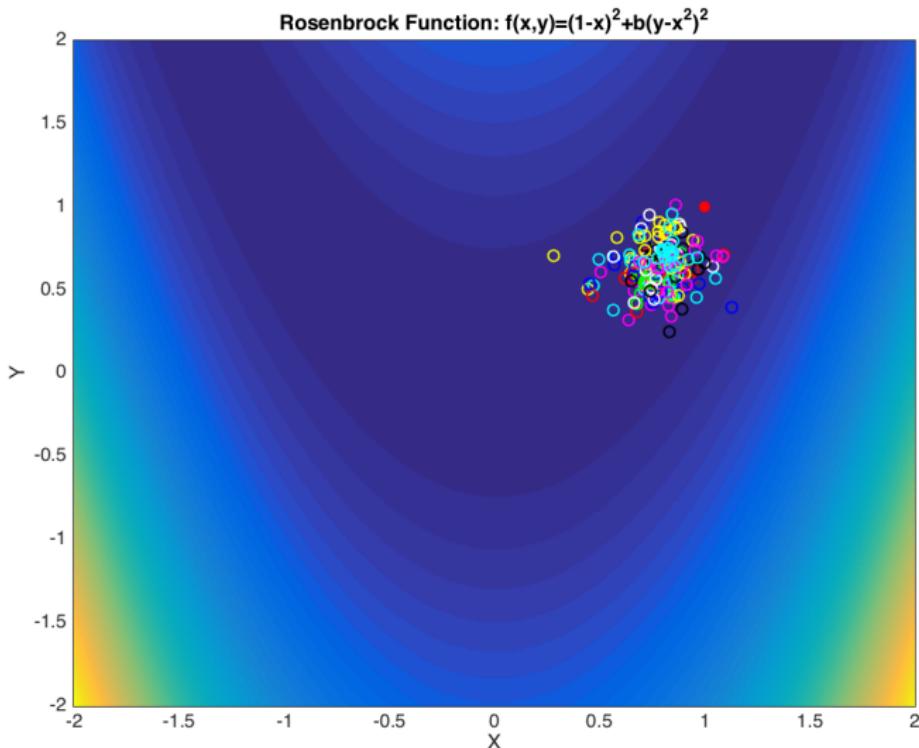
# GENETIC ALGORITHM



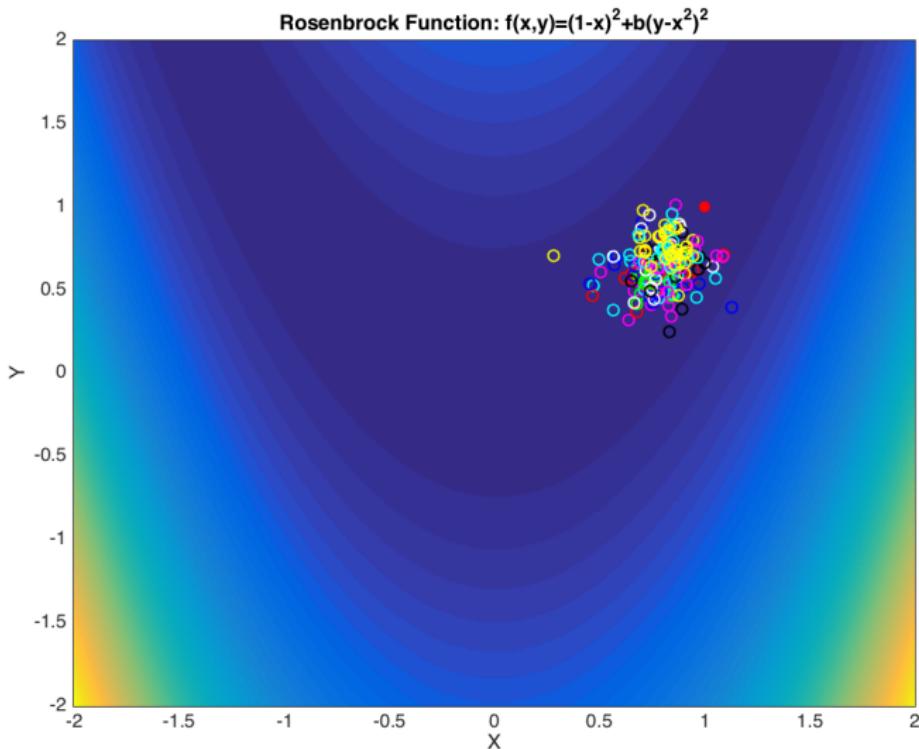
# GENETIC ALGORITHM



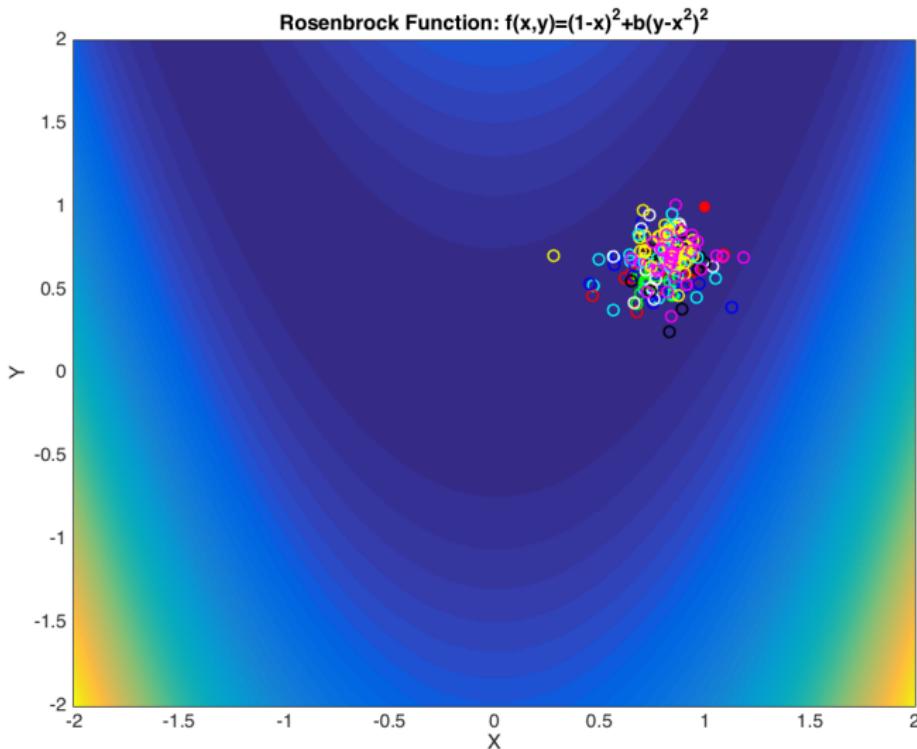
# GENETIC ALGORITHM



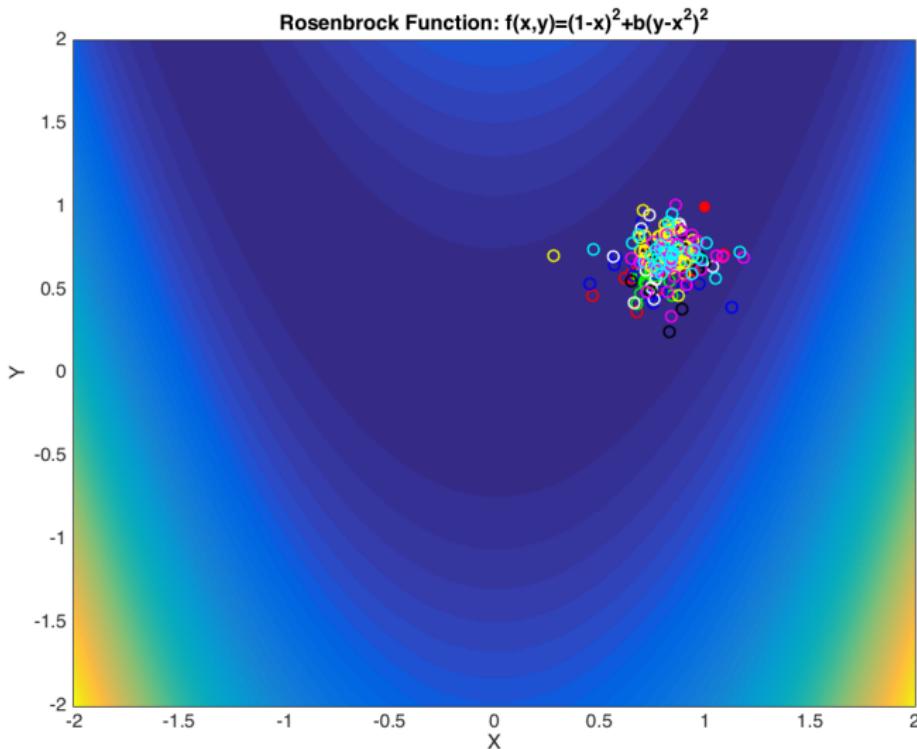
# GENETIC ALGORITHM



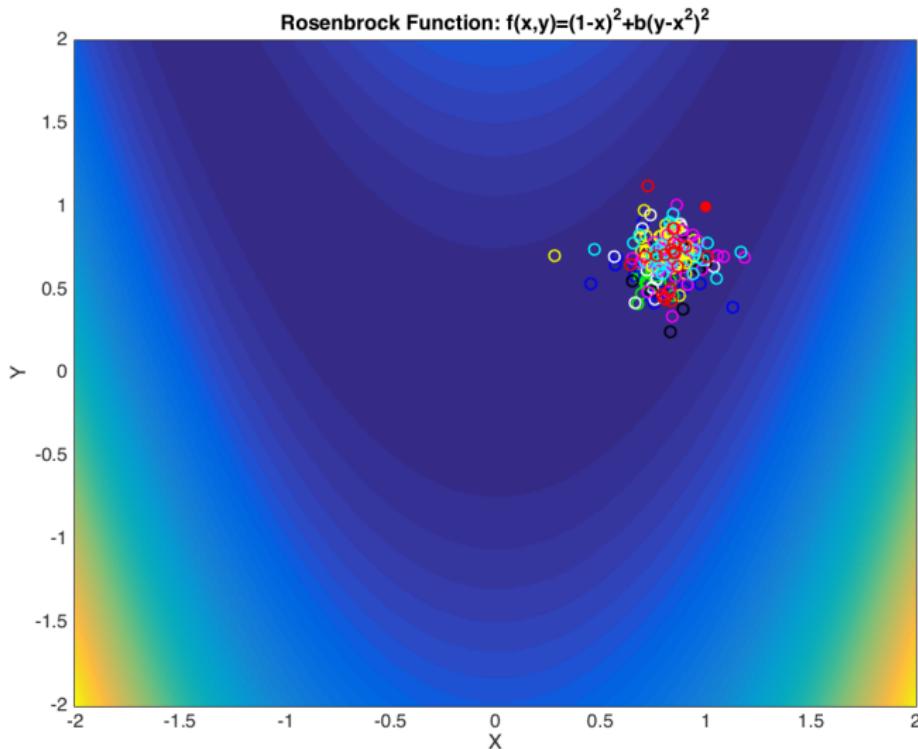
# GENETIC ALGORITHM



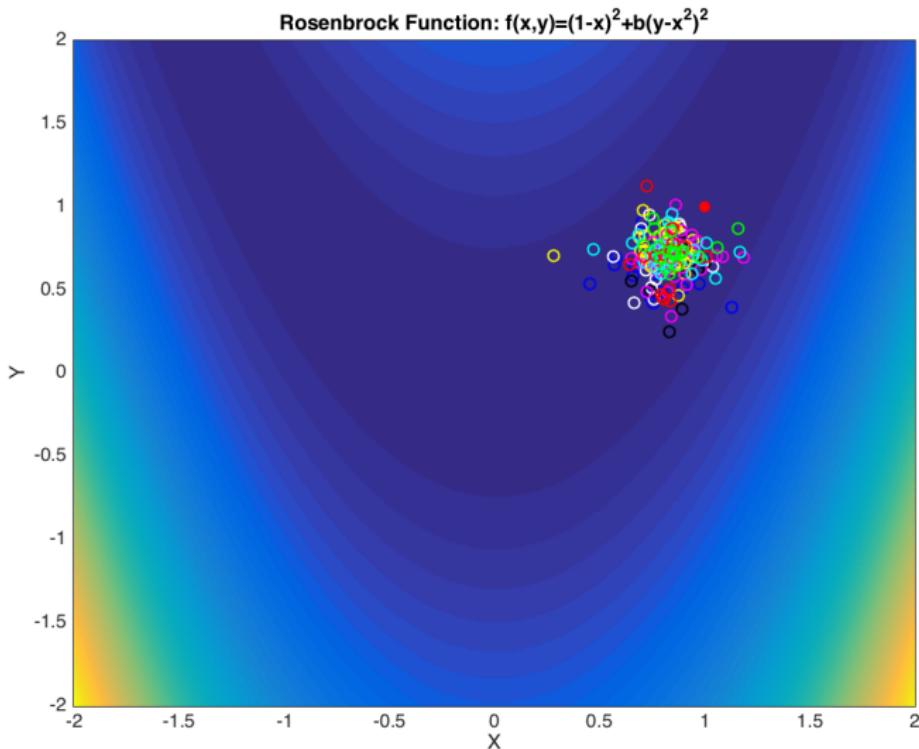
# GENETIC ALGORITHM



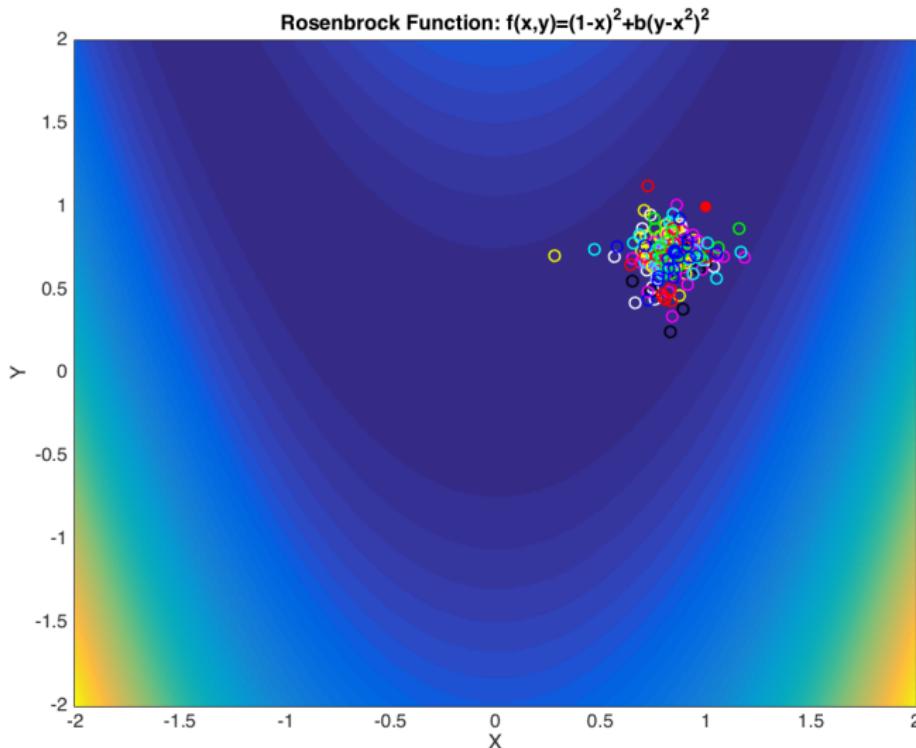
# GENETIC ALGORITHM



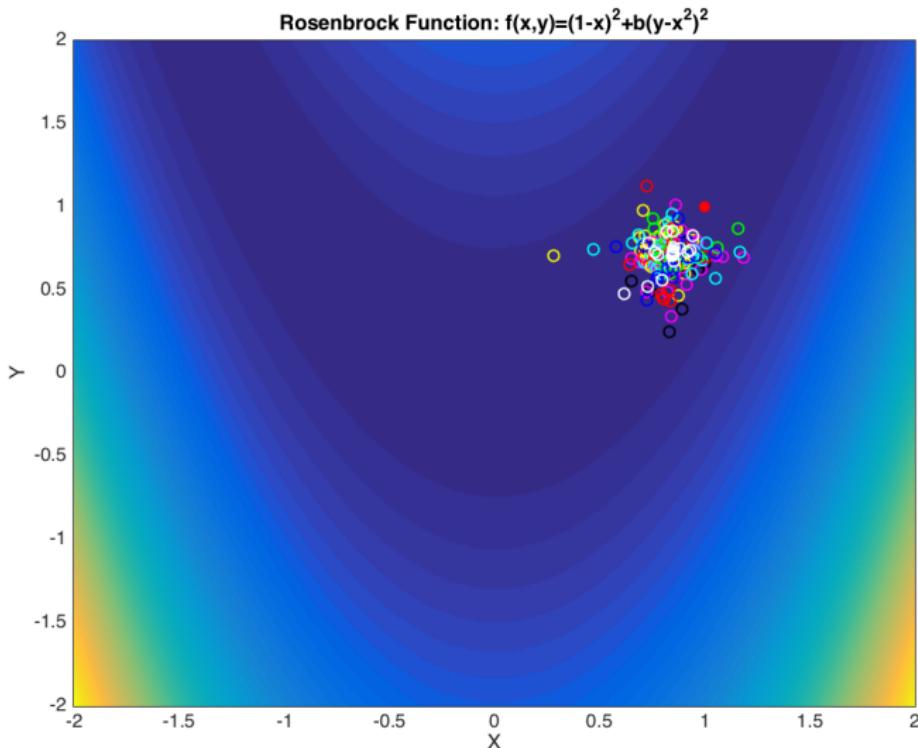
# GENETIC ALGORITHM



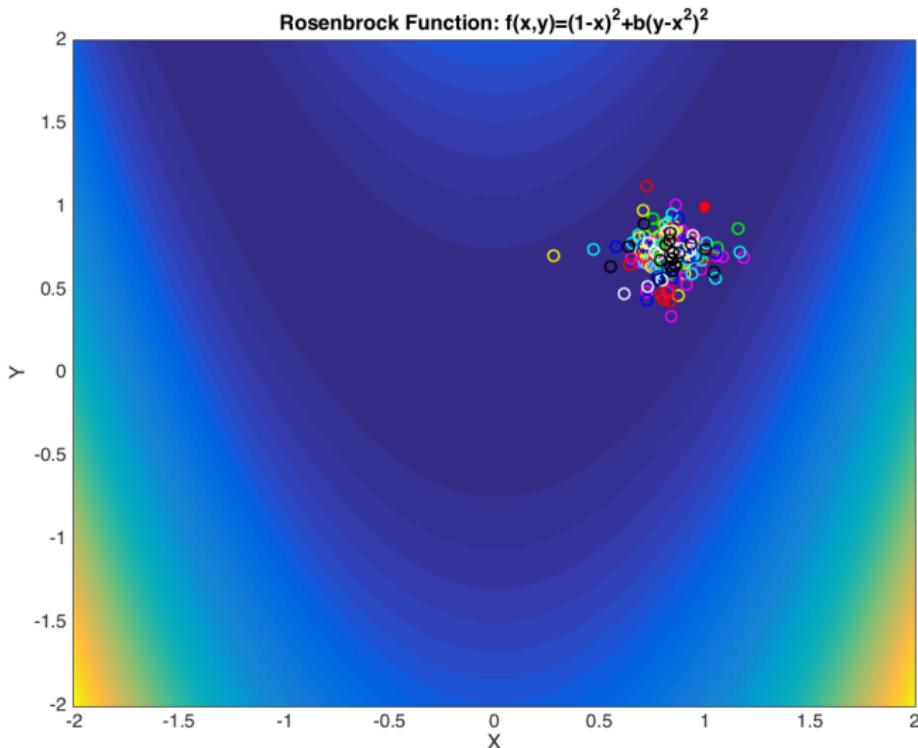
# GENETIC ALGORITHM



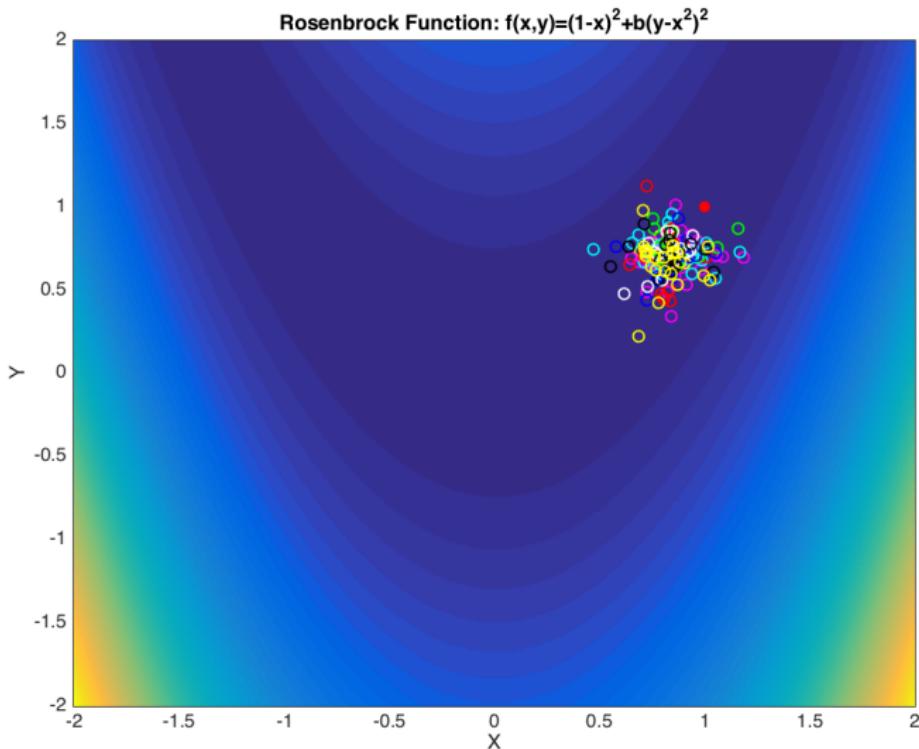
# GENETIC ALGORITHM



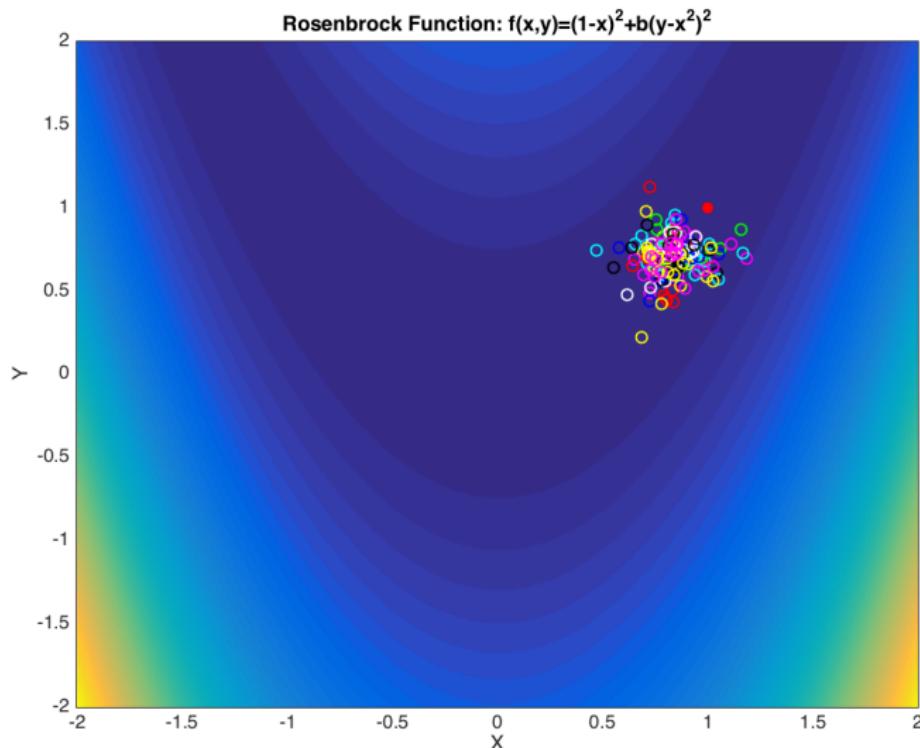
# GENETIC ALGORITHM



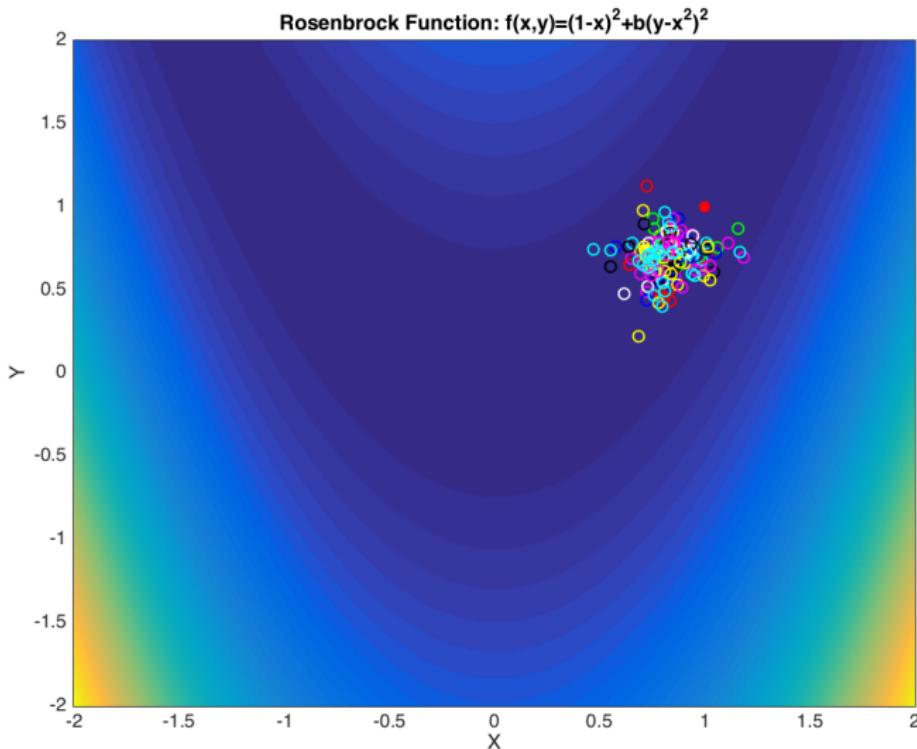
# GENETIC ALGORITHM



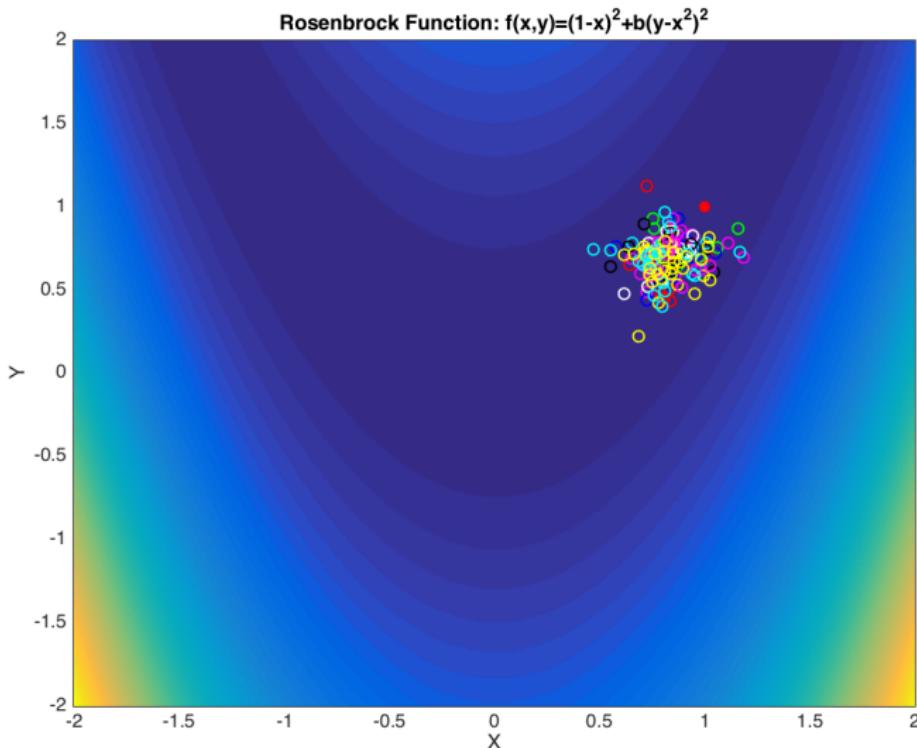
# GENETIC ALGORITHM



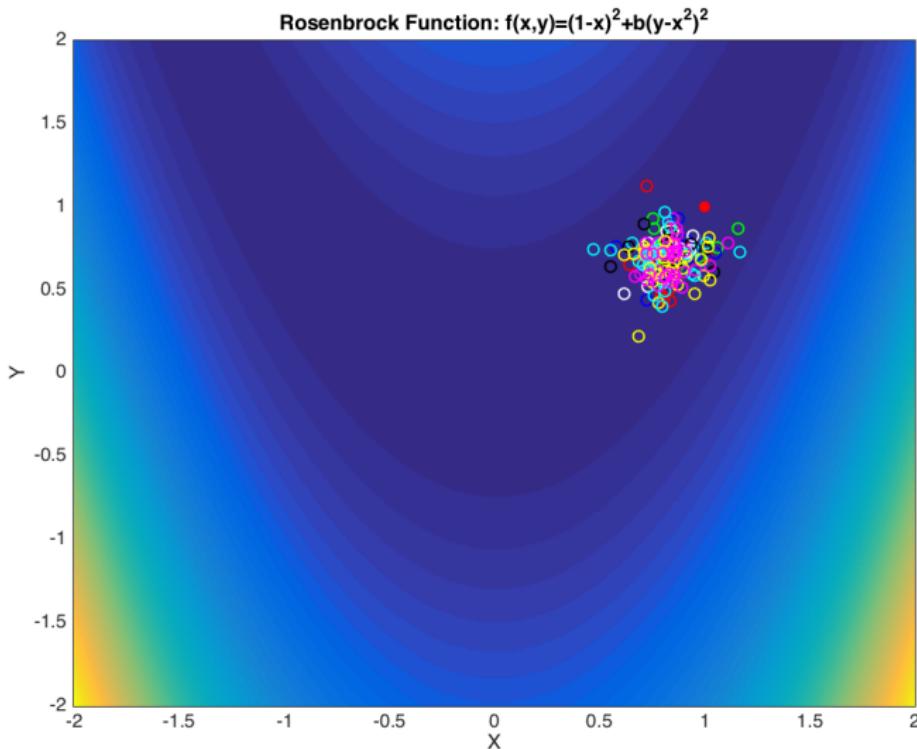
