UNCONSTRAINED OPTIMIZATION: THEORY AND APPLICATIONS: PART ONE

National Institute of Aerospace Teaching and Learning Lab (NIATLL)

National Institute of Aerospace Teaching and Learning Lab (NIATLL)

- Objectives
  - Prepare NIA students for teaching-oriented career paths
  - Facilitate interdisciplinary collaboration via sharing of research skills and knowledge
  - Provide a platform for testing experimental teaching methods
  - Strengthen the graduate experience at the NIA

Optimization—Background

- Optimization encompasses the desire to find the best system, approach, or solution to a problem
- In mathematics, it represents finding the minimum or maximum of a function
- In engineering, it is the search for the design which best meets some figure(s) of merit within the constraints of a challenge

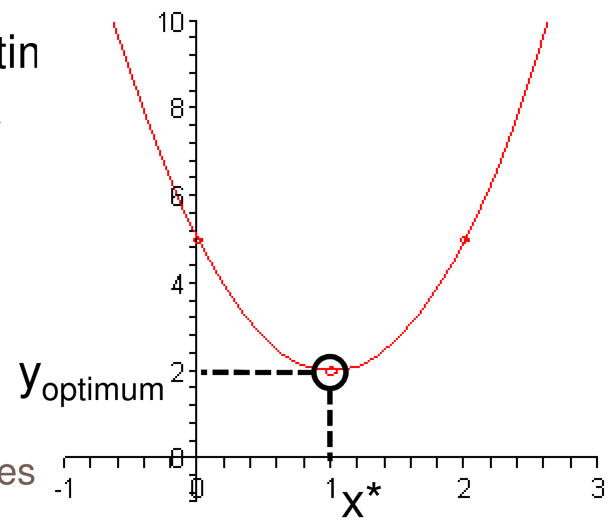
Optimization--Concepts

- For most deterministic methods, optimization consists of a looping algorithm repeated until some stopping criteria is reached
- Each iteration of the loop moves closer to the optimum
- Many methods use information from previous iterations to inform the actions in the next iteration
- The loop is generally stopped when some convergence criteria is met, such as the change in result between iterations being sufficiently small

1D Optimization—Derivative Method

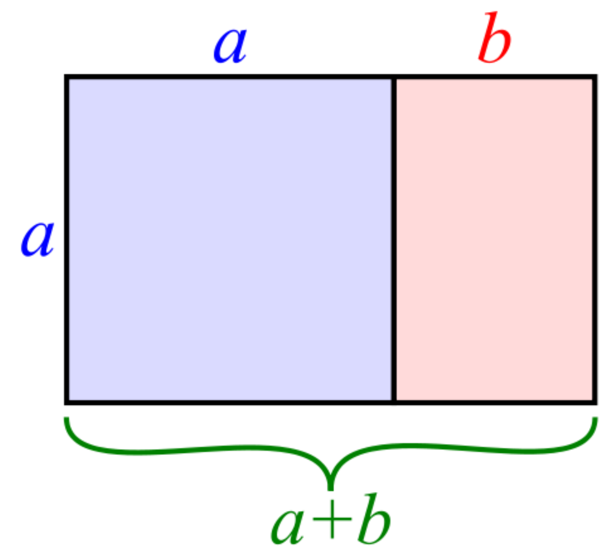
- For continuous, differentiable functions of one variable
  - $y = f(x)$
  - $dy/dx = f'(x)$
  - Set $dy/dx = 0$ and solve for x; call this x^*
  - $y_{optimum} = f(x^*)$

- Example: Polynomial
  - $y = ax^2 + bx + c$
  - $dy/dx = 2ax + b = 0$
  - $x^* = -b/2a$
  - y_{optin} + c