# GENETIC ALGORITHM



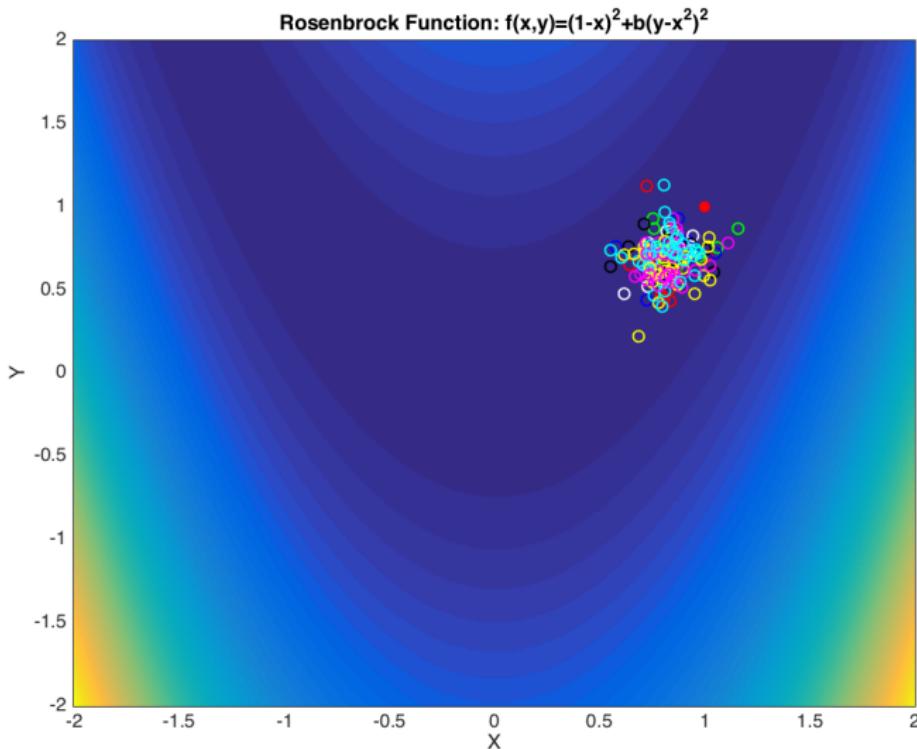
# GENETIC ALGORITHM



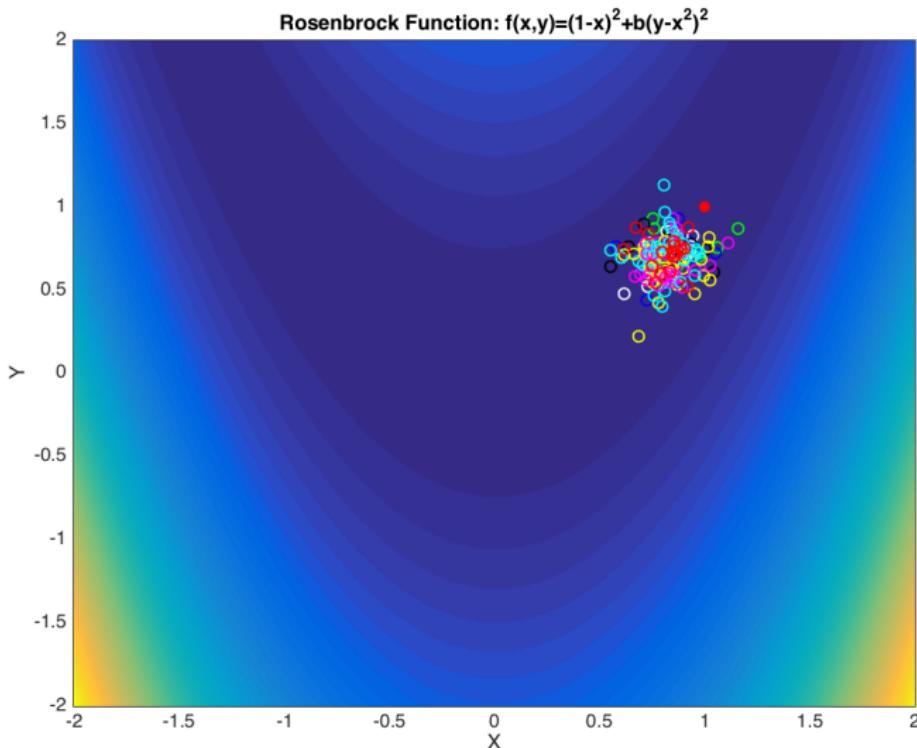
# GENETIC ALGORITHM



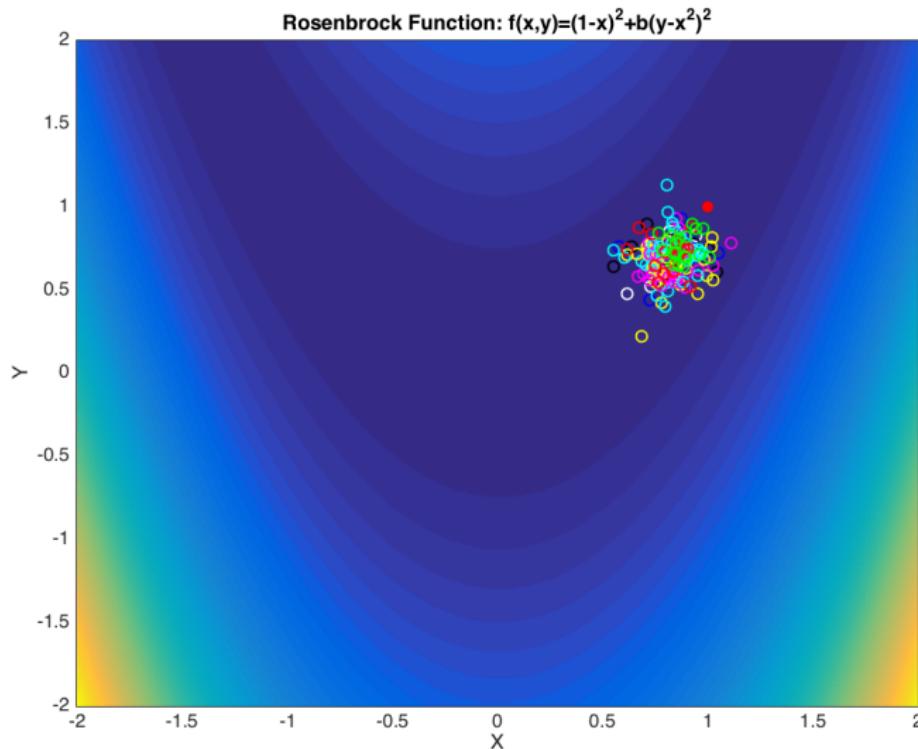
# GENETIC ALGORITHM



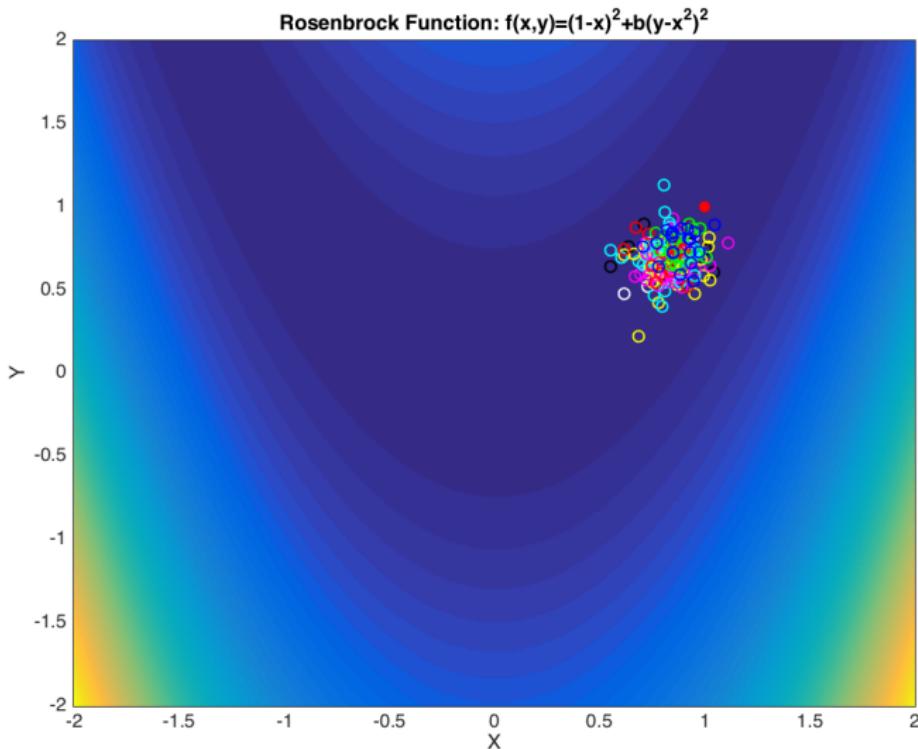
# GENETIC ALGORITHM



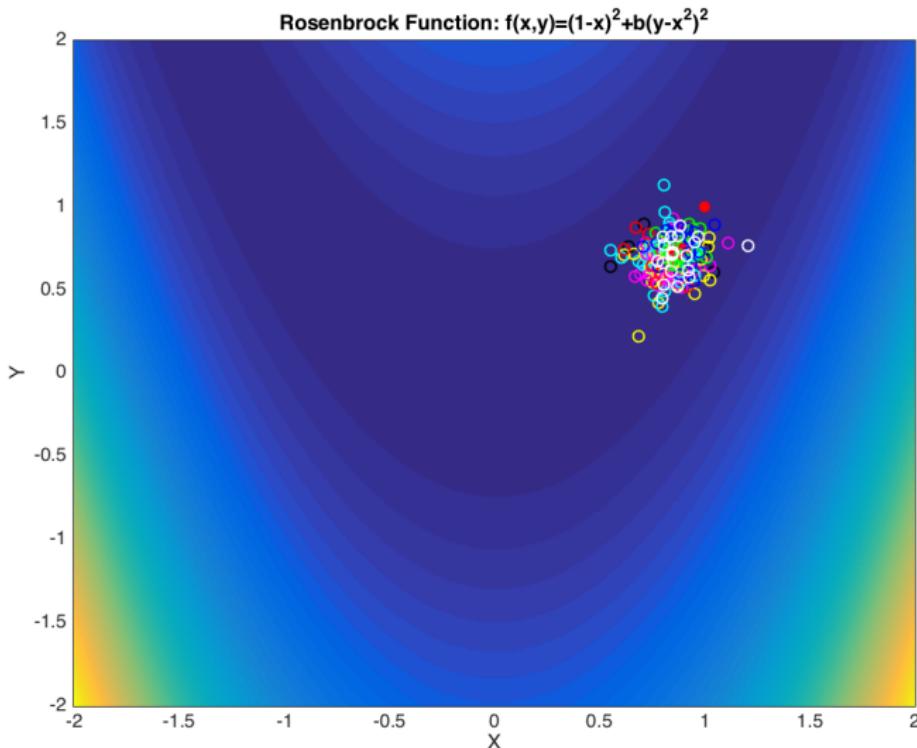
# GENETIC ALGORITHM



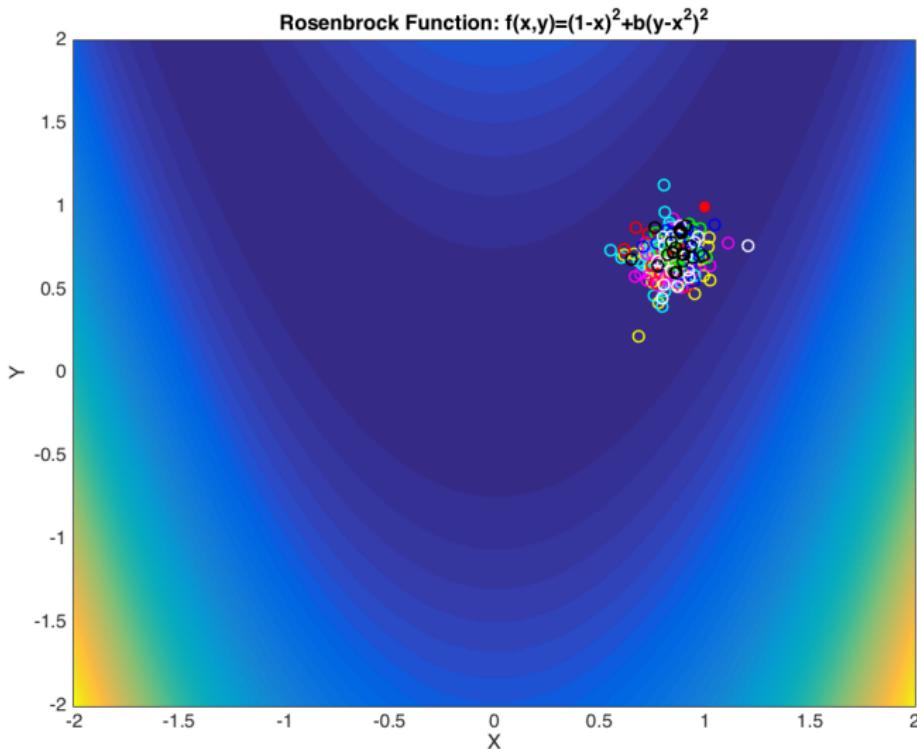
# GENETIC ALGORITHM



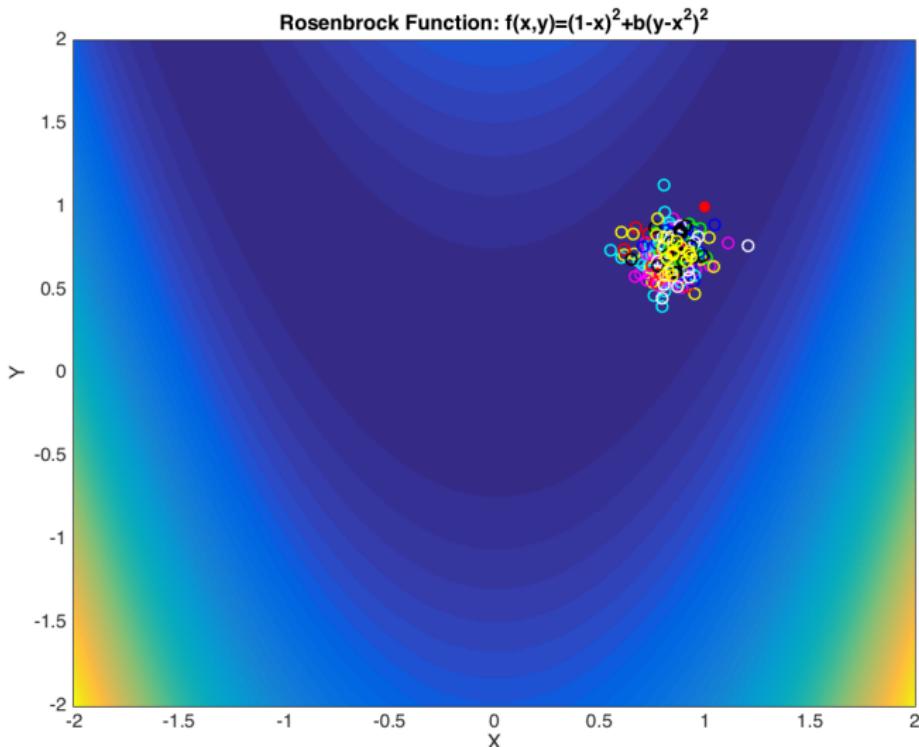
# GENETIC ALGORITHM



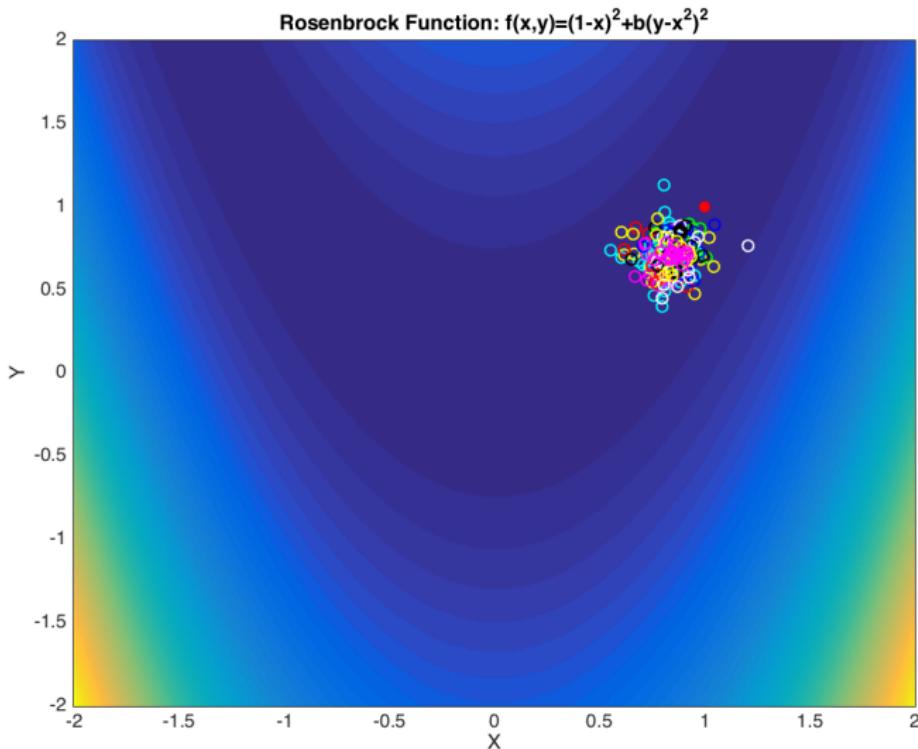
# GENETIC ALGORITHM



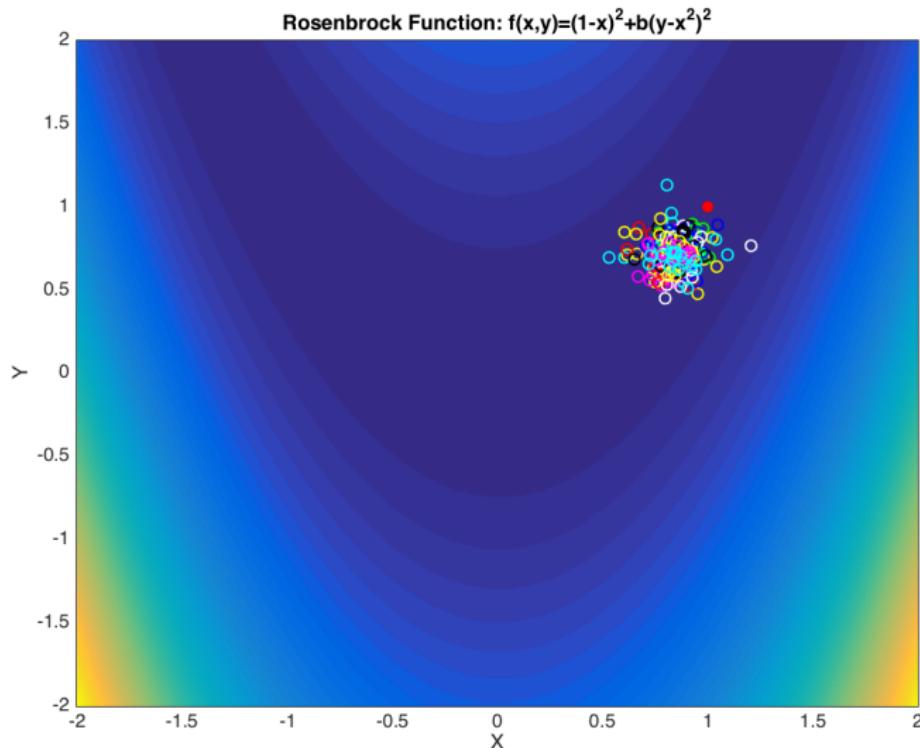
# GENETIC ALGORITHM



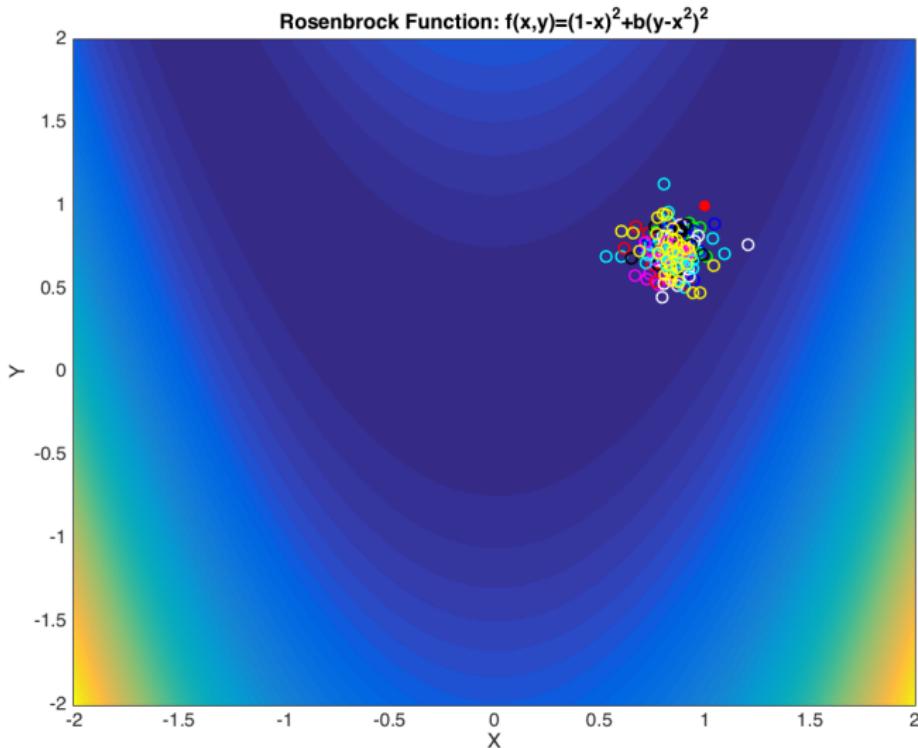
# GENETIC ALGORITHM



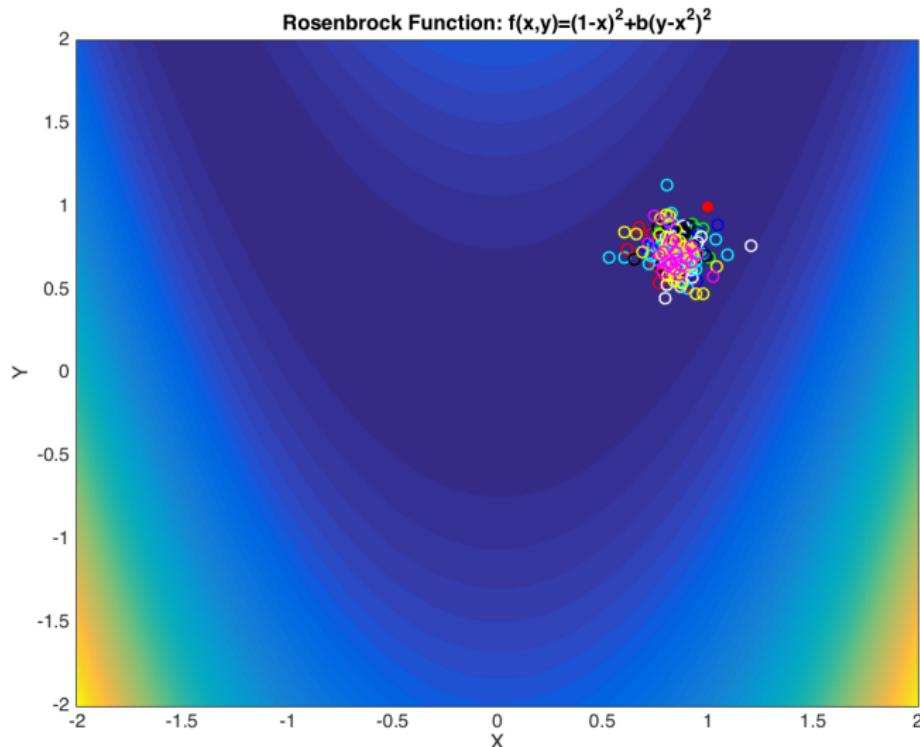
# GENETIC ALGORITHM



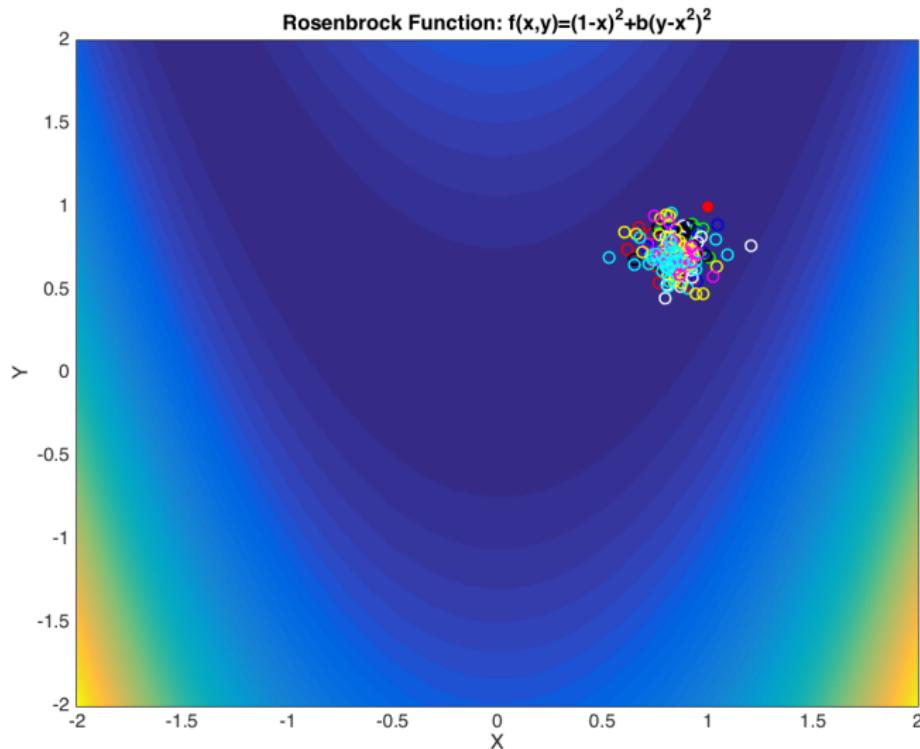
# GENETIC ALGORITHM



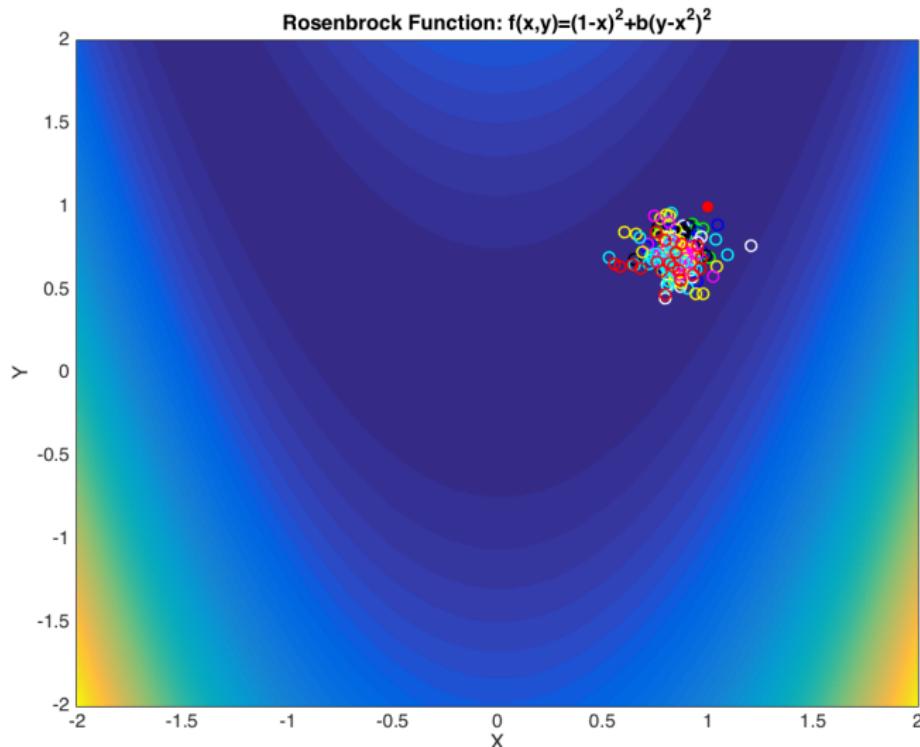
# GENETIC ALGORITHM



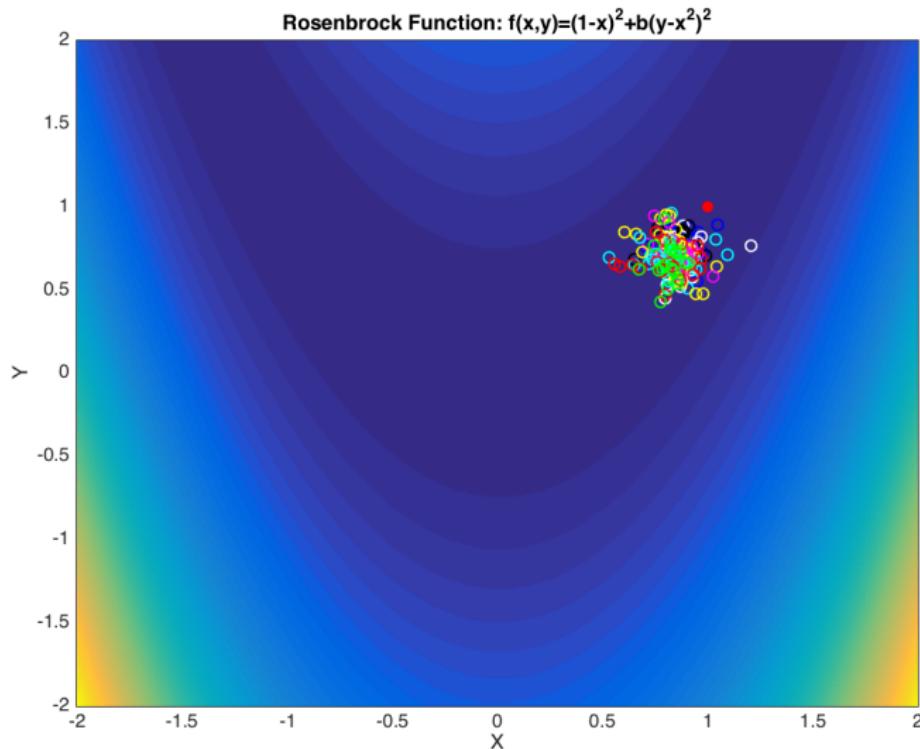
# GENETIC ALGORITHM



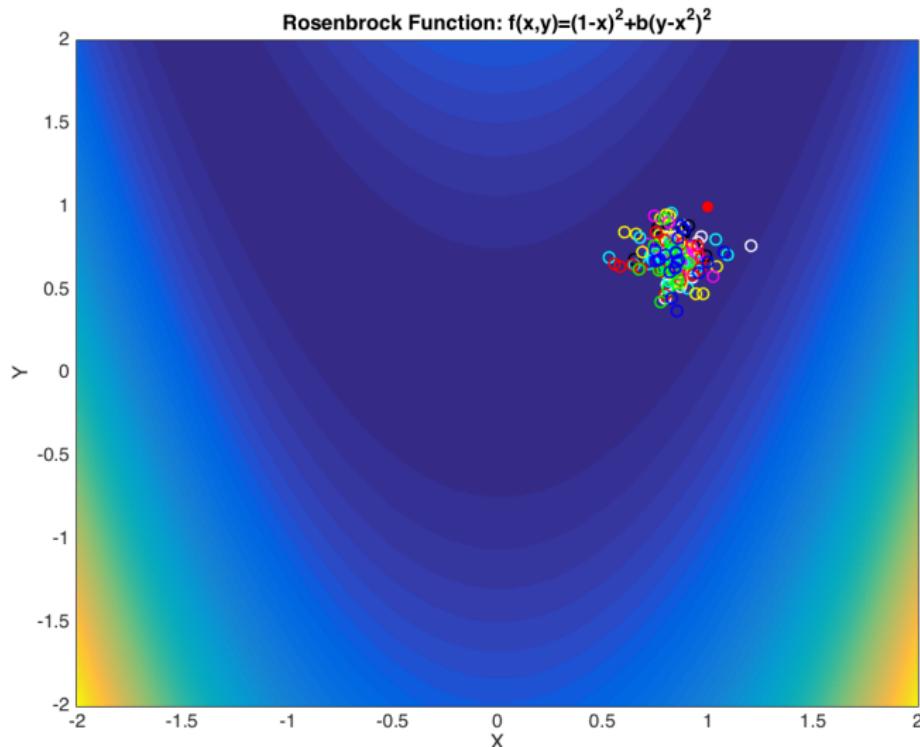
# GENETIC ALGORITHM



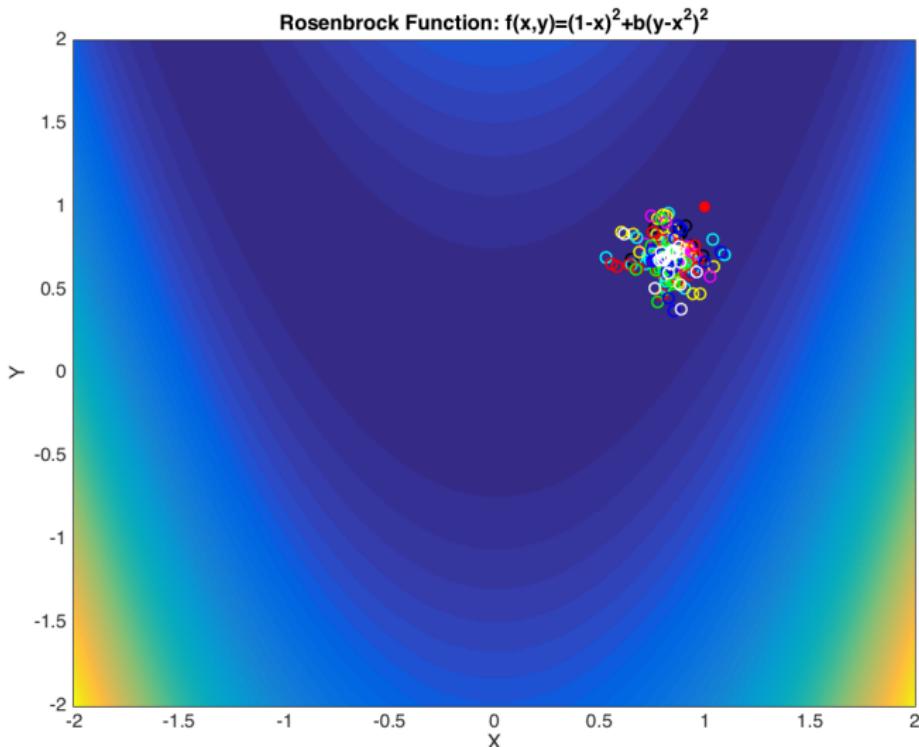
# GENETIC ALGORITHM



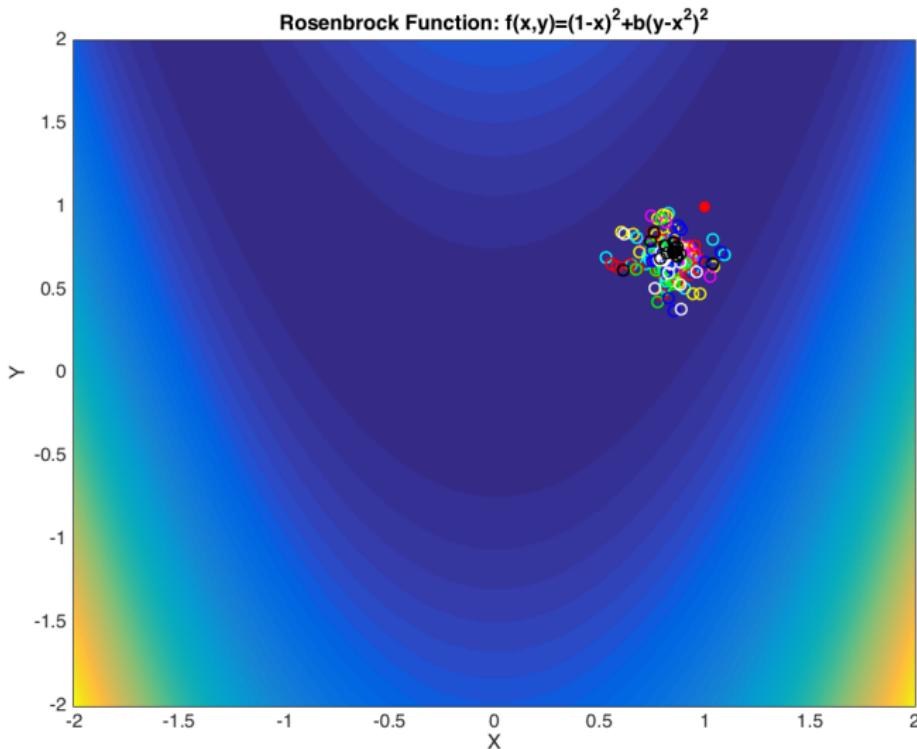
# GENETIC ALGORITHM



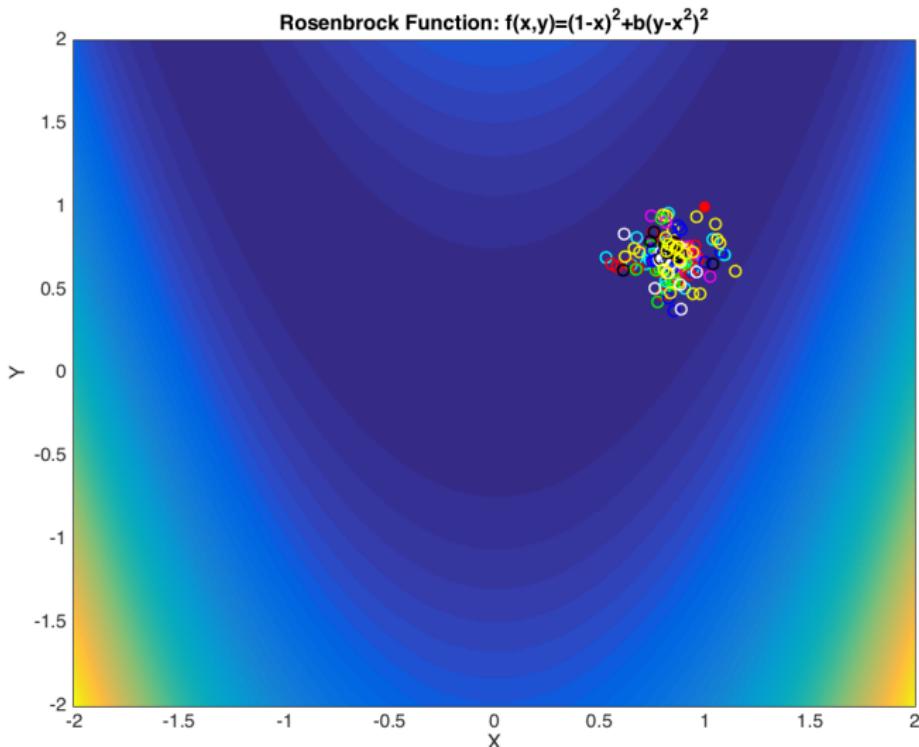
# GENETIC ALGORITHM



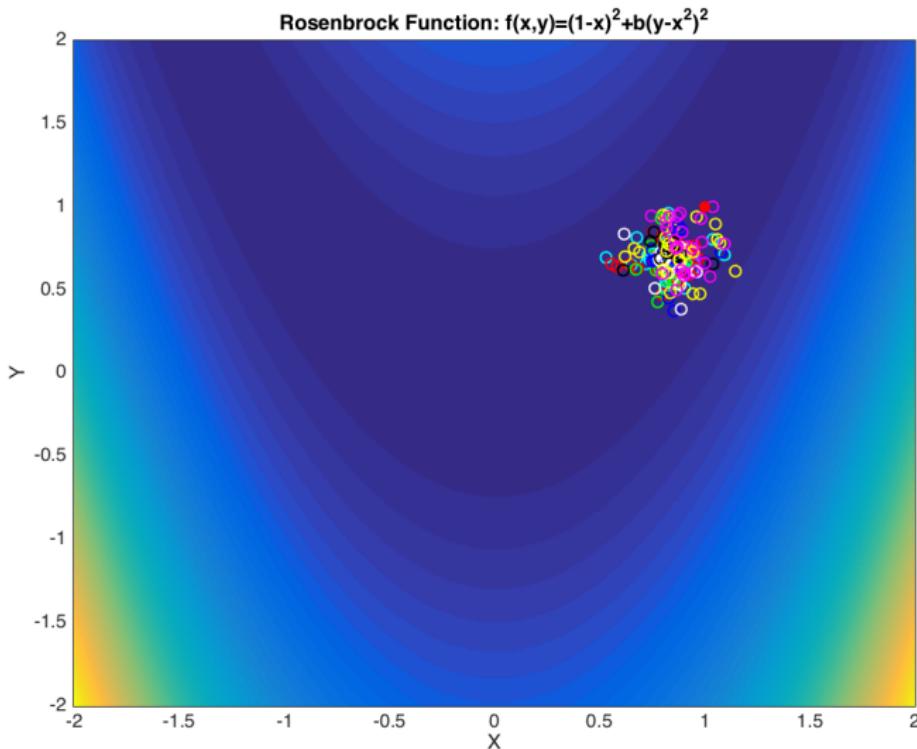
# GENETIC ALGORITHM



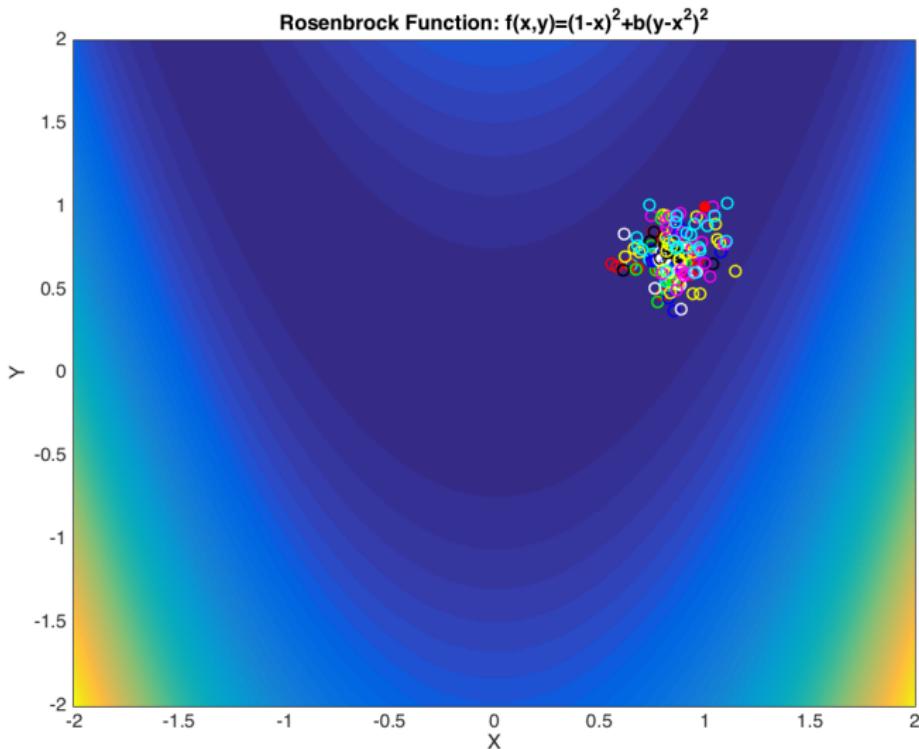
# GENETIC ALGORITHM



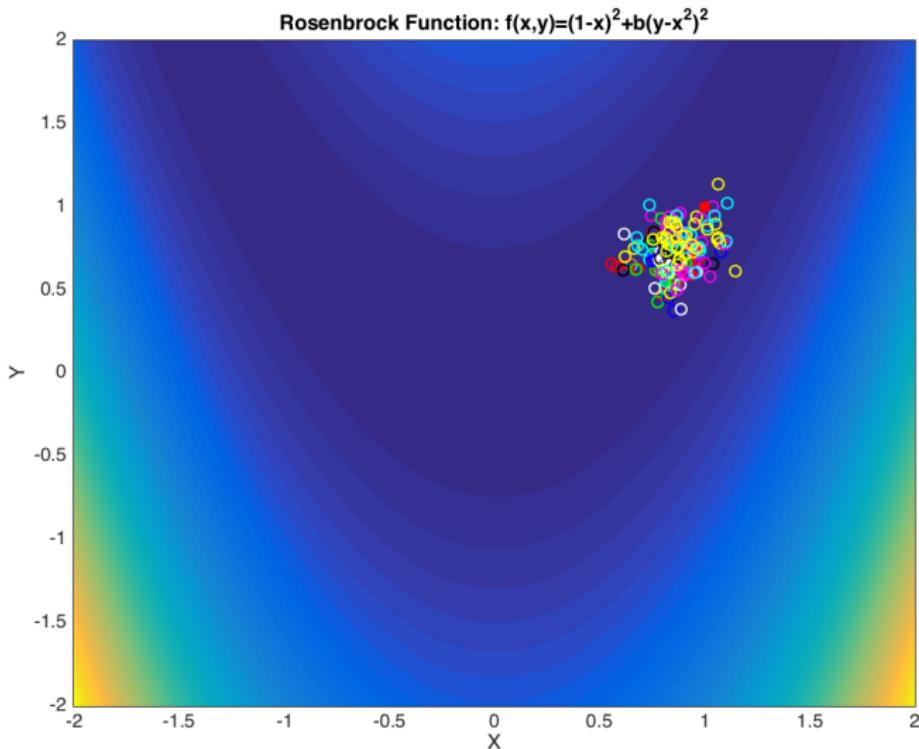
# GENETIC ALGORITHM



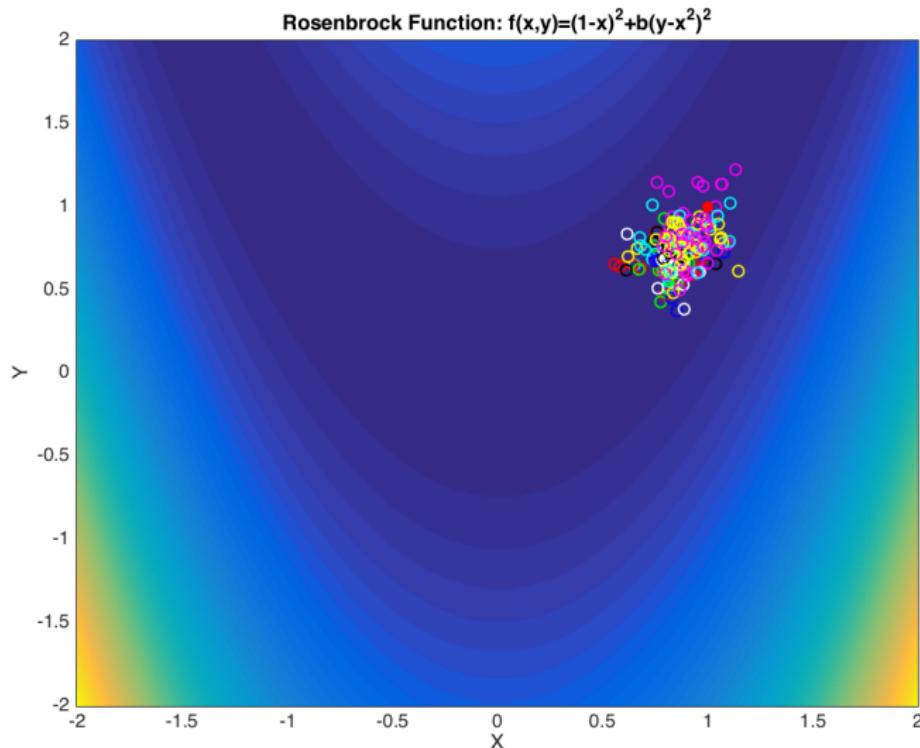
# GENETIC ALGORITHM



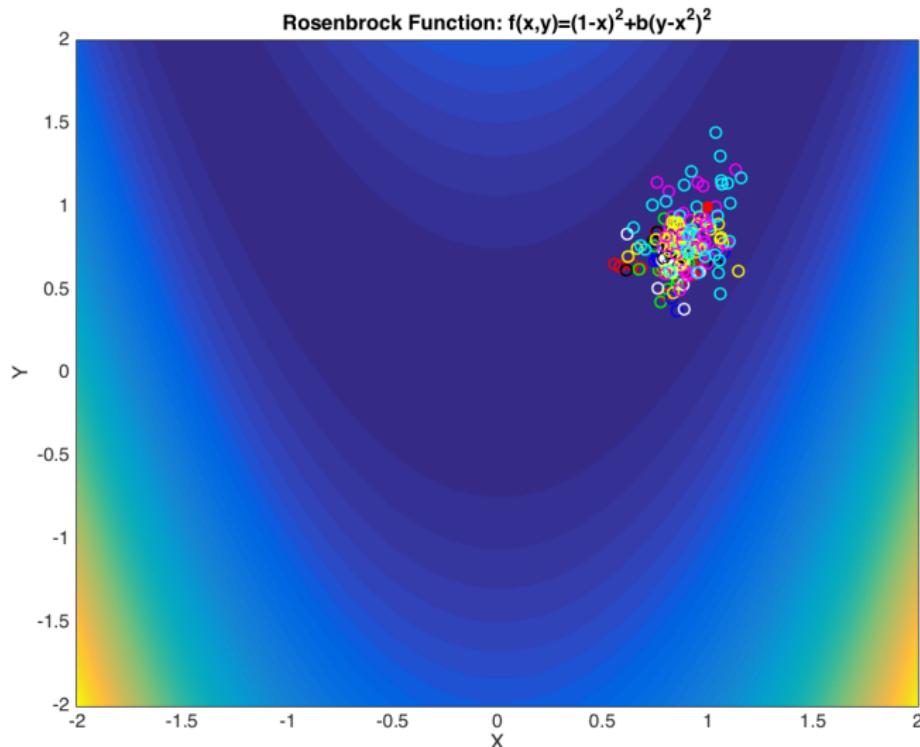
# GENETIC ALGORITHM



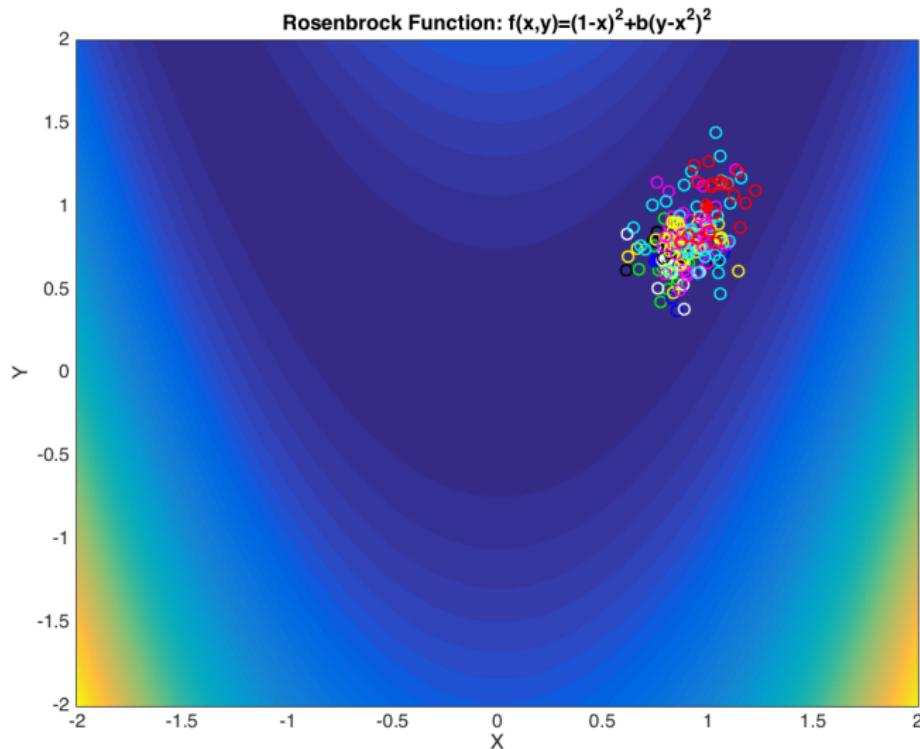
# GENETIC ALGORITHM



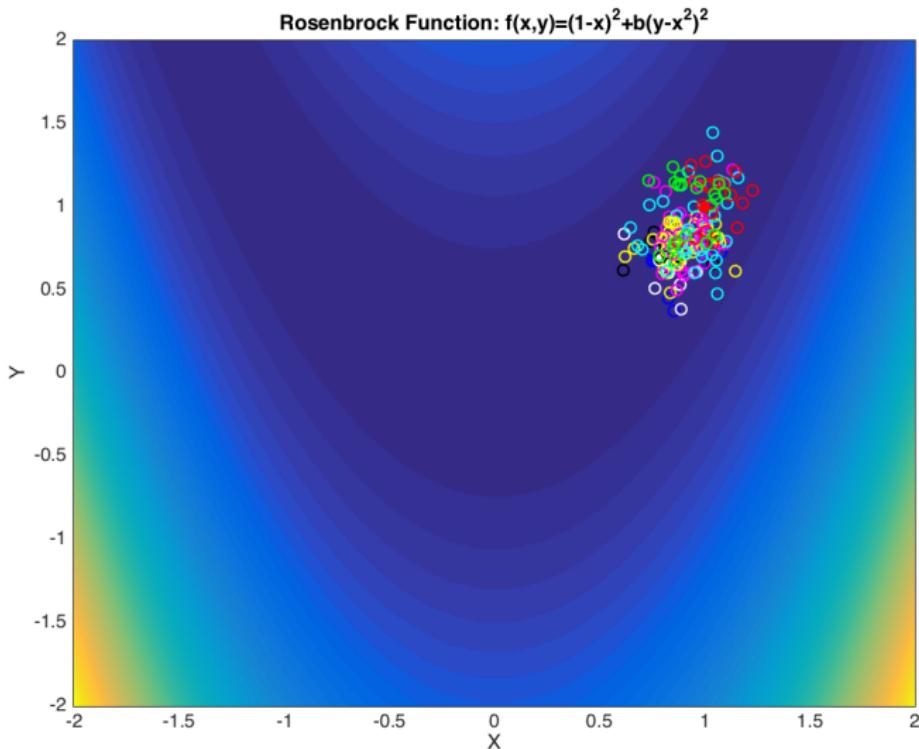
# GENETIC ALGORITHM



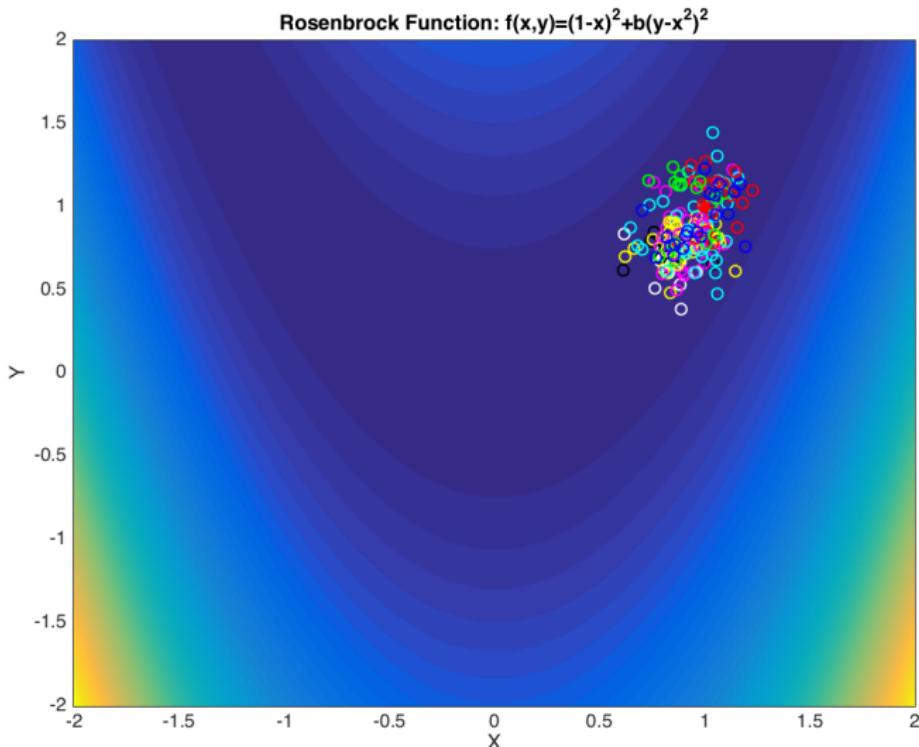
# GENETIC ALGORITHM



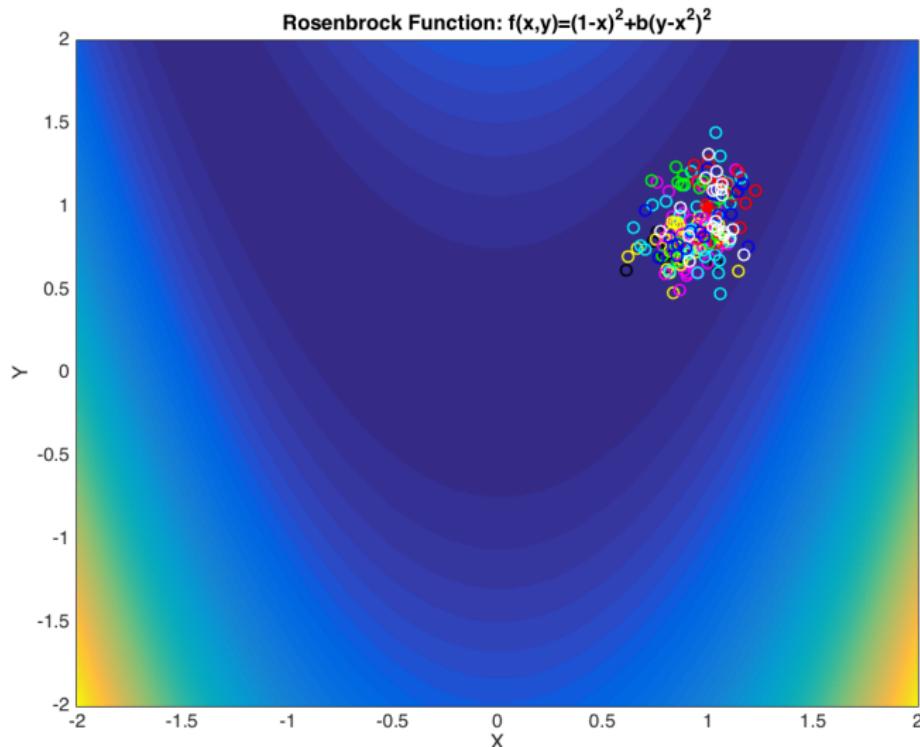
# GENETIC ALGORITHM



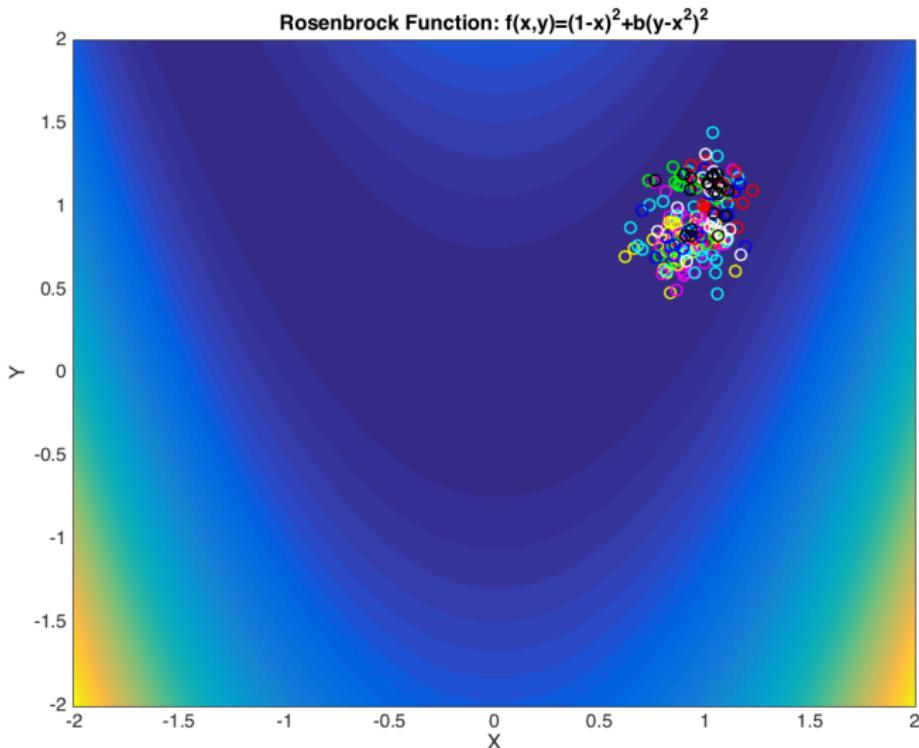
# GENETIC ALGORITHM



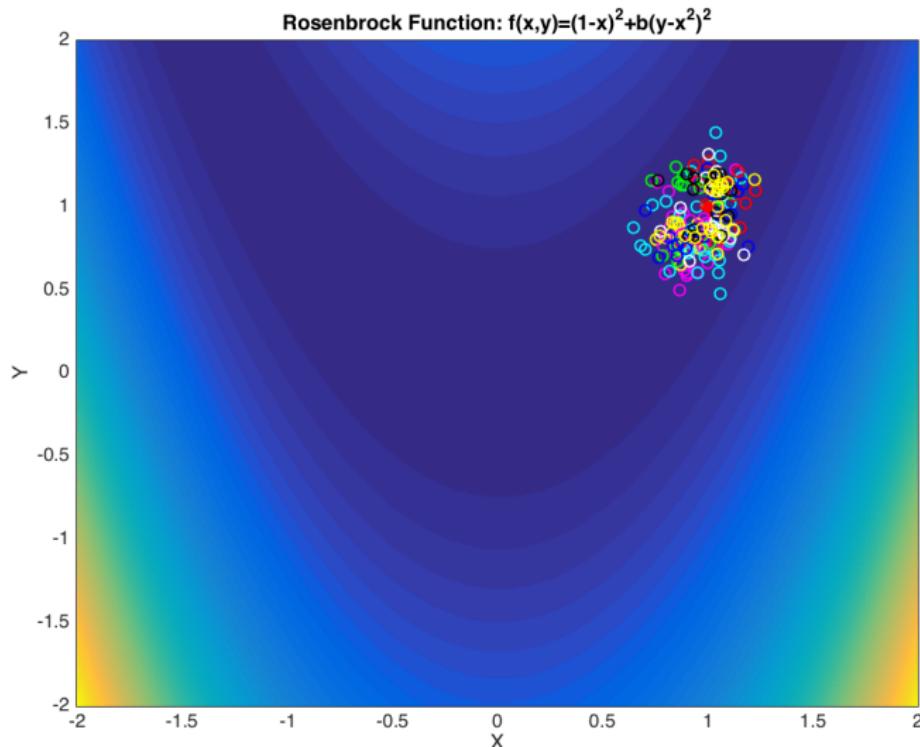
# GENETIC ALGORITHM



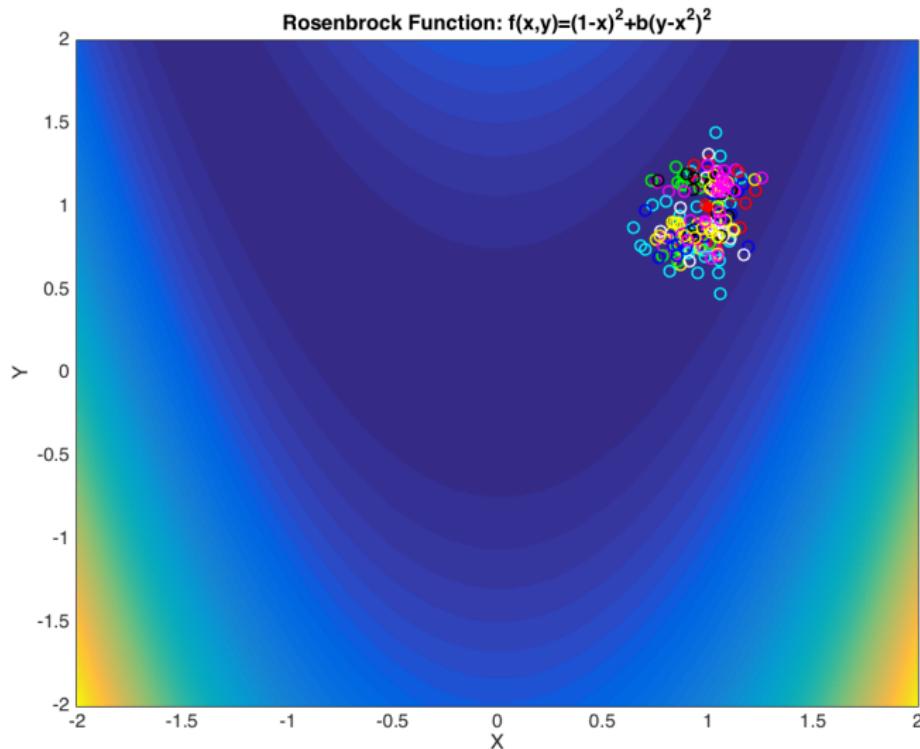
# GENETIC ALGORITHM



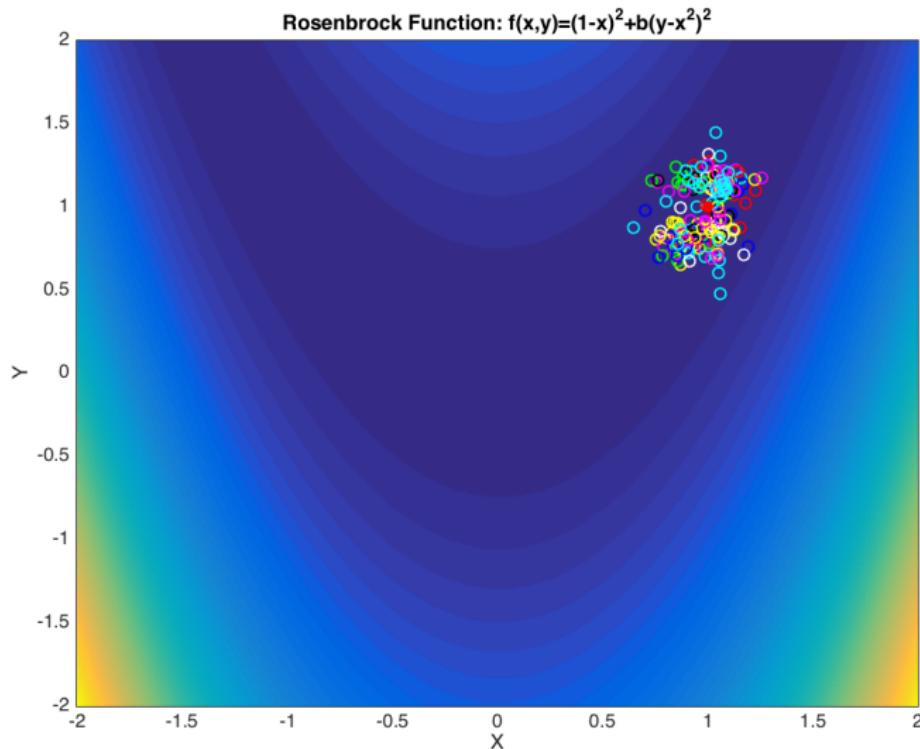
# GENETIC ALGORITHM



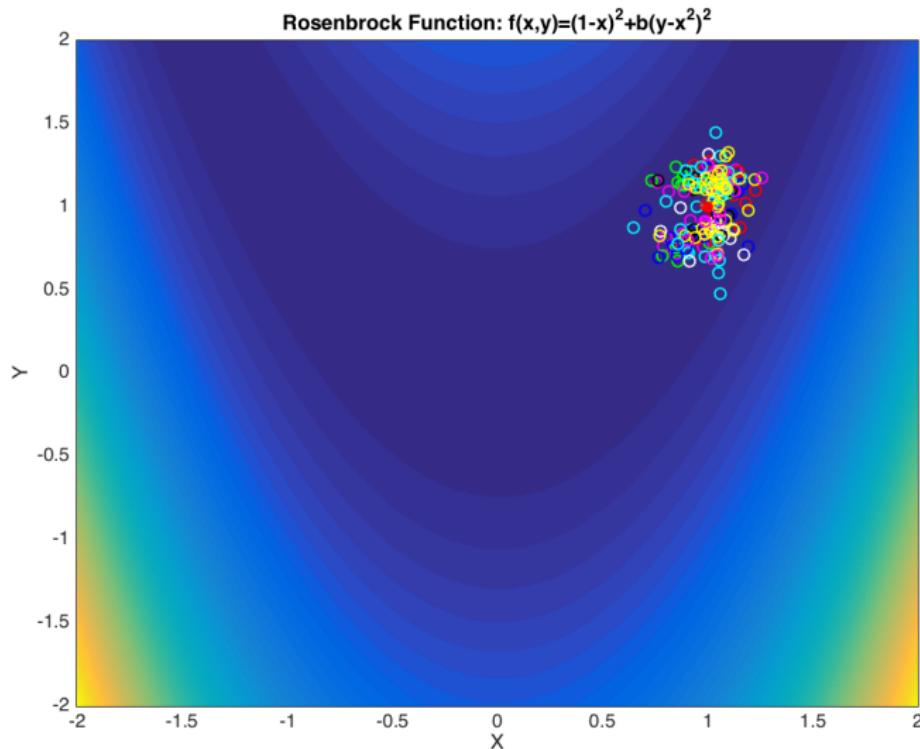
# GENETIC ALGORITHM



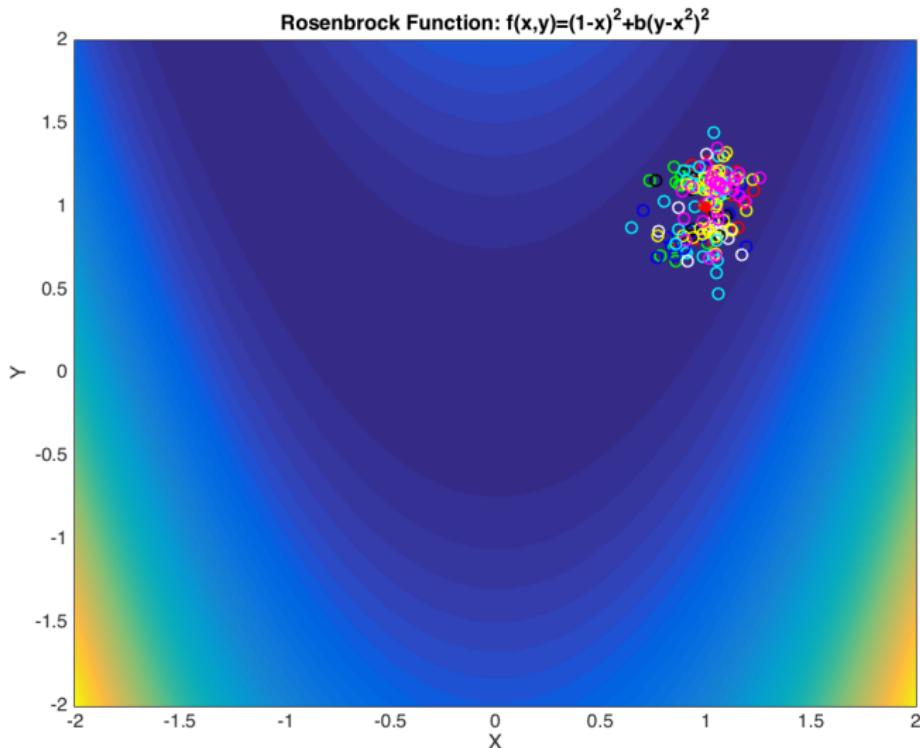
# GENETIC ALGORITHM



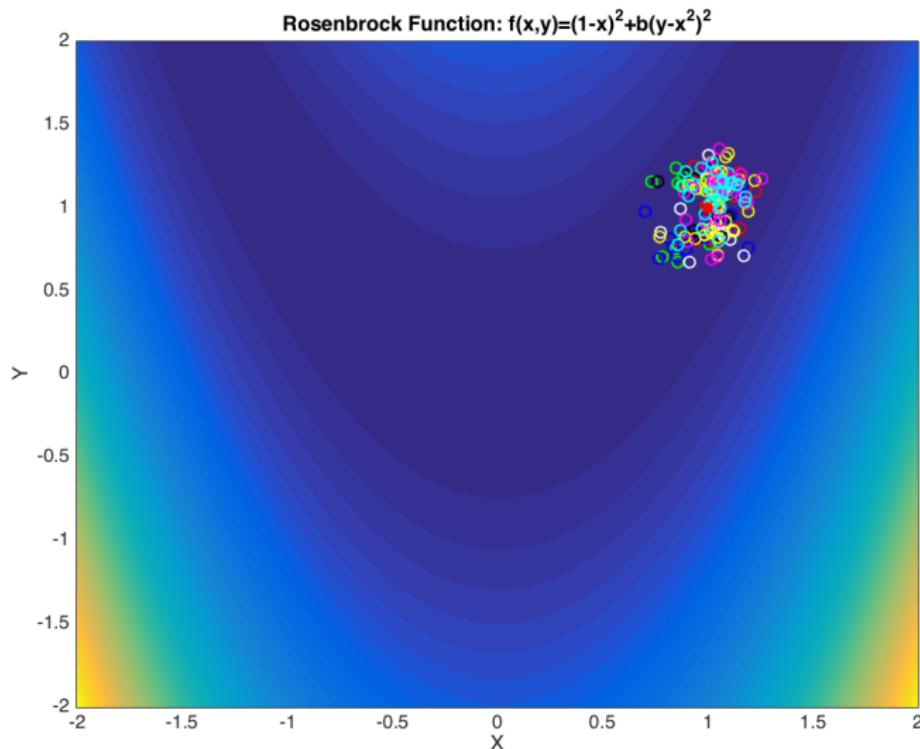
# GENETIC ALGORITHM



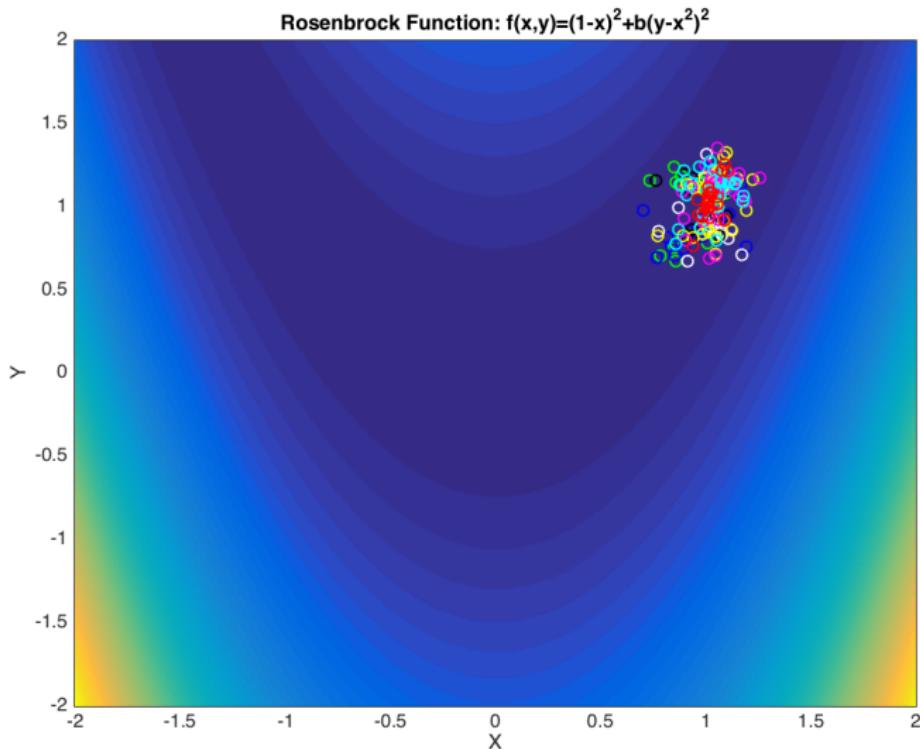
# GENETIC ALGORITHM



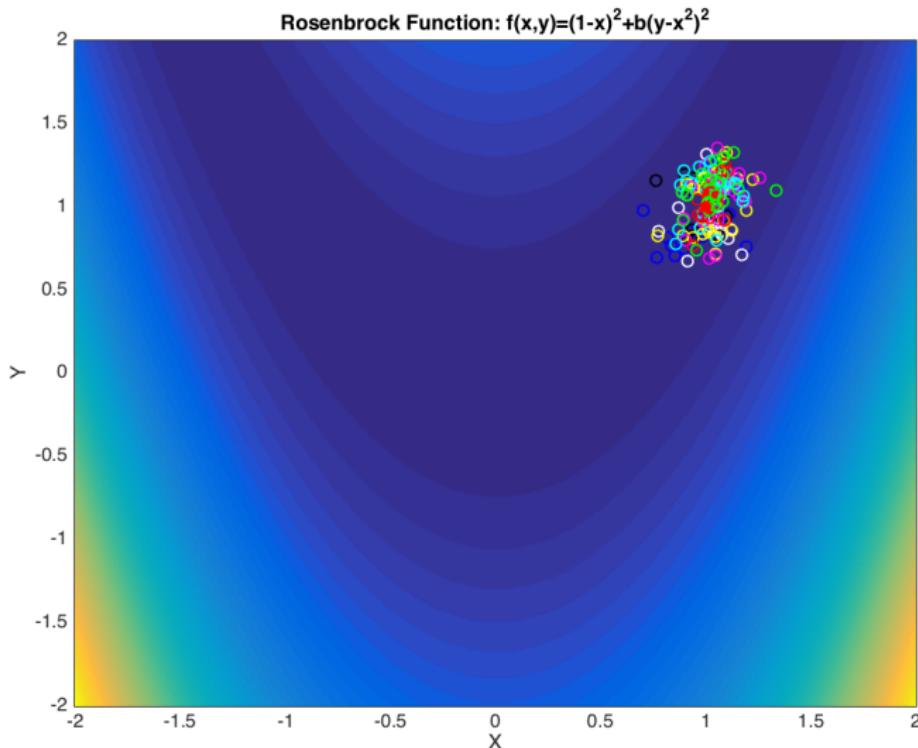
# GENETIC ALGORITHM



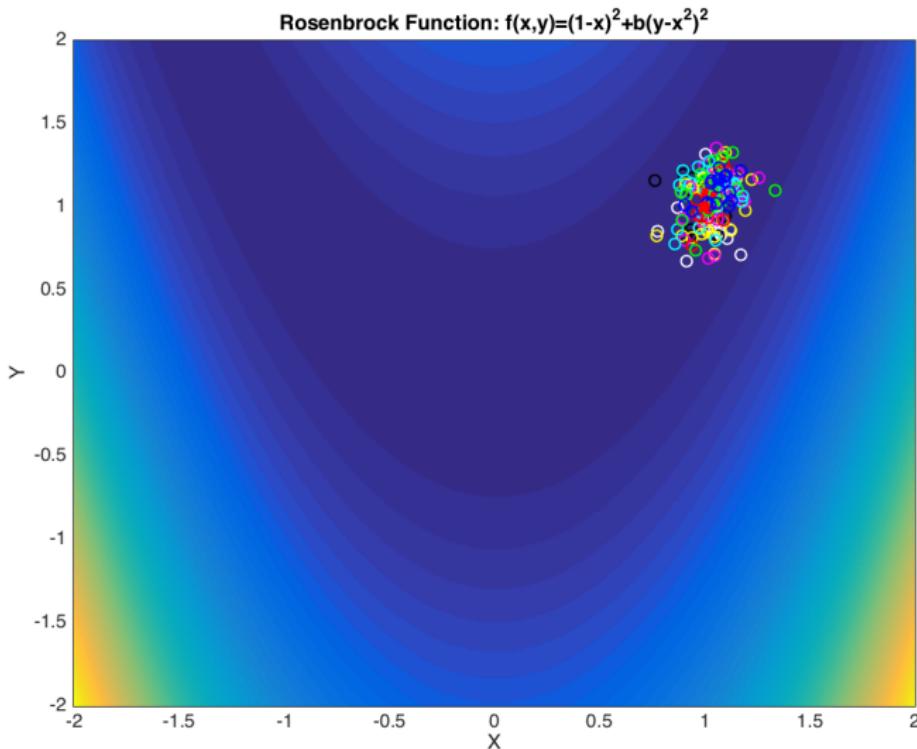
# GENETIC ALGORITHM



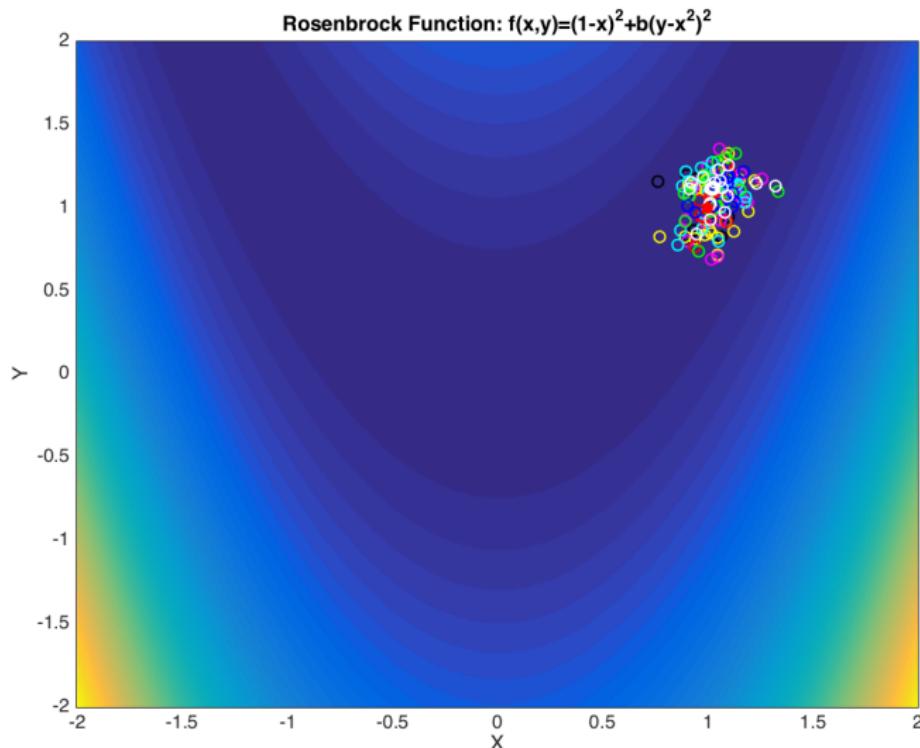
# GENETIC ALGORITHM



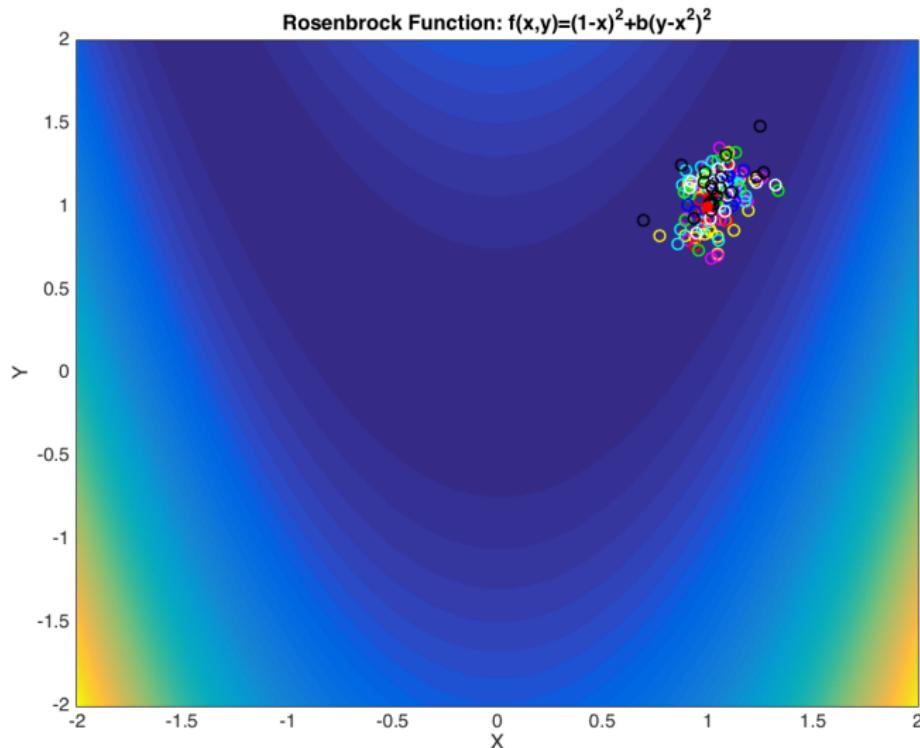
# GENETIC ALGORITHM



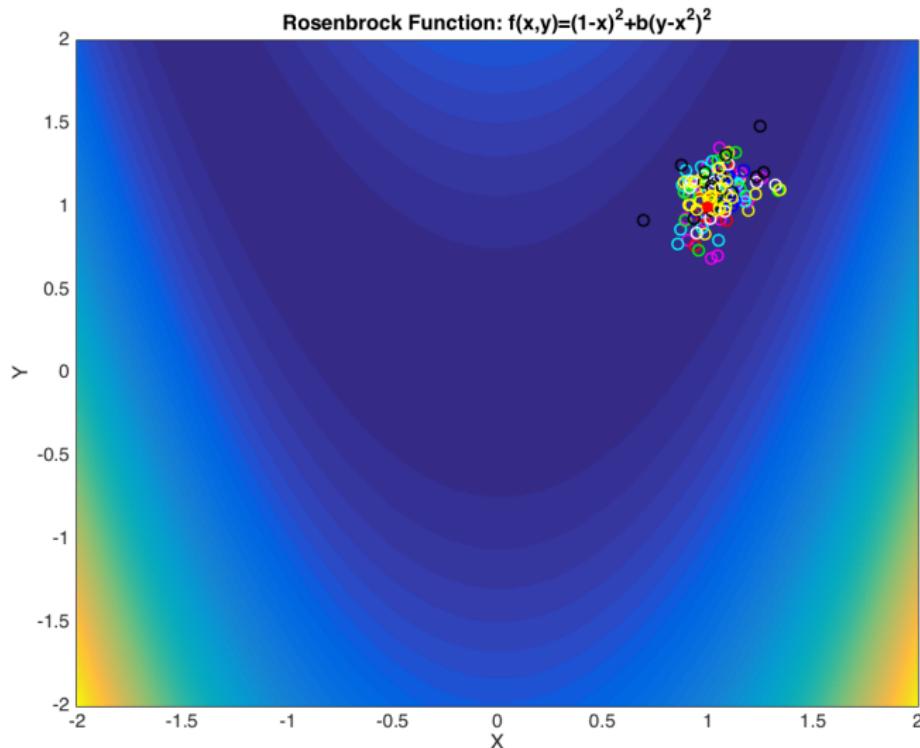
# GENETIC ALGORITHM



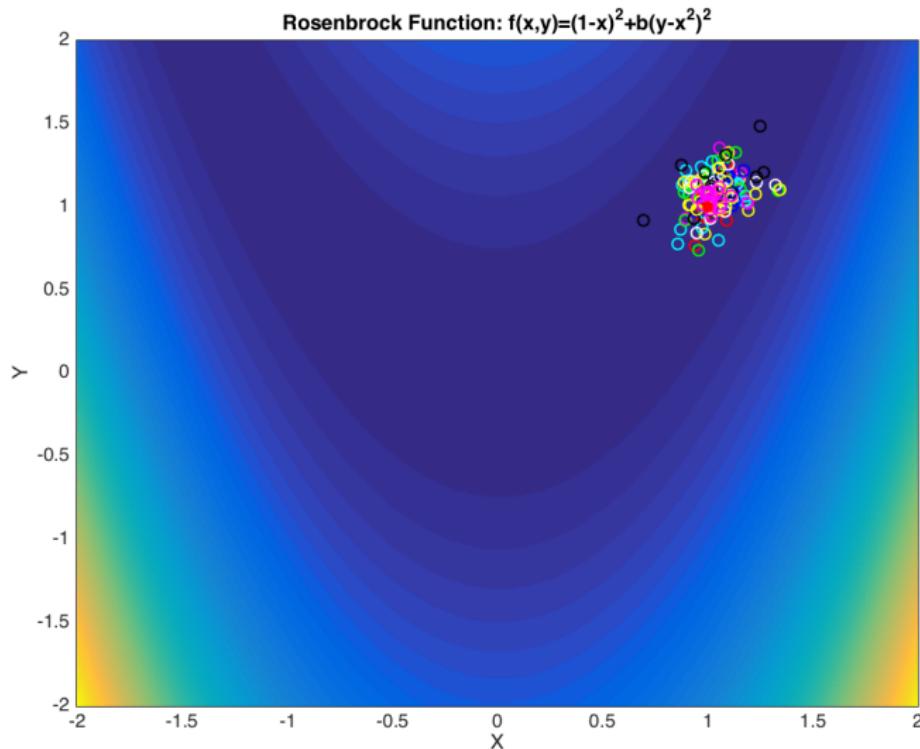
# GENETIC ALGORITHM



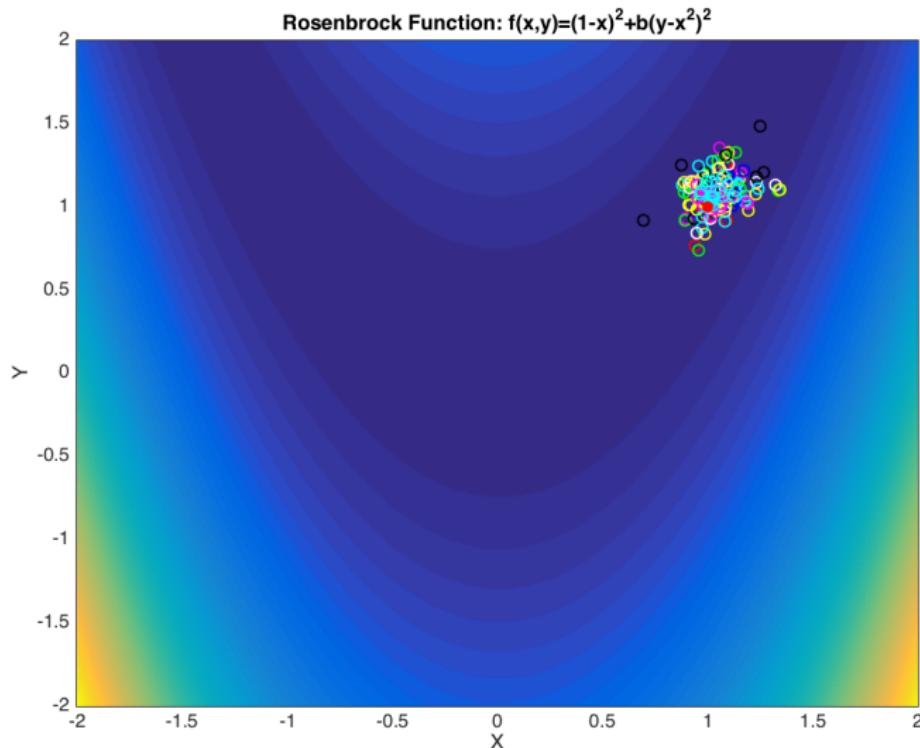
# GENETIC ALGORITHM



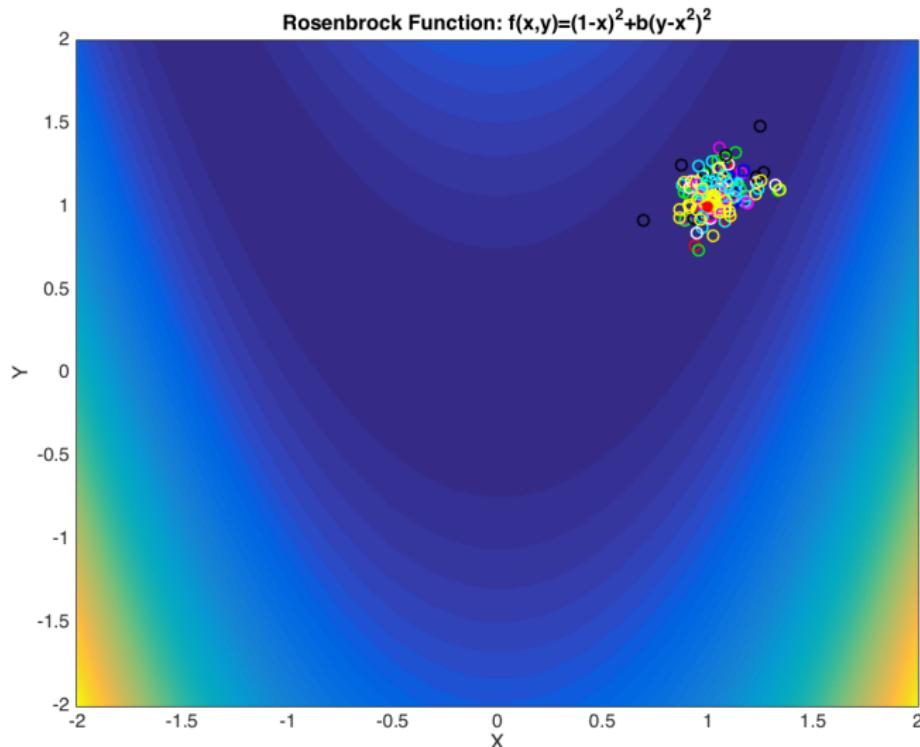
# GENETIC ALGORITHM



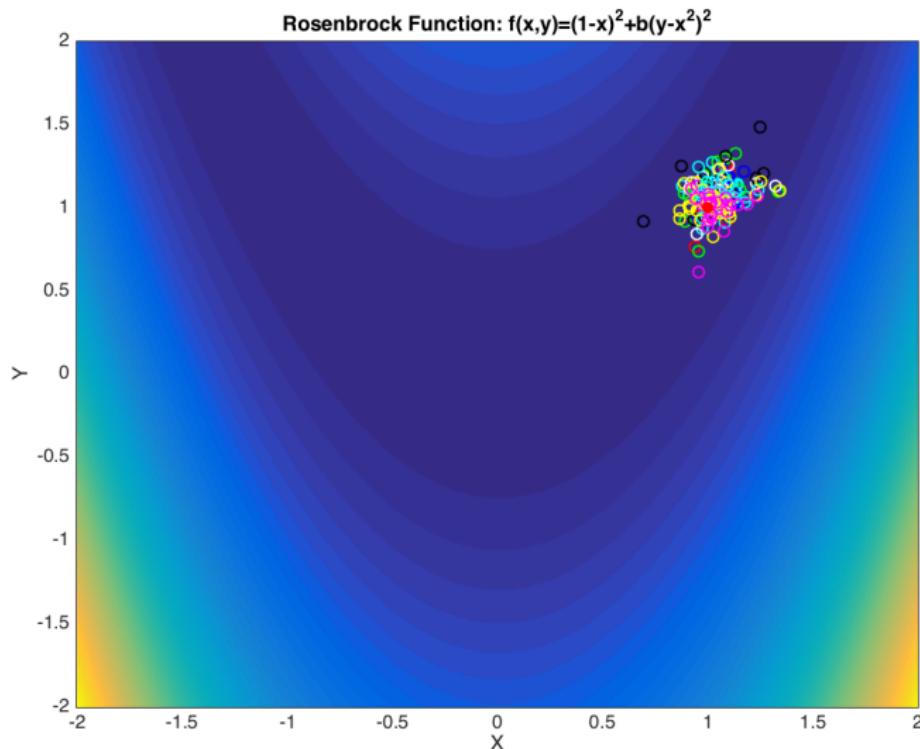
# GENETIC ALGORITHM



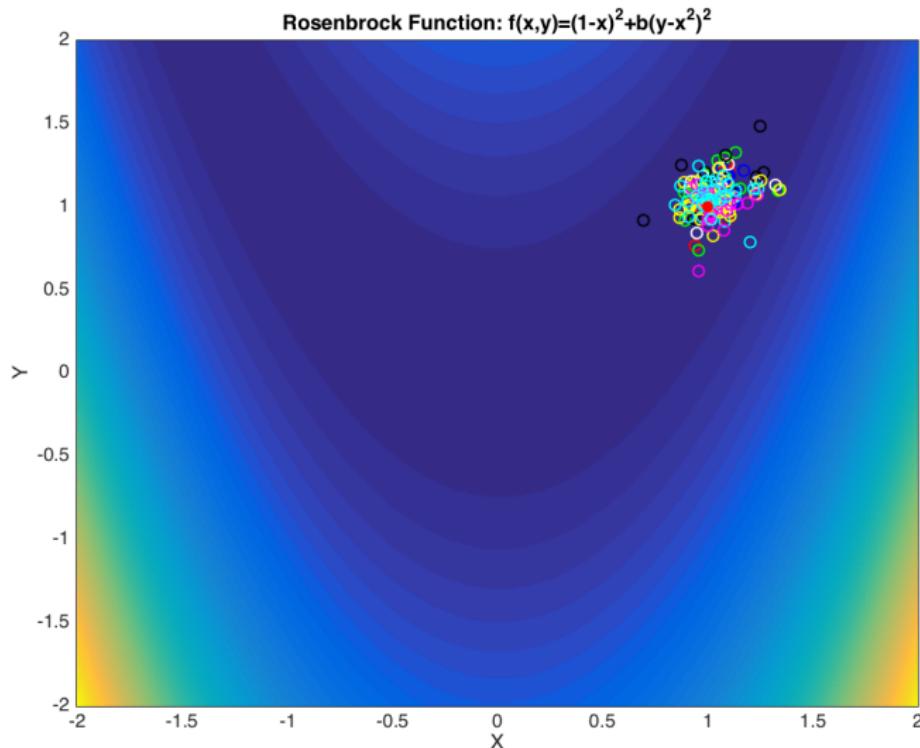
# GENETIC ALGORITHM



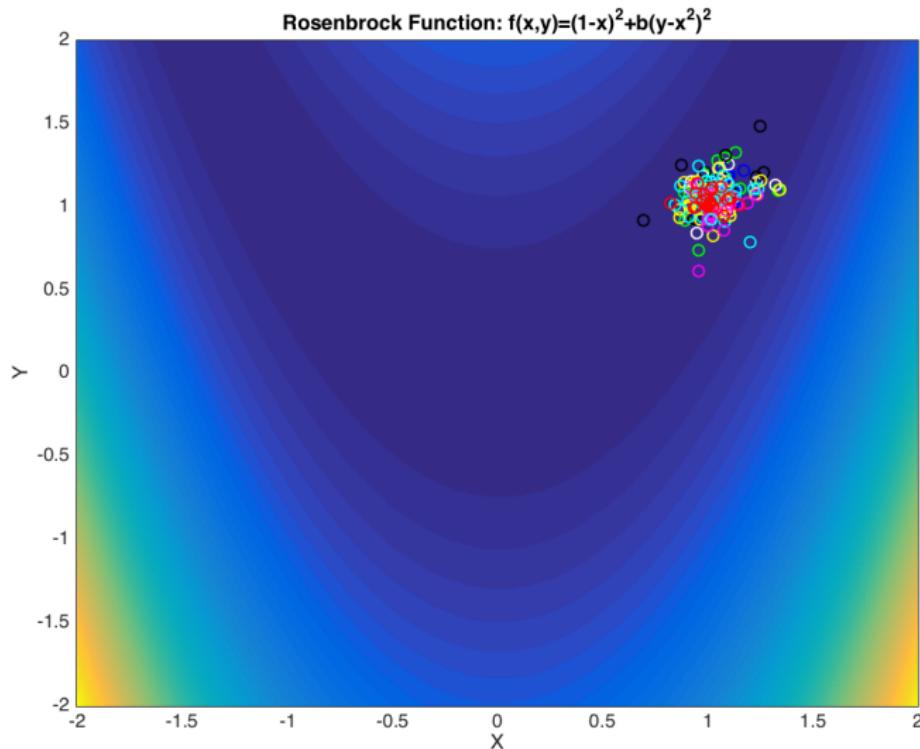
# GENETIC ALGORITHM



# GENETIC ALGORITHM



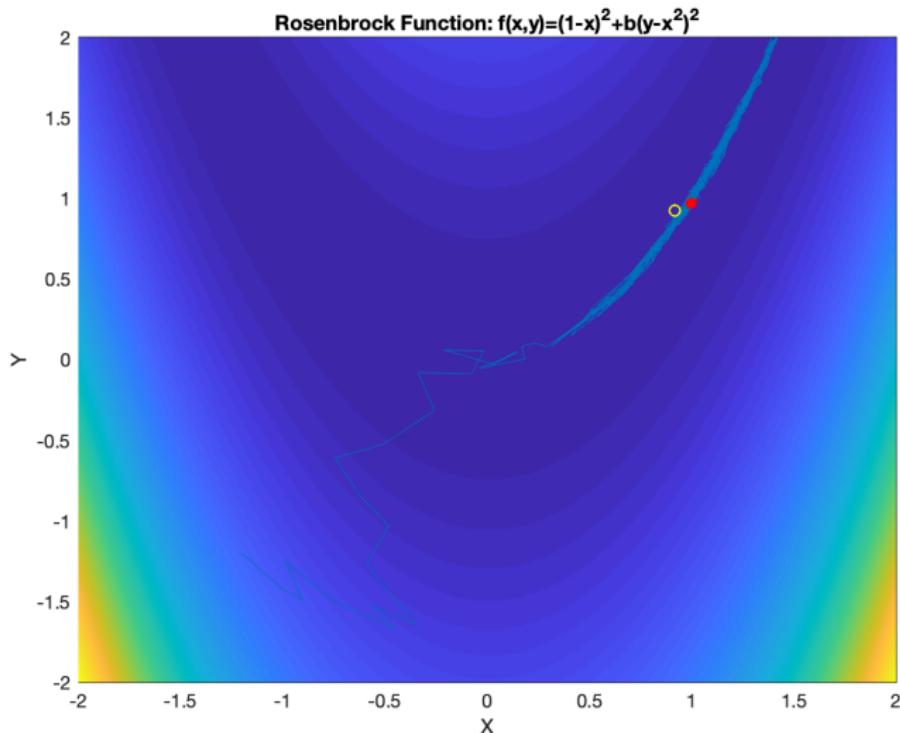
# GENETIC ALGORITHM



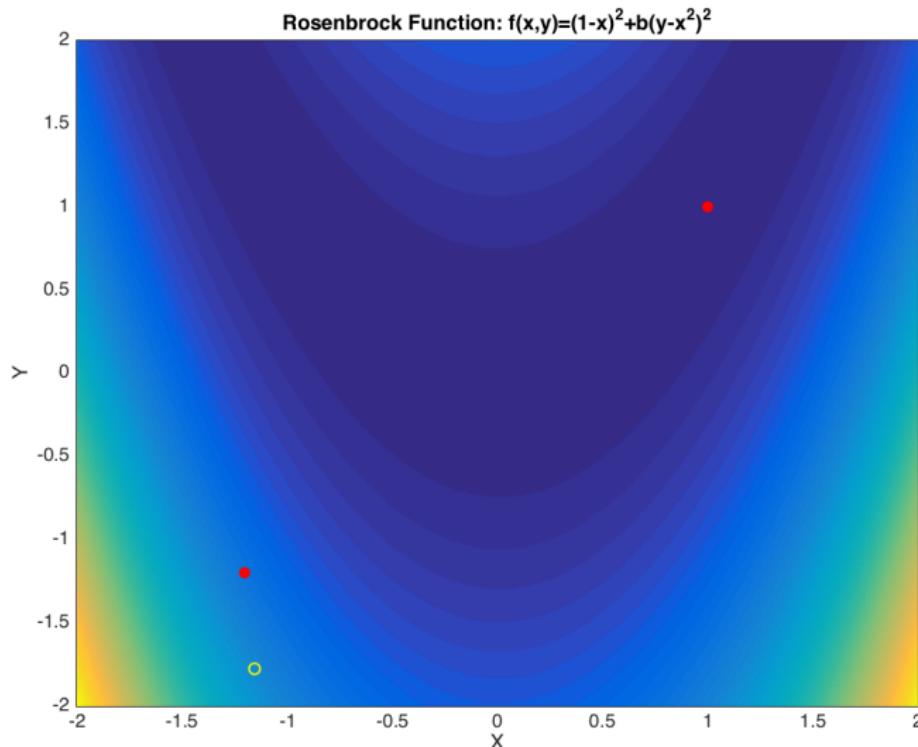
## SIMULATED ANNEALING

- ▶ Simulated annealing is pretty cool
- ▶ Phenomenally slow in my experience
  1. Start with a point and a “temperature”
  2. Look at new point, evaluate “energy” of both points.
  3. Difference between old and new plus temperature is positive, move
  4. Otherwise, stay in same place
  5. Slowly lower the temperature, repeat
- ▶ If temperature was zero, only accept lower points

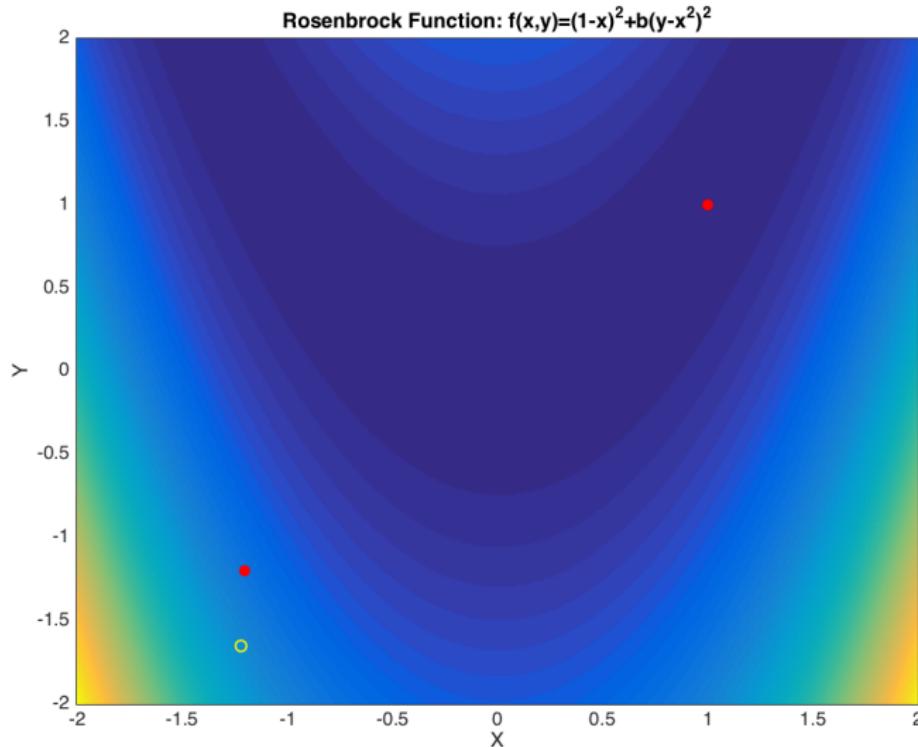
# SIMULATED ANNEALING



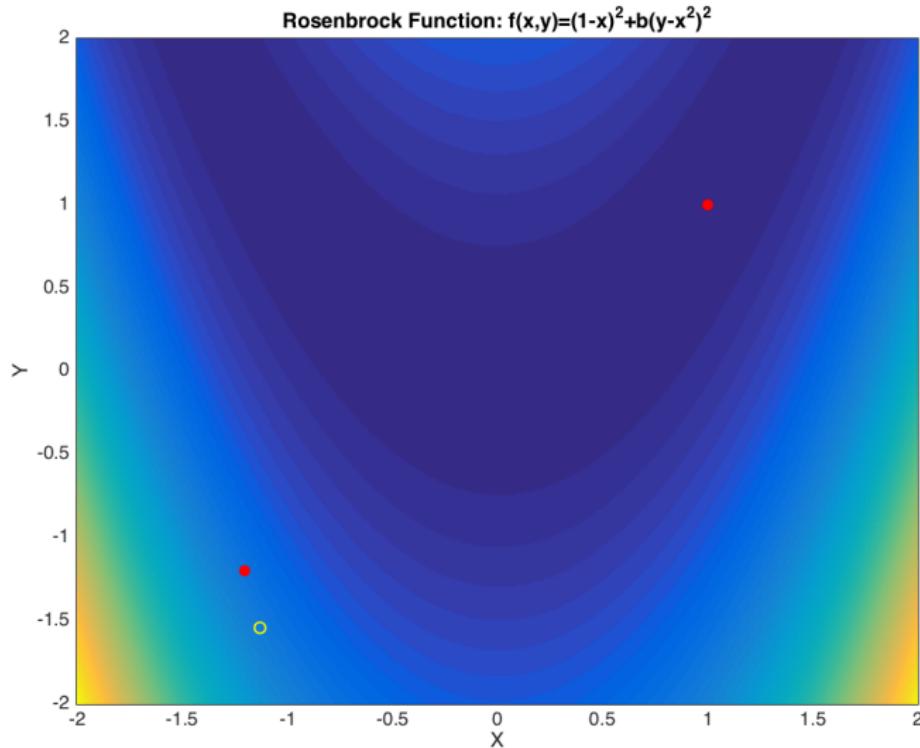
# SIMULATED ANNEALING



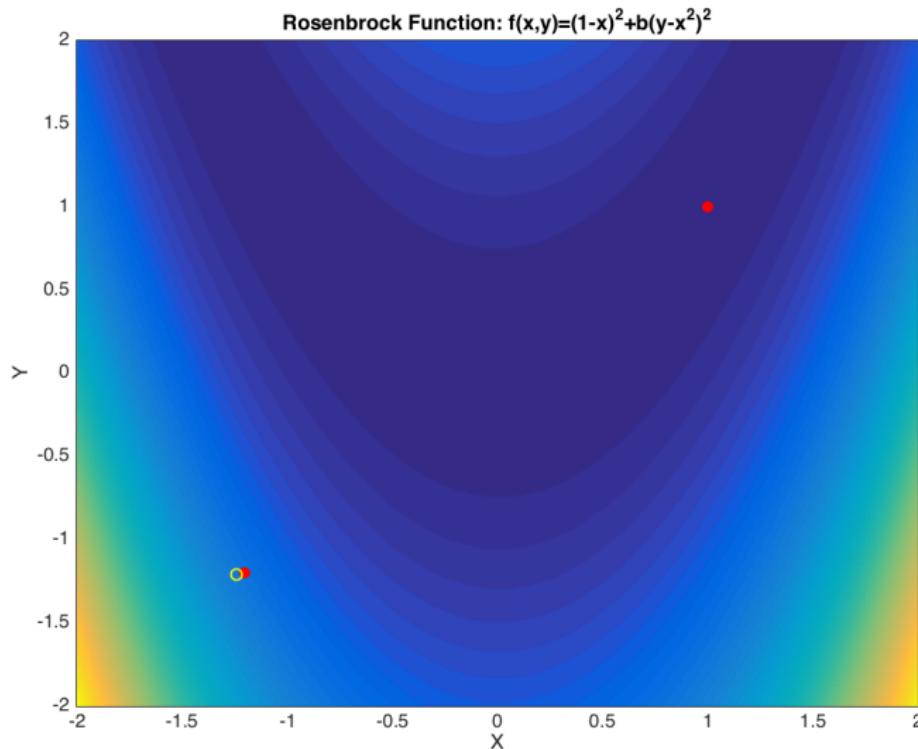
# SIMULATED ANNEALING



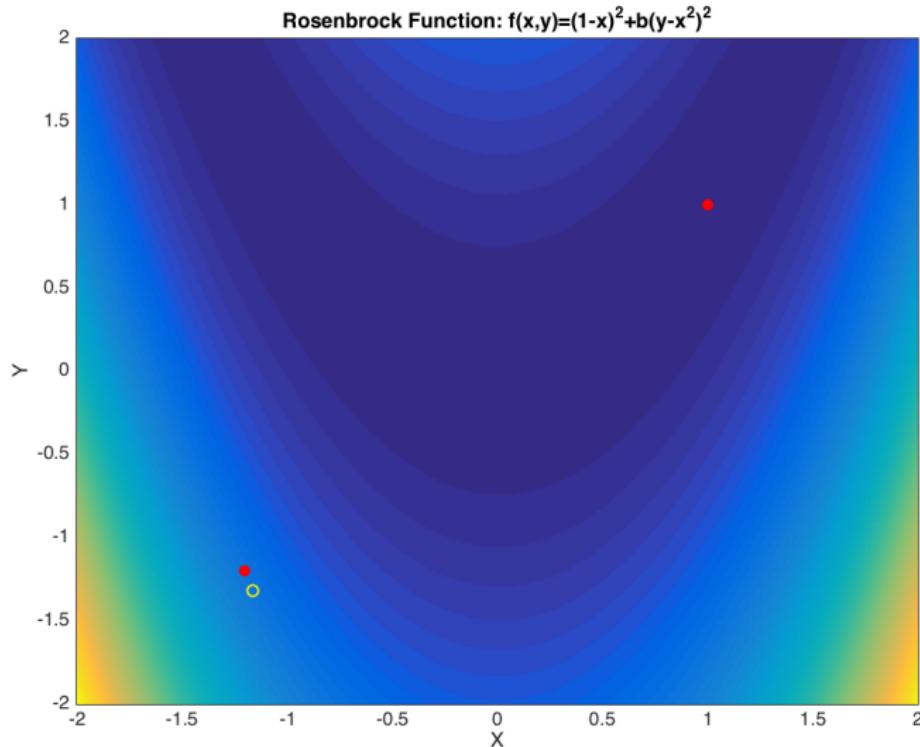
# SIMULATED ANNEALING



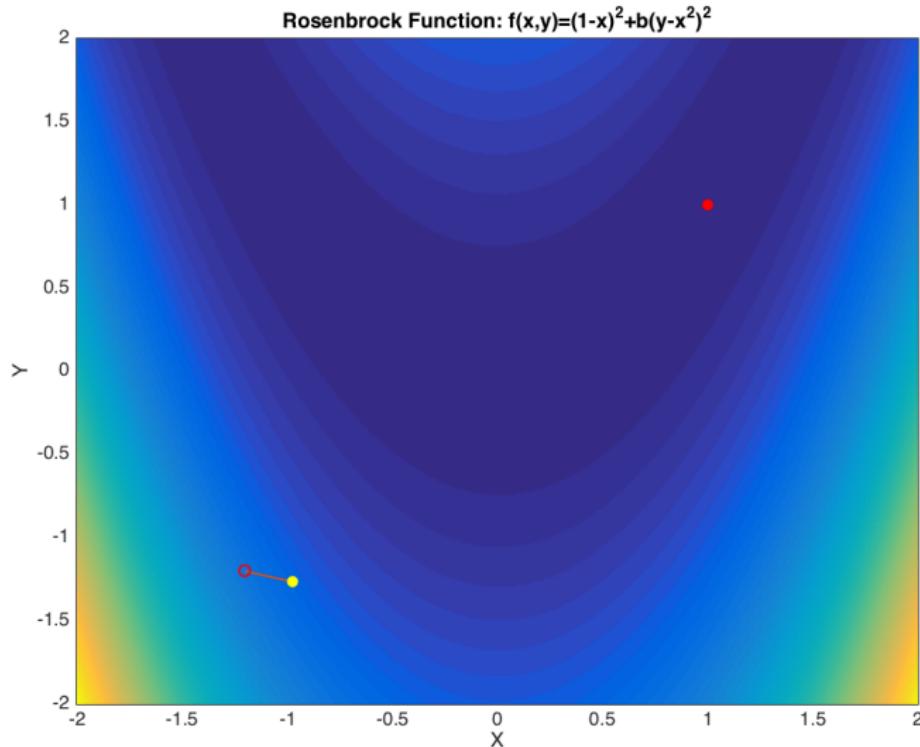
# SIMULATED ANNEALING



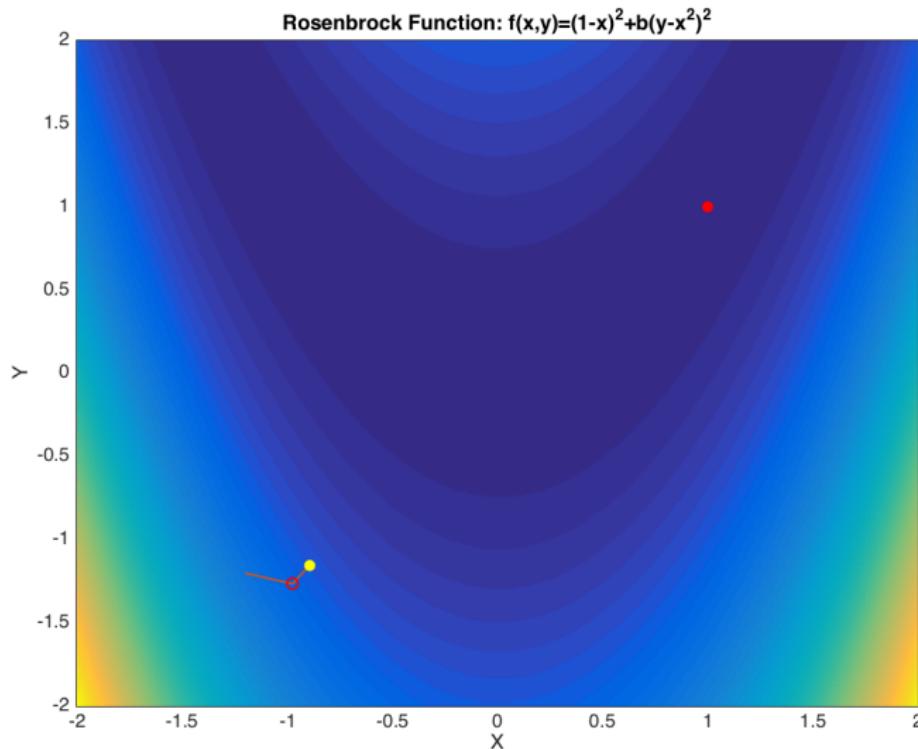
# SIMULATED ANNEALING



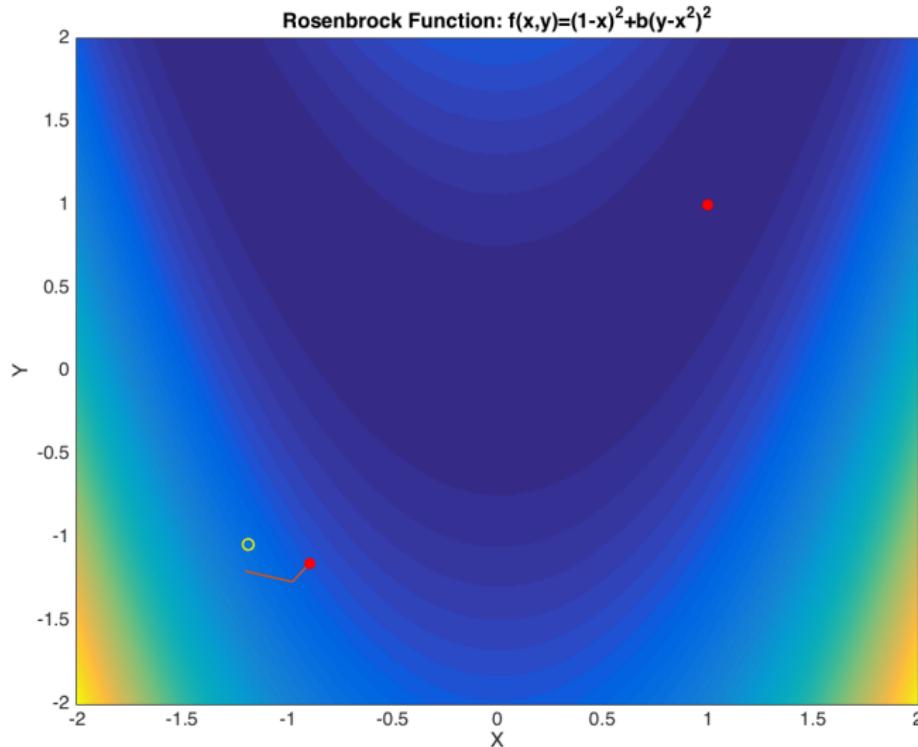
# SIMULATED ANNEALING



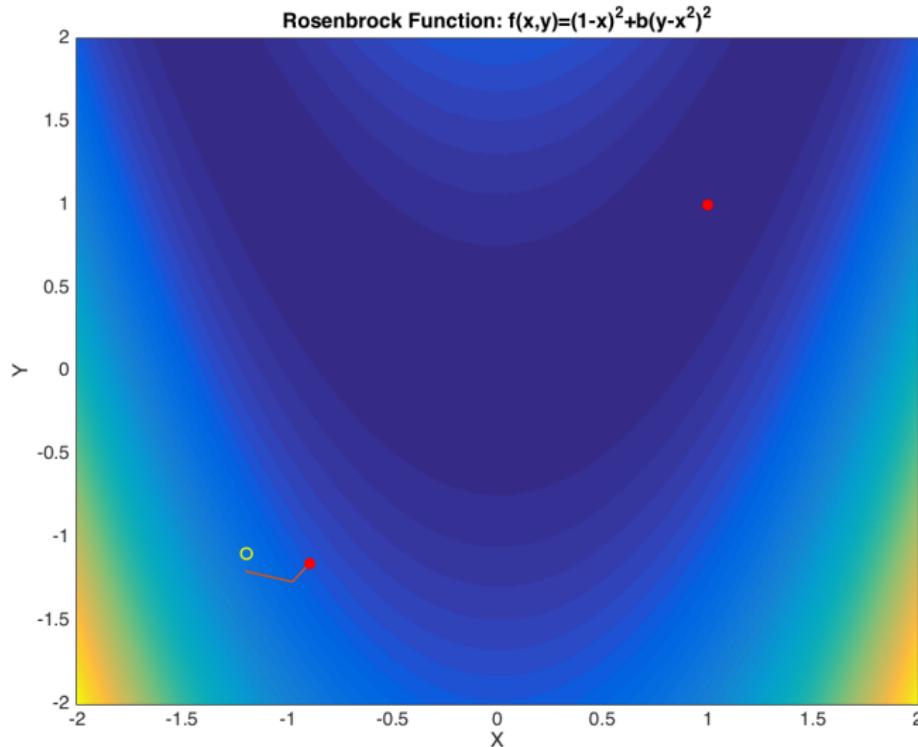
# SIMULATED ANNEALING



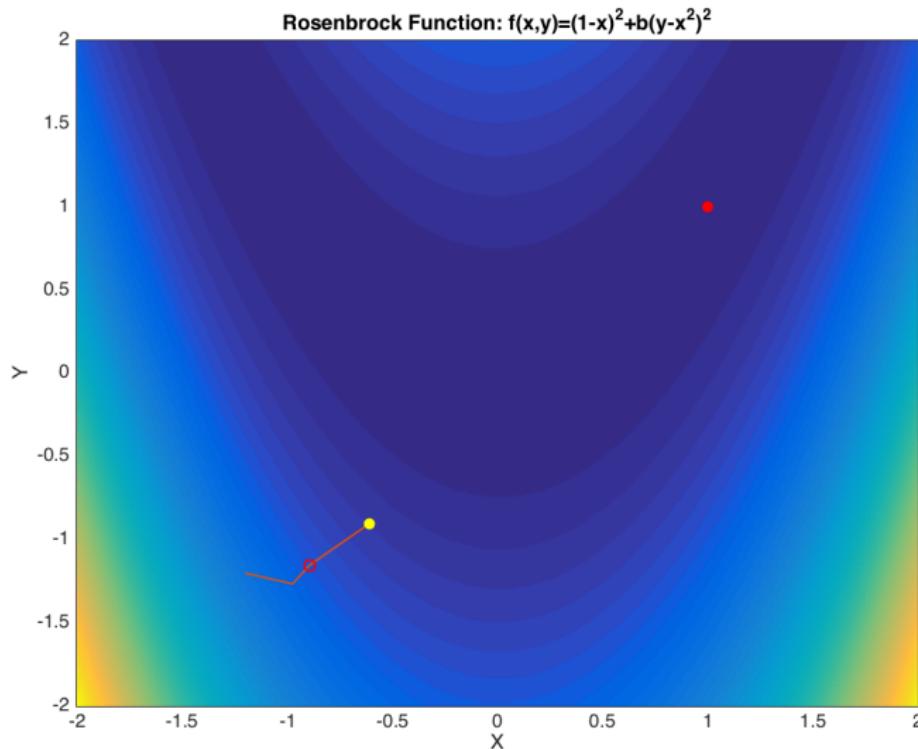
# SIMULATED ANNEALING



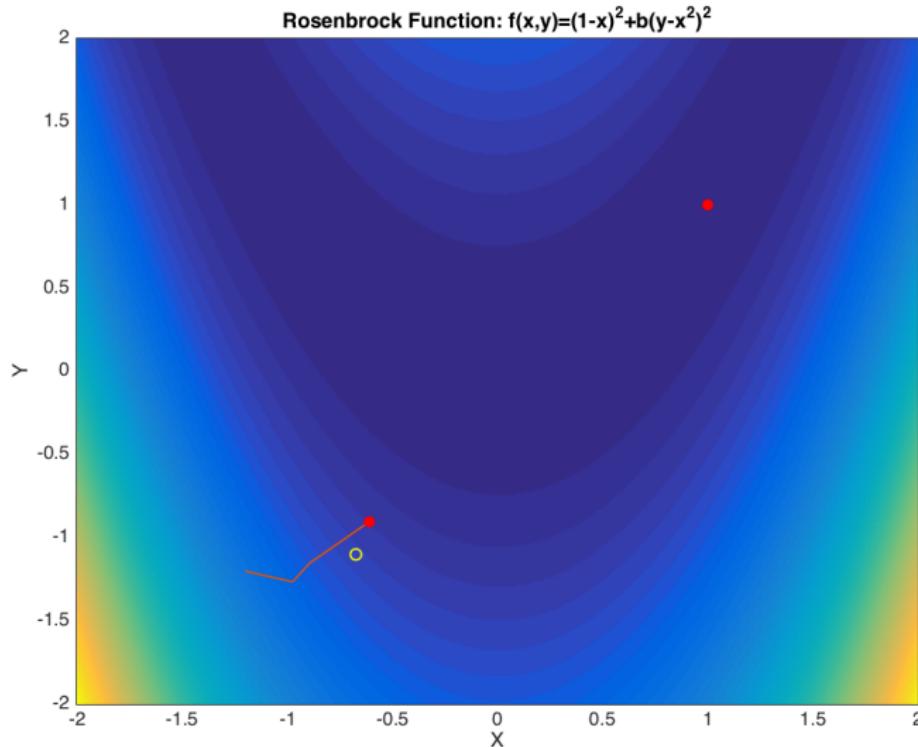
# SIMULATED ANNEALING



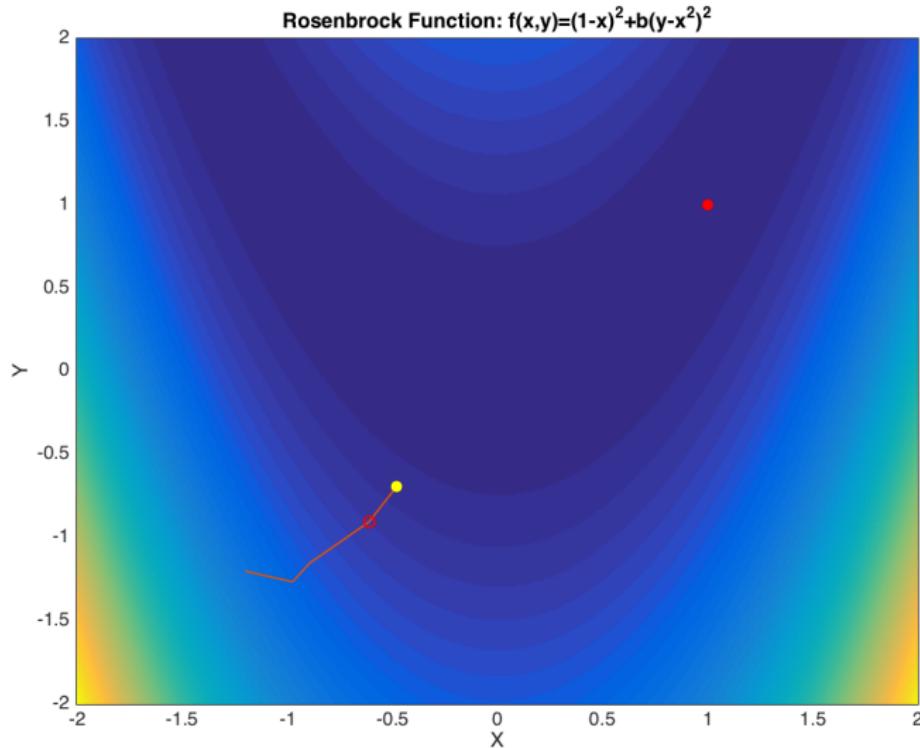
# SIMULATED ANNEALING



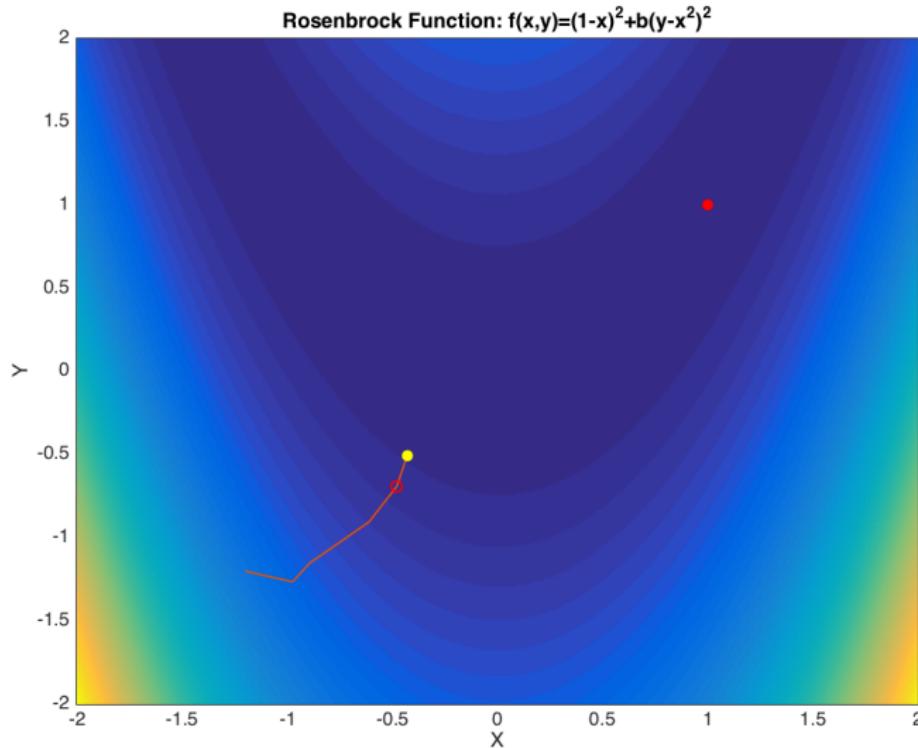
# SIMULATED ANNEALING



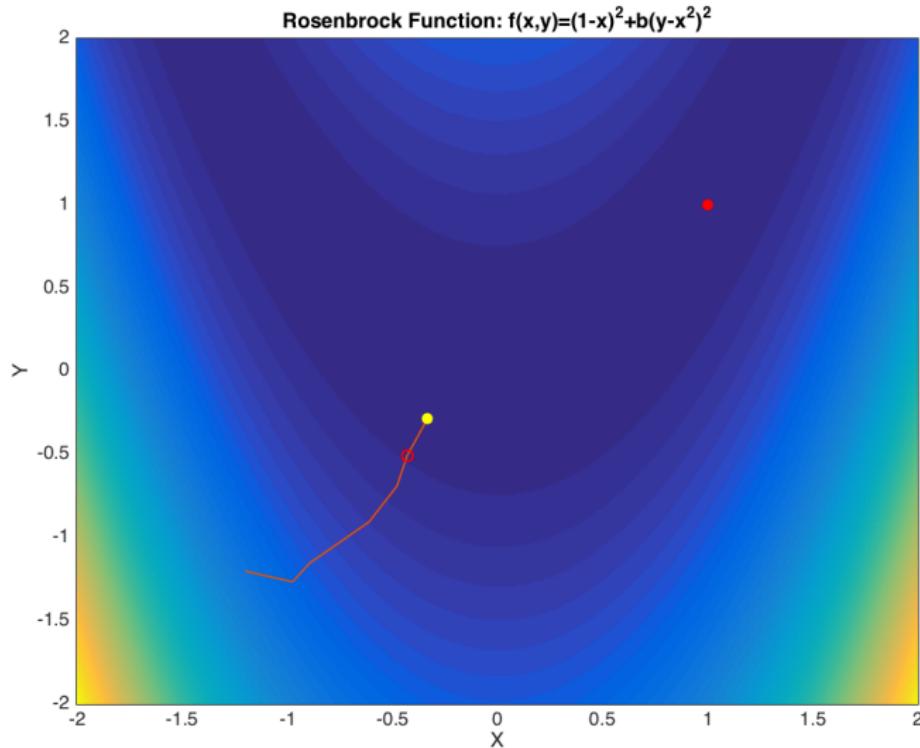
# SIMULATED ANNEALING



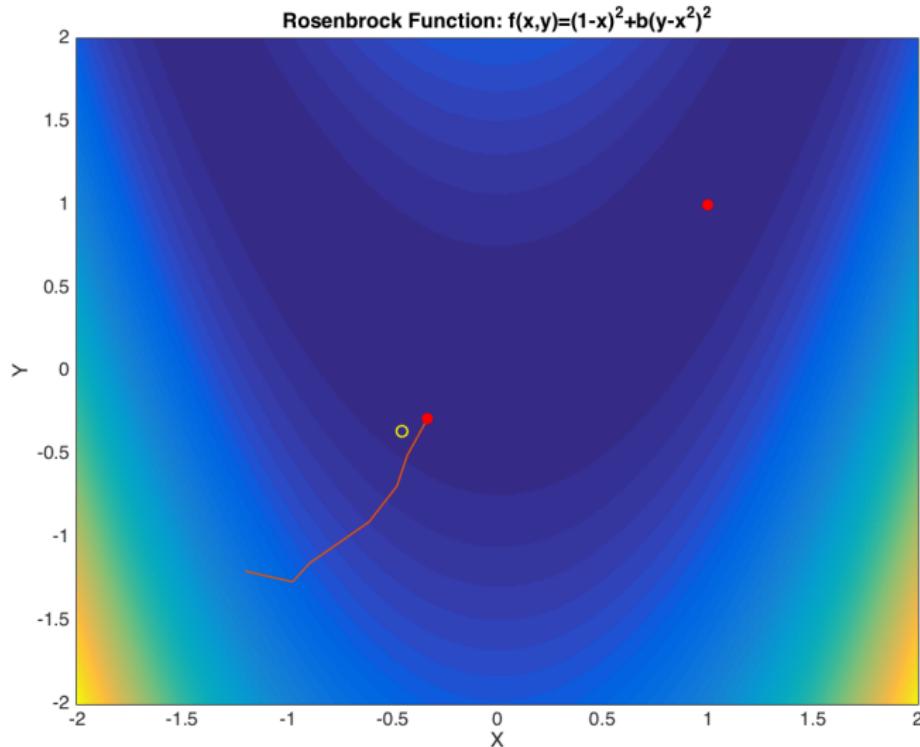
# SIMULATED ANNEALING



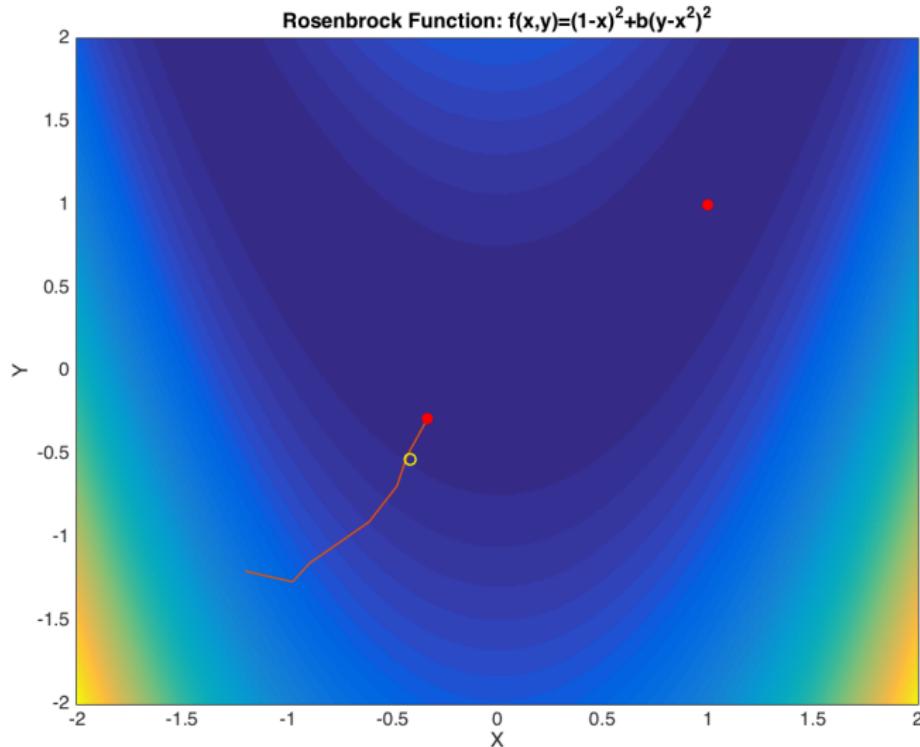
# SIMULATED ANNEALING



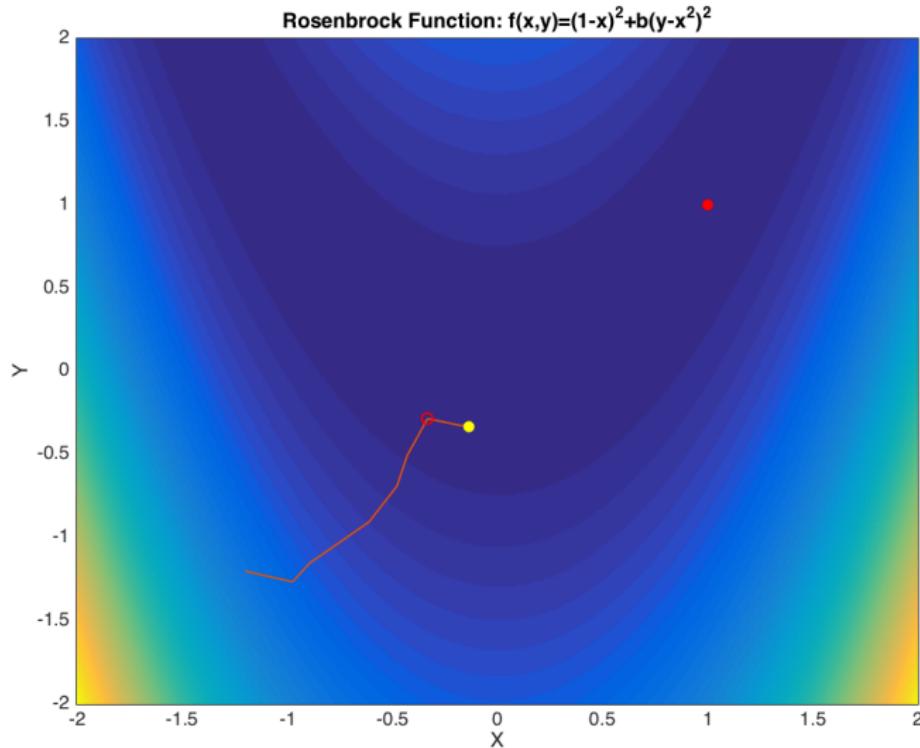
# SIMULATED ANNEALING



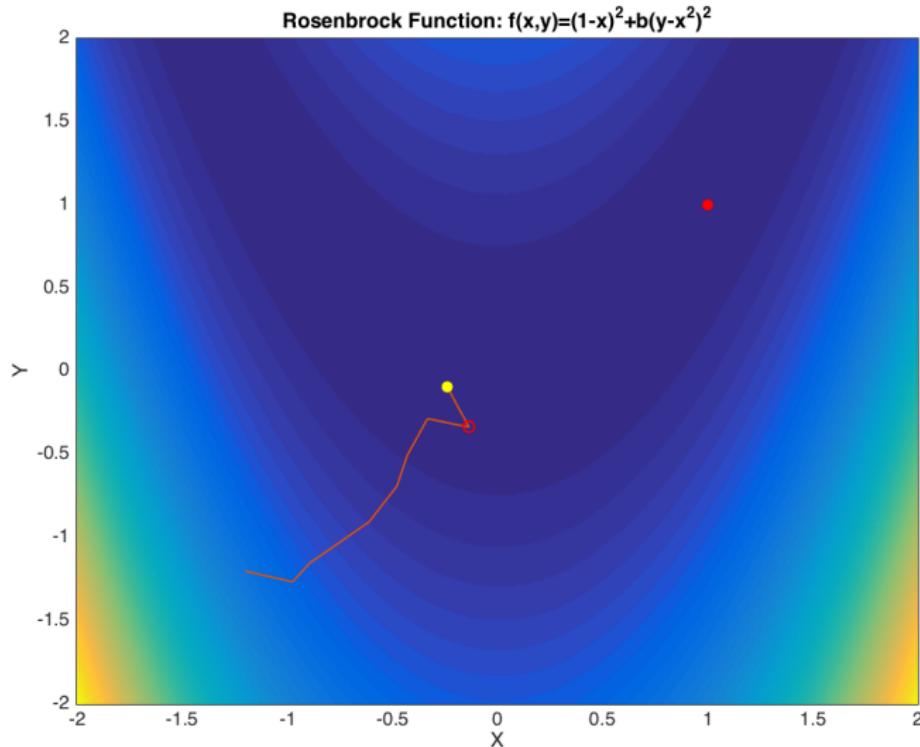
# SIMULATED ANNEALING



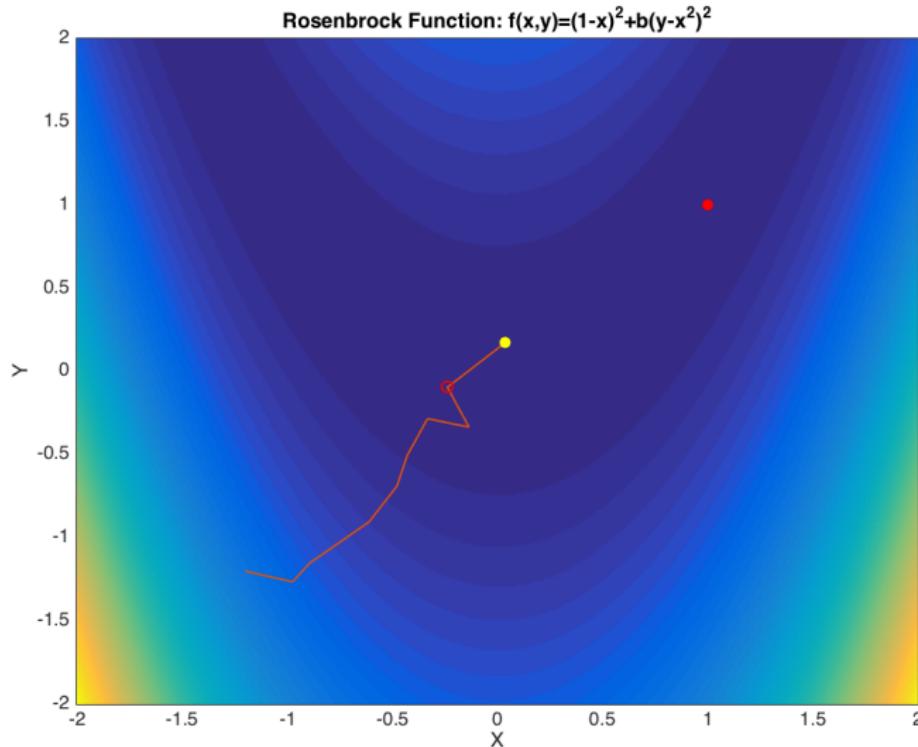
# SIMULATED ANNEALING



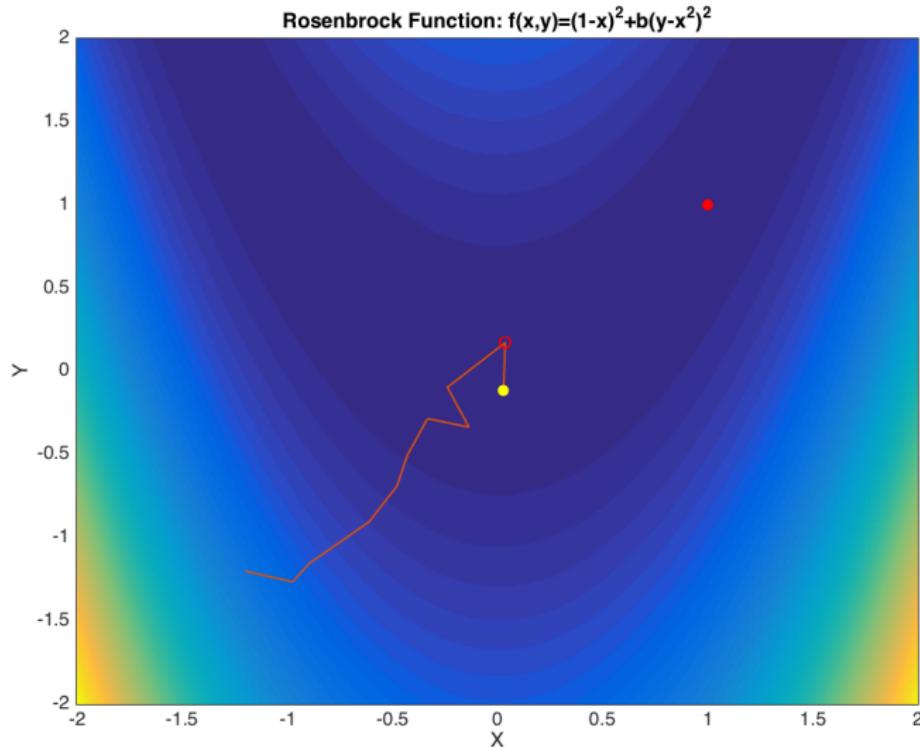
# SIMULATED ANNEALING



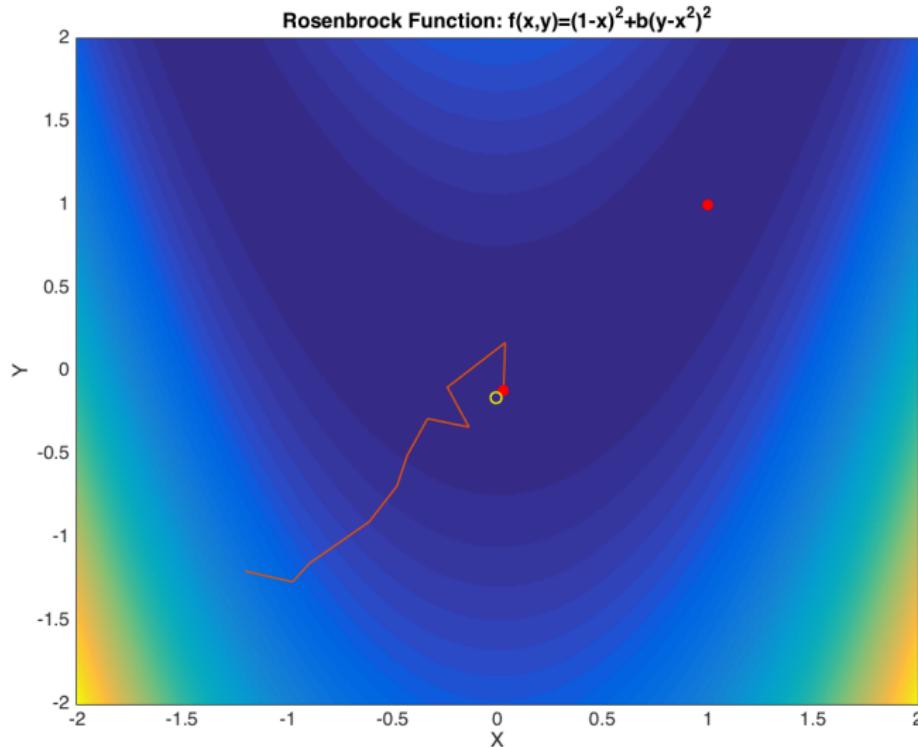
# SIMULATED ANNEALING



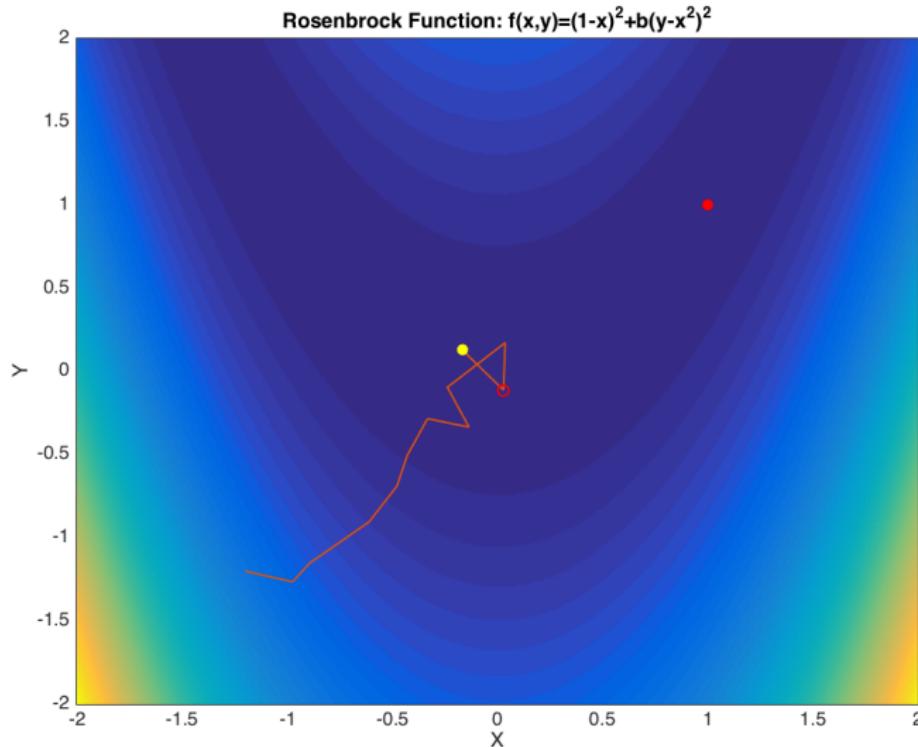
# SIMULATED ANNEALING



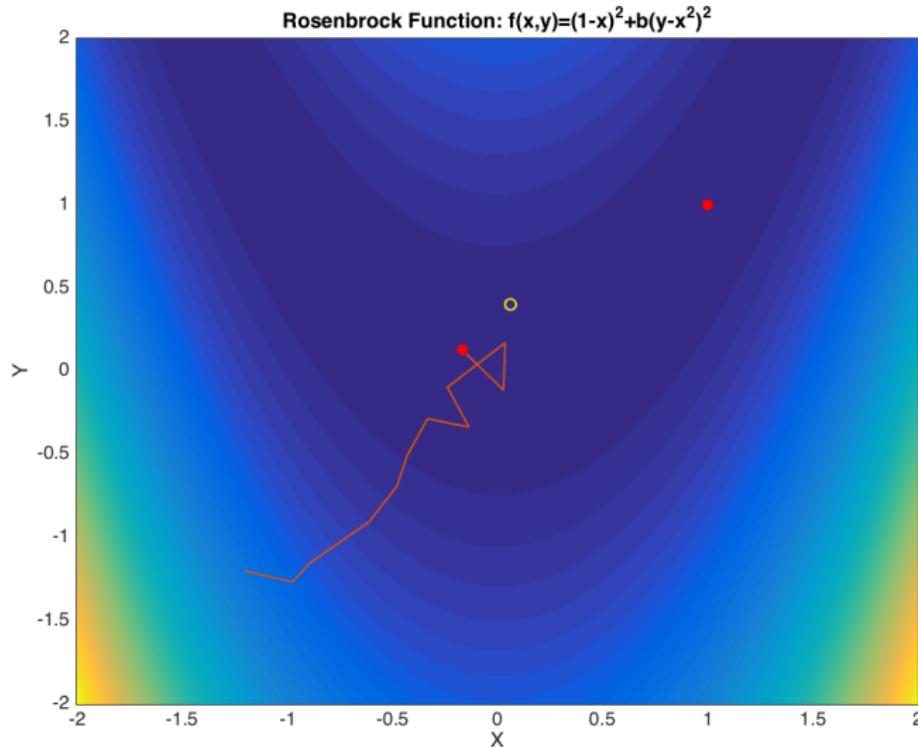
# SIMULATED ANNEALING



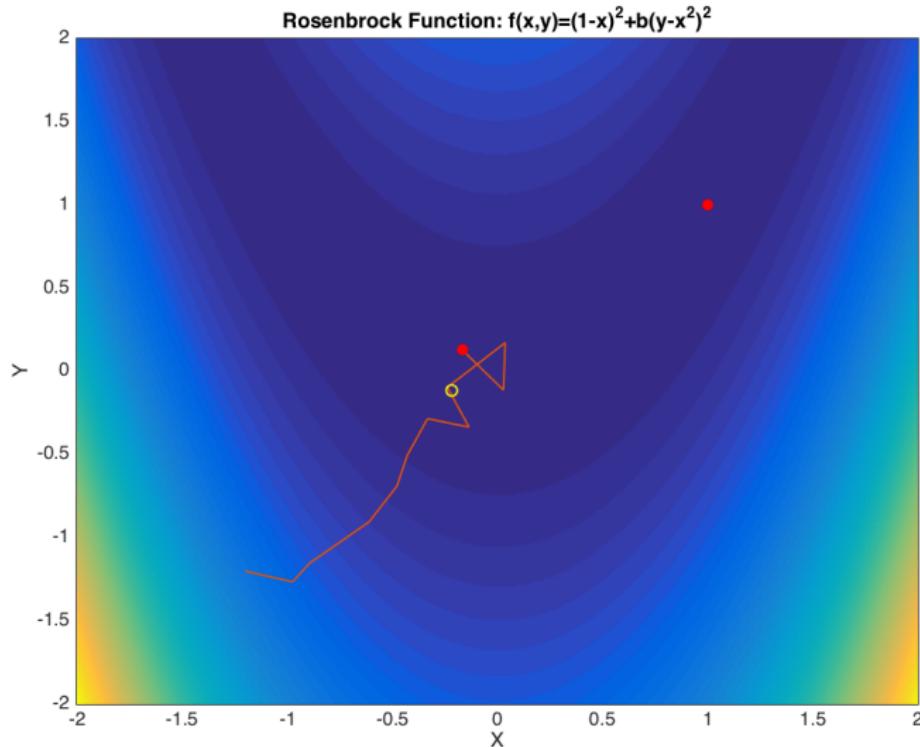
# SIMULATED ANNEALING



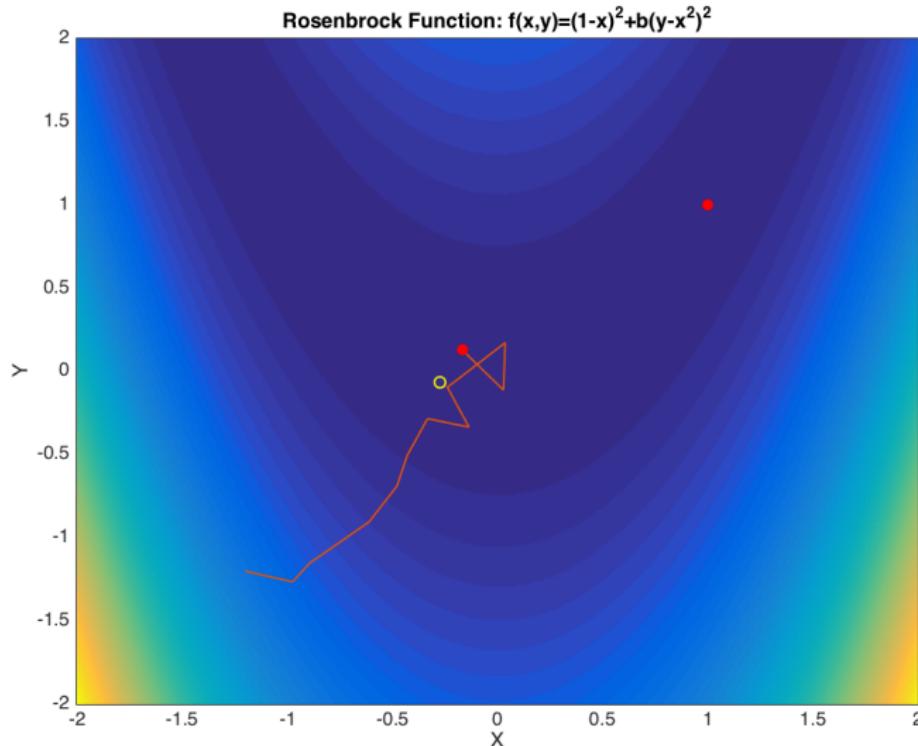
# SIMULATED ANNEALING



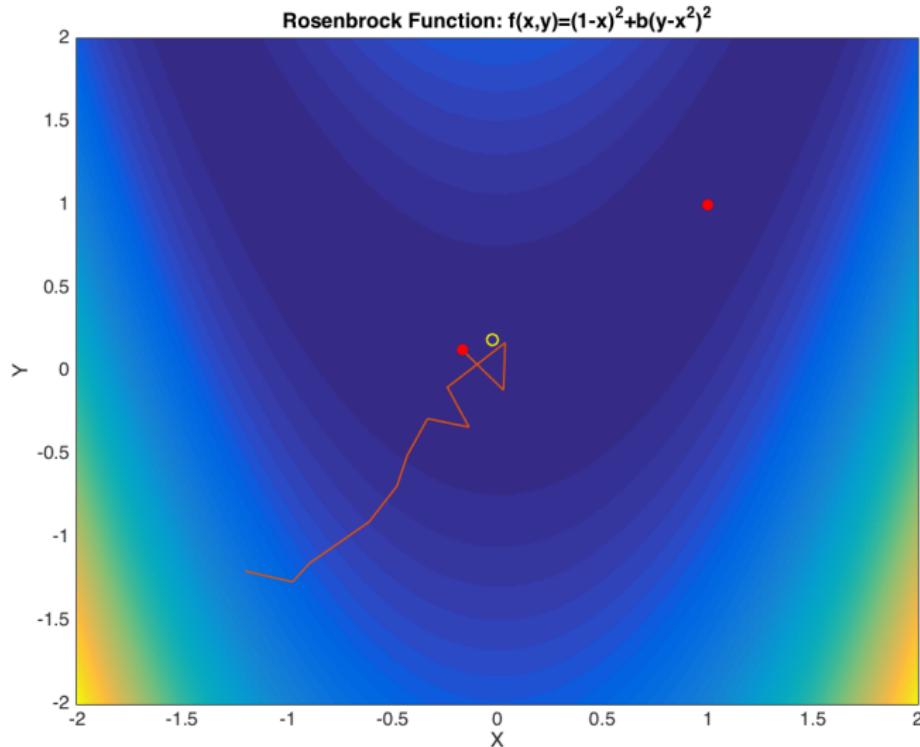
# SIMULATED ANNEALING



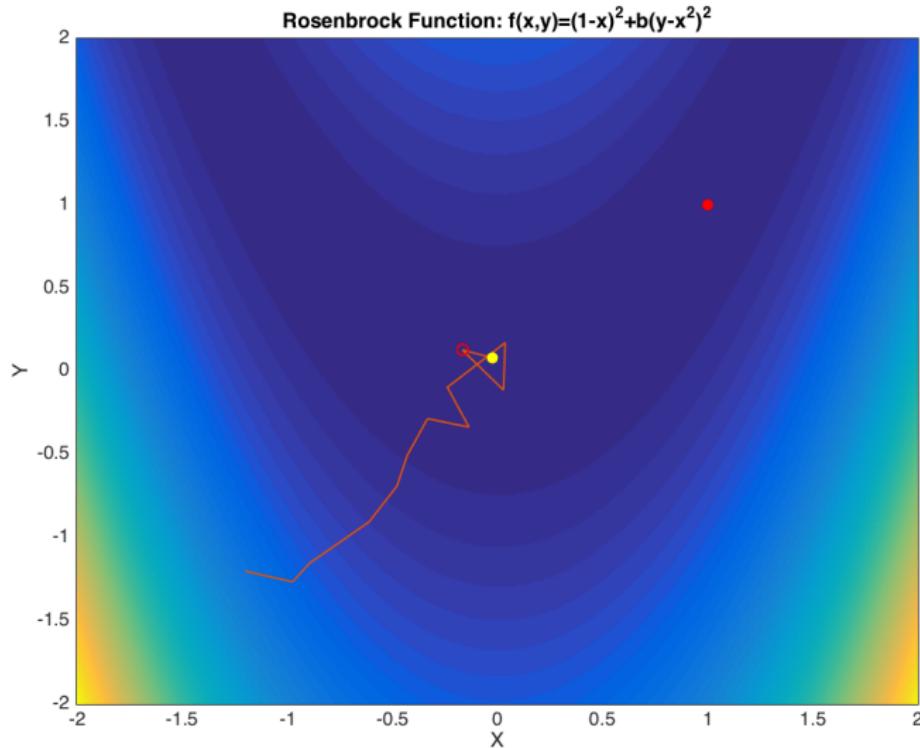
# SIMULATED ANNEALING



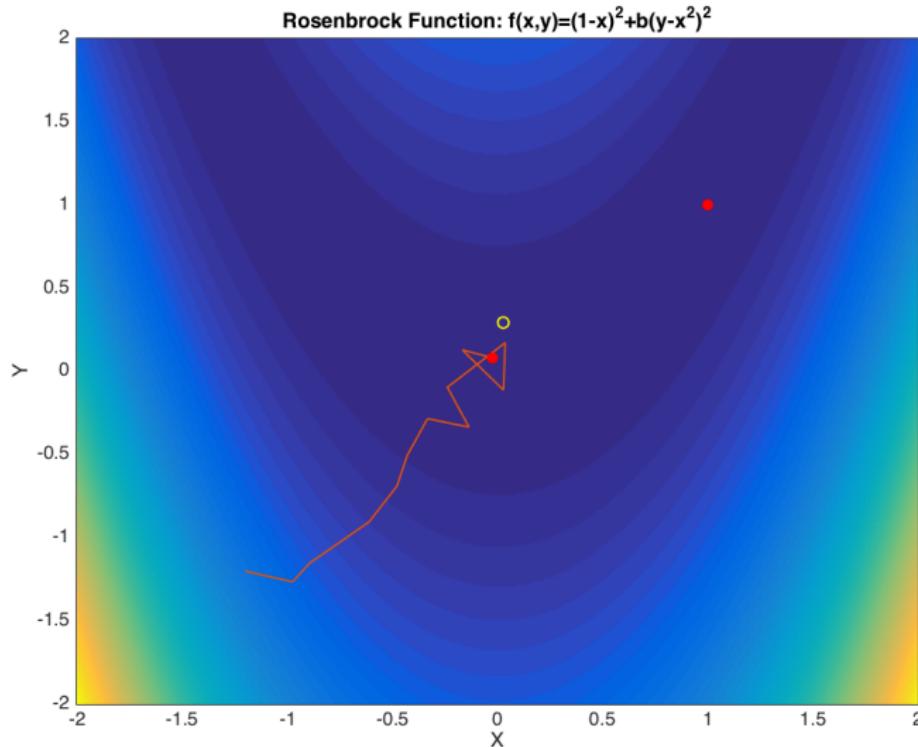
# SIMULATED ANNEALING



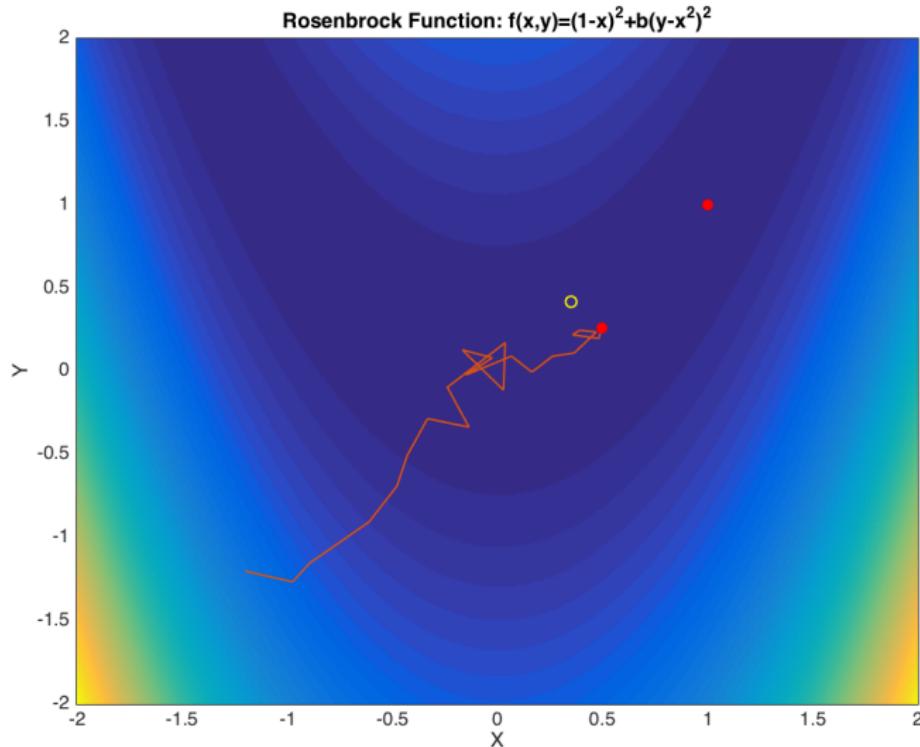
# SIMULATED ANNEALING



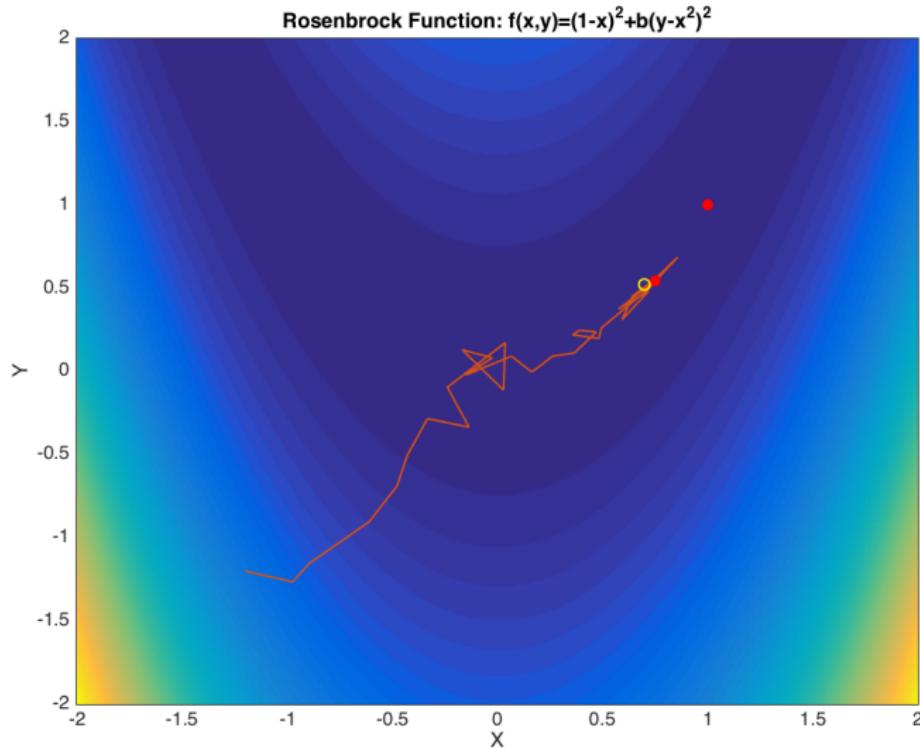
# SIMULATED ANNEALING



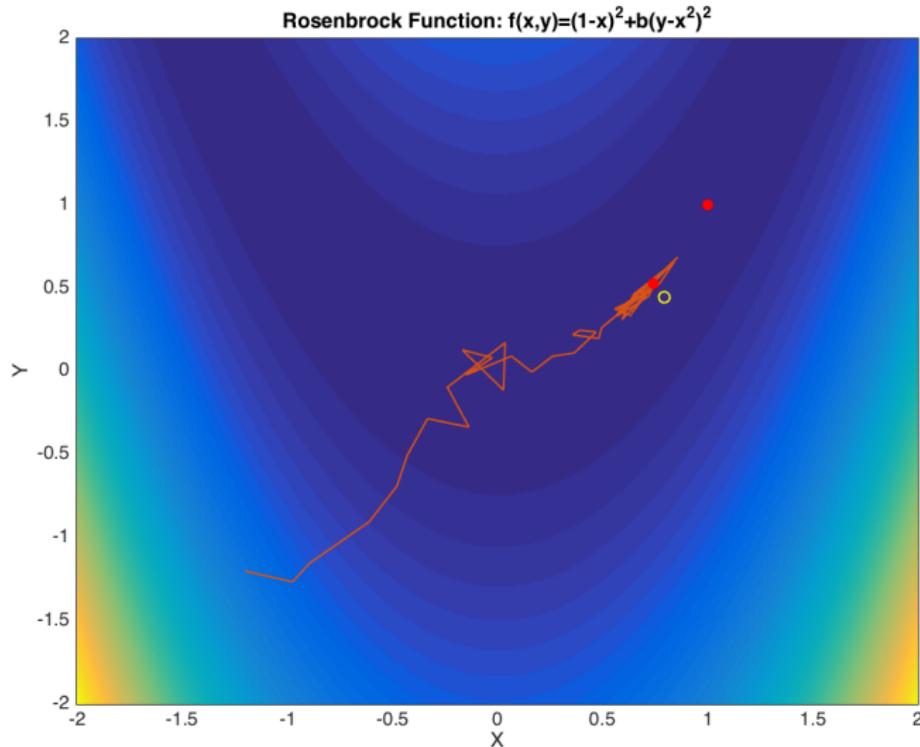
# SIMULATED ANNEALING



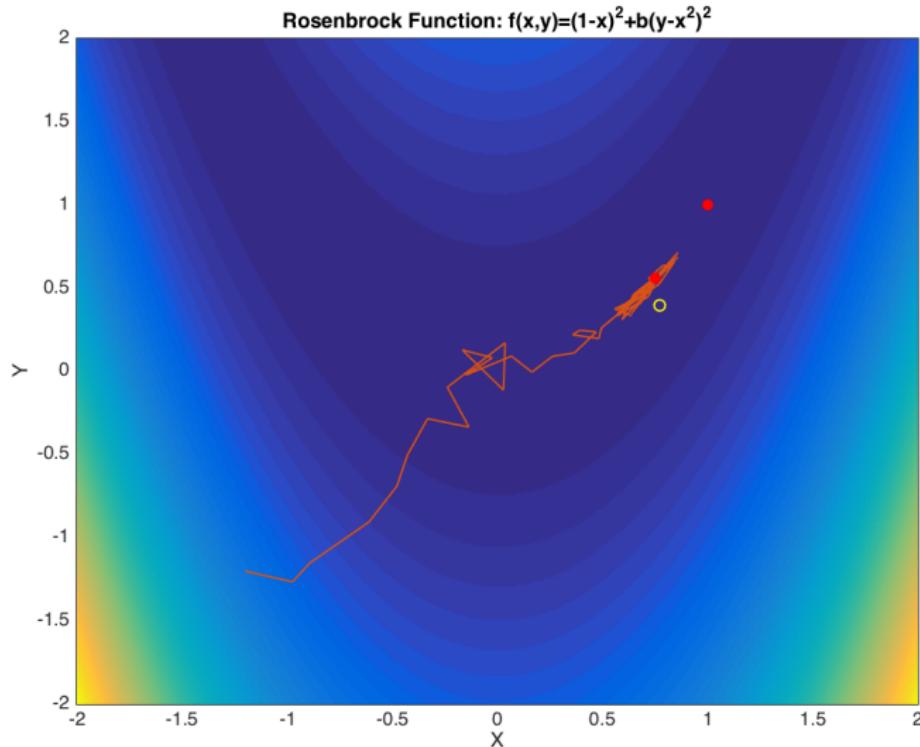
# SIMULATED ANNEALING



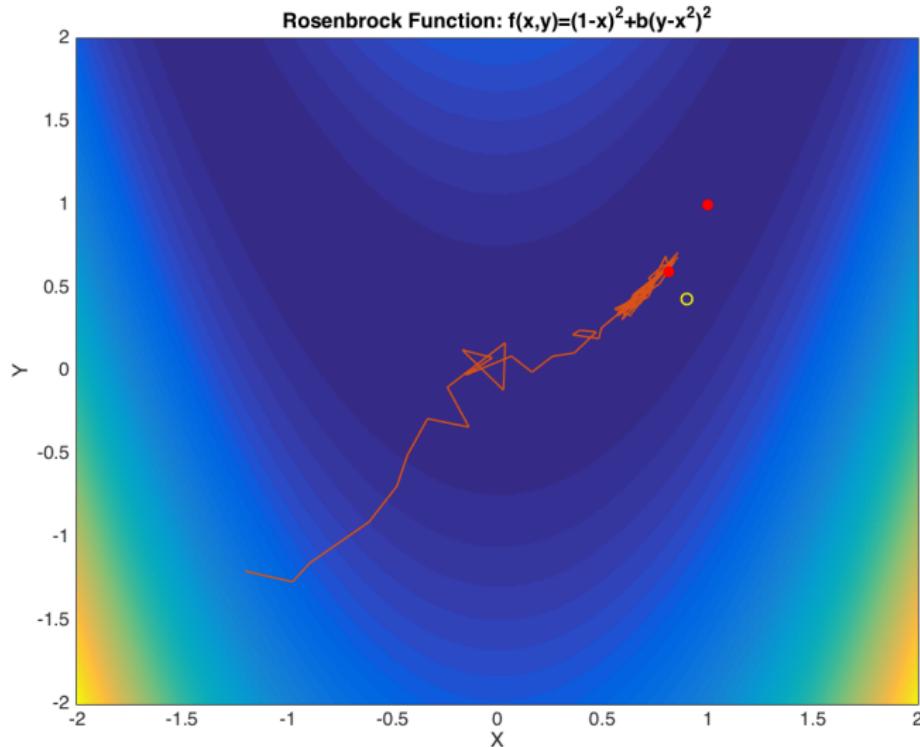
# SIMULATED ANNEALING



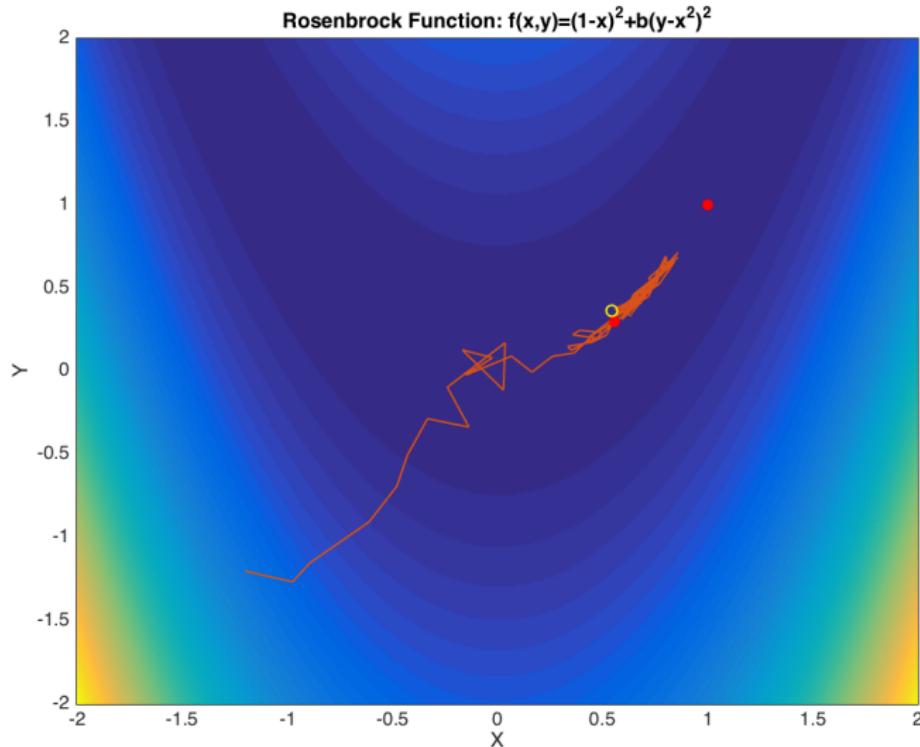
# SIMULATED ANNEALING



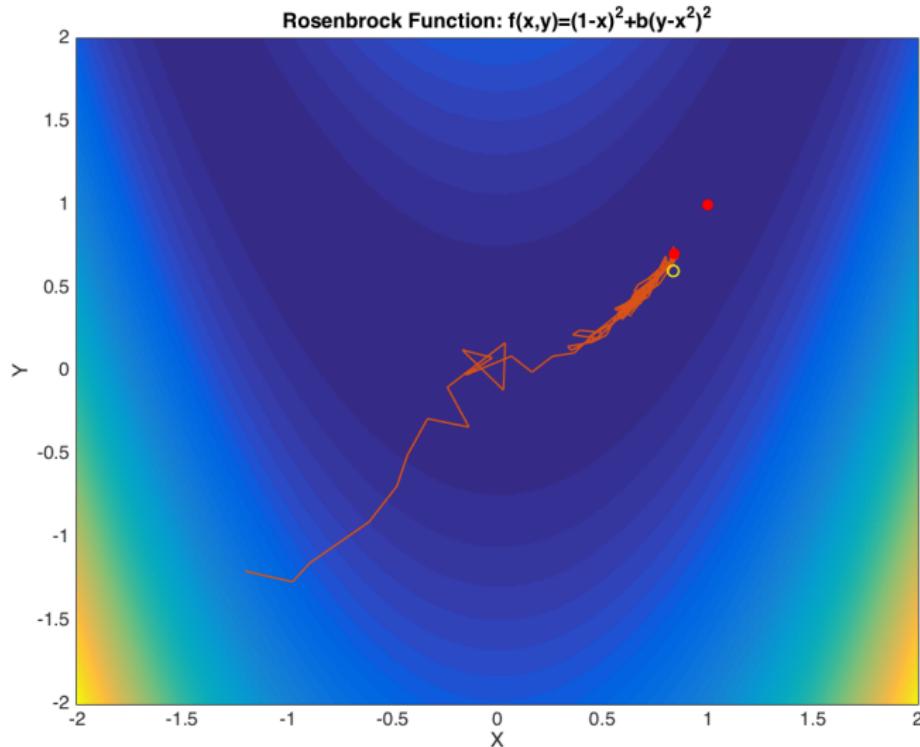
# SIMULATED ANNEALING



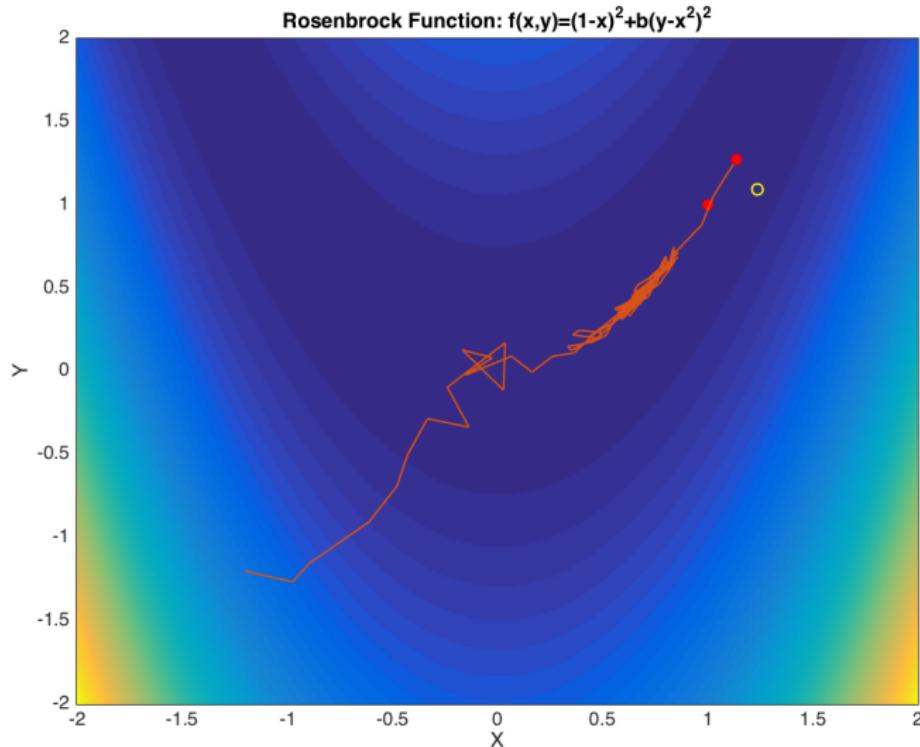
# SIMULATED ANNEALING



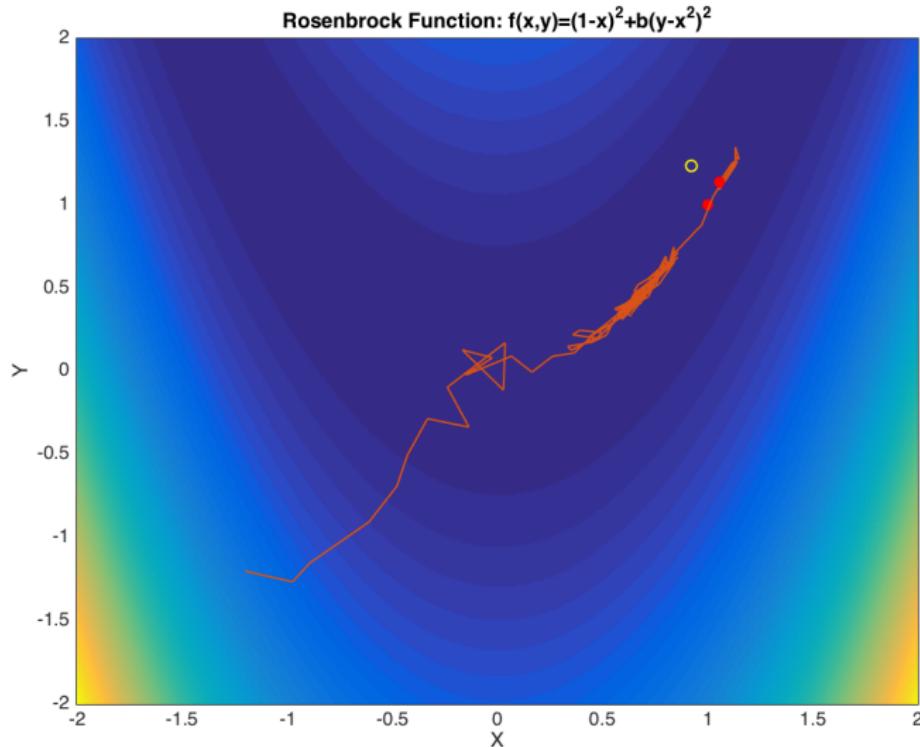
# SIMULATED ANNEALING



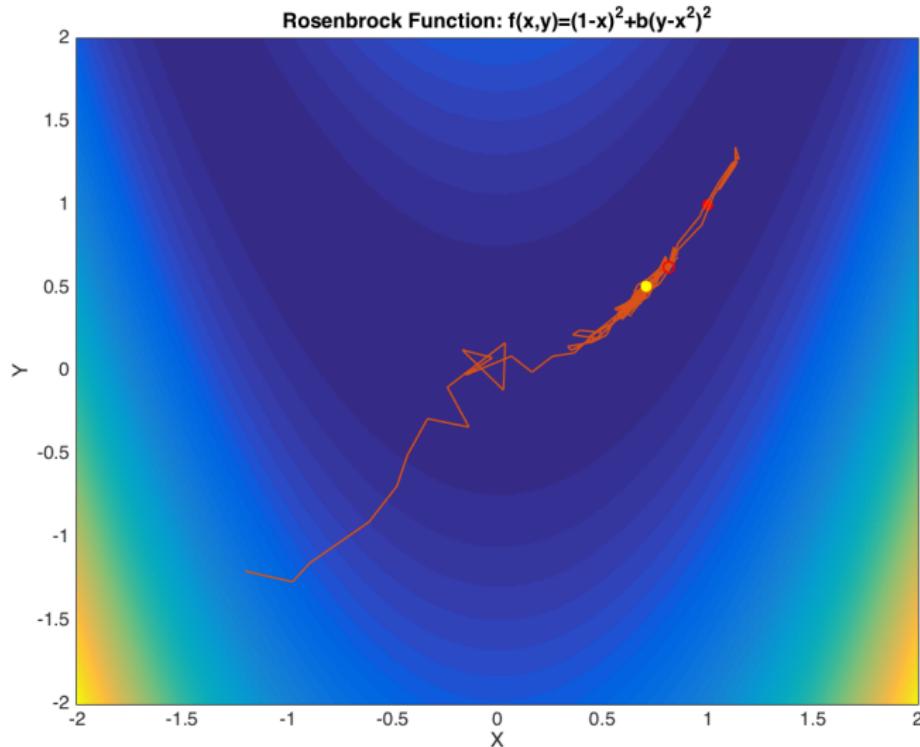
# SIMULATED ANNEALING



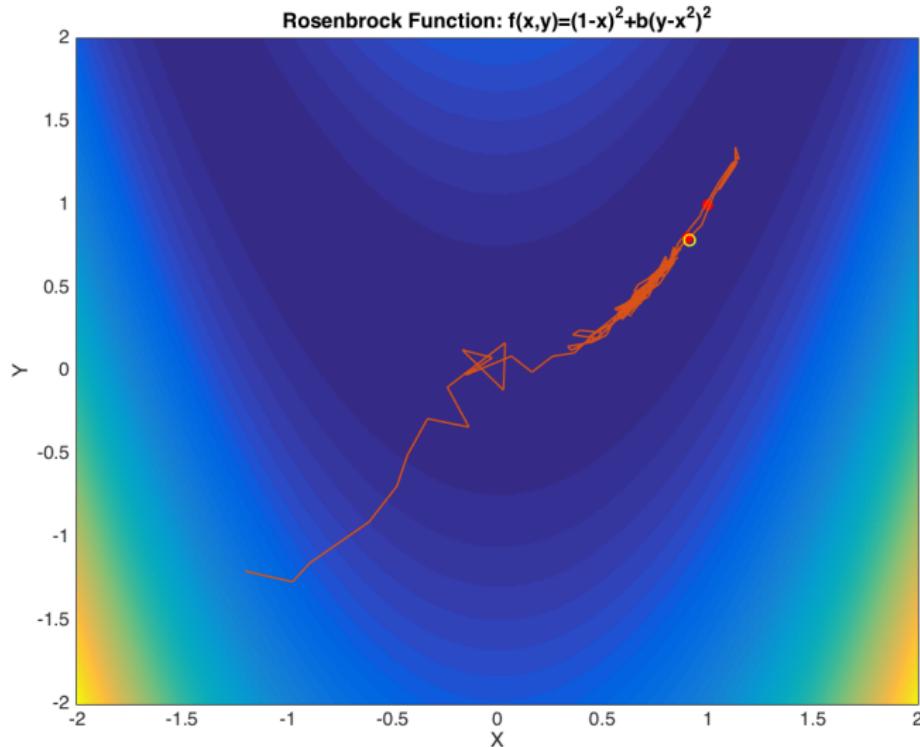
# SIMULATED ANNEALING



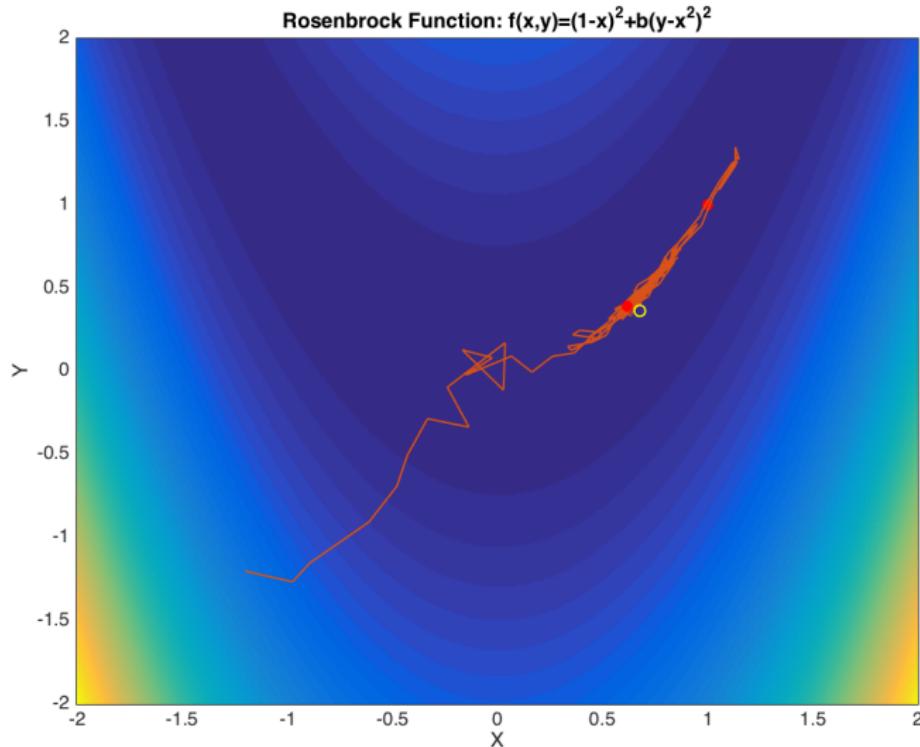
# SIMULATED ANNEALING



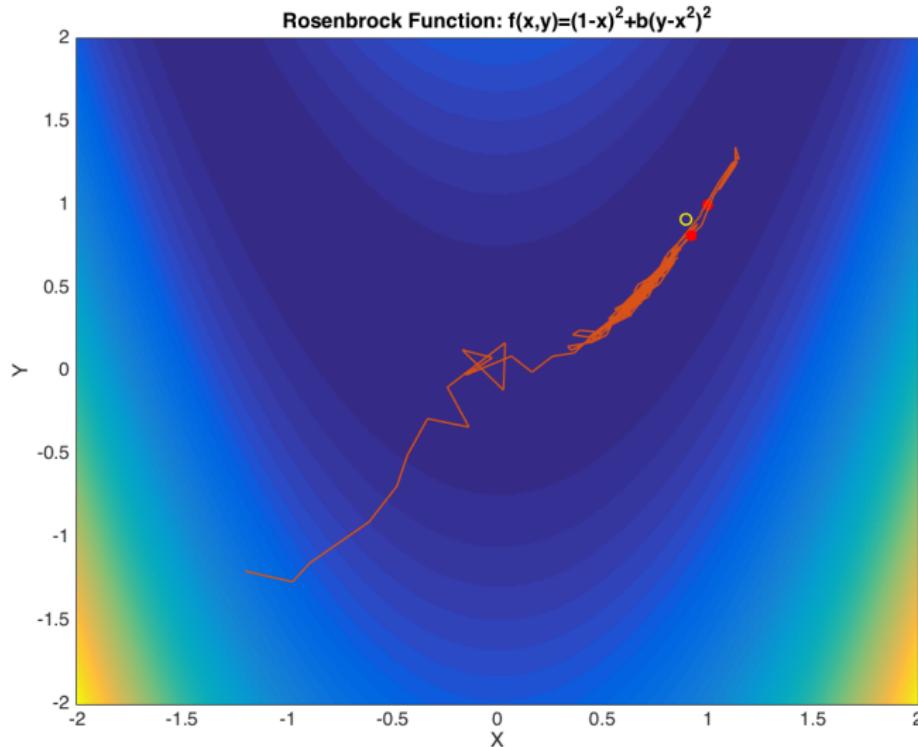
# SIMULATED ANNEALING



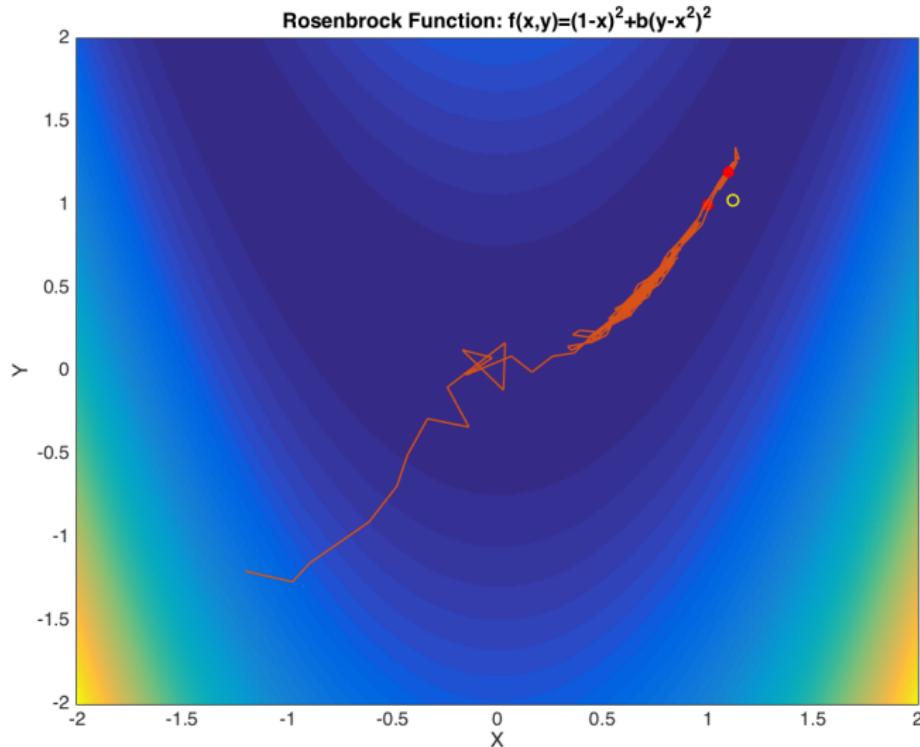
# SIMULATED ANNEALING



# SIMULATED ANNEALING



# SIMULATED ANNEALING



## IMPLEMENTATION

- ▶ Bisection method: `fzero(function,(xlow, xhi))`
- ▶ Golden section search: `fminbnd(function,(xlow, xhi))`
- ▶ Conjugate gradient: some external functions, but `fminunc`, `fmincon`, `lsqnonlin` used for similar purposes: `fminunc(fun,x0)`, `fmincon(fun,x0,...)`, `lsqnonlin(vectorfun,x0,...)`
- ▶ Nelder-Mead: `fminsearch(function,x0)`
- ▶ Pattern Search: `patternsearch(function,x0,...)`
- ▶ Differential Evolution: (external, downloadable function)
- ▶ Genetic Algorithm:  
`ga(fun,length(x0),...,gaoptimset('InitialPopulation',x0))`
- ▶ ParticleSwarm:  
`particleswarm(fun,length(x0),...,gaoptimset('InitialPopulation',x0))`
- ▶ Simulated Annealing: `simulannealbnd(fun,x0)`
- ▶ Multistart, GlobalSearch: can try a method multiple times:  
see “MultiStart” and “GlobalSearch”
- ▶ New: SurrogateOptimization