1D Optimization—Golden Section

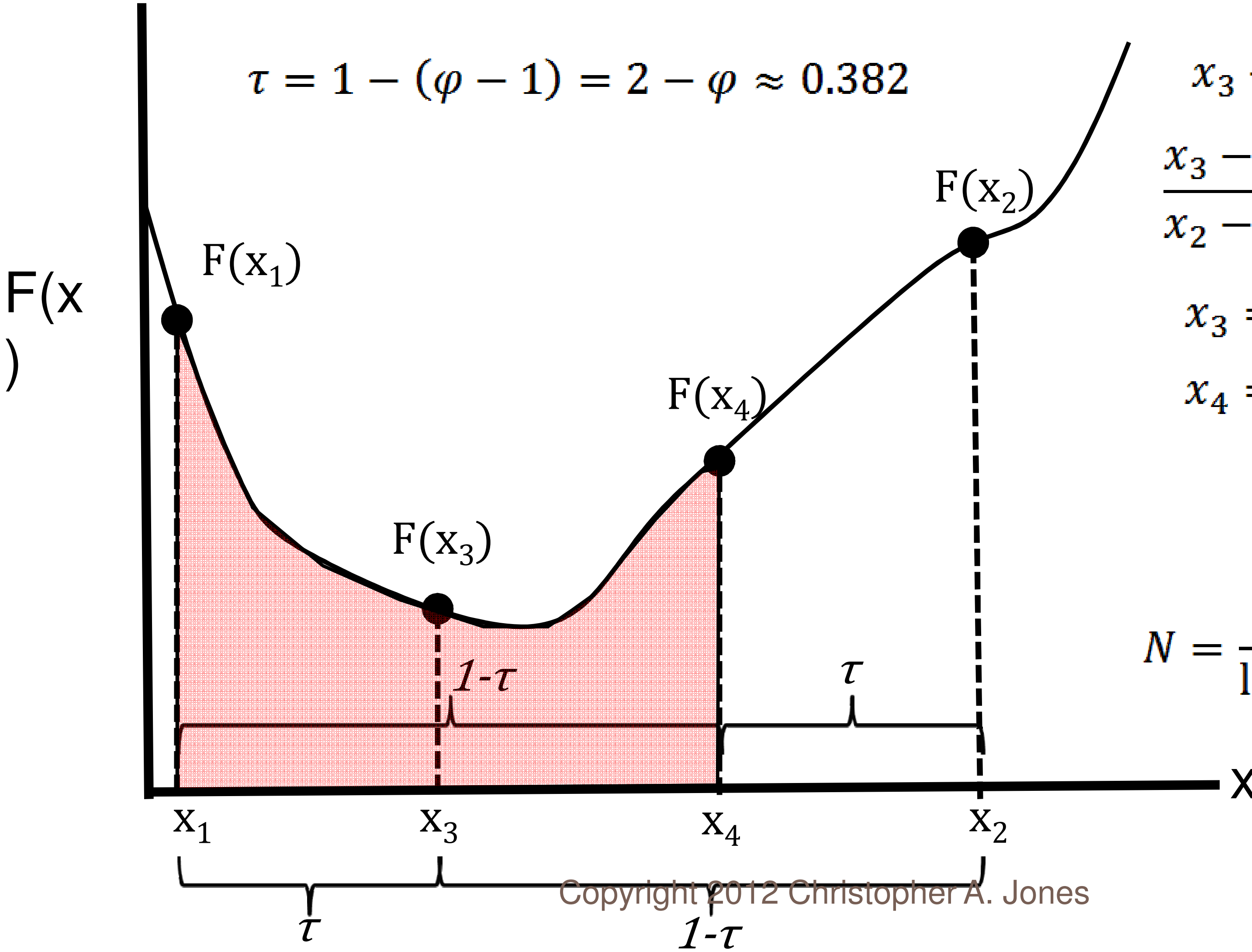
- Golden ratio $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$
- Golden section
  - Searches a bounded part of a function
  - Reduces range by a fixed fraction
  - Can deal with non-continuous functions
  - Has a known rate of convergence



$$\frac{a+b}{a} = \frac{a}{b} = \varphi$$

1D Optimization—Golden Section

$$\tau = 1 - (\varphi - 1) = 2 - \varphi \approx 0.382$$



$$x_3 - x_1 = x_2 - x_4$$

$$\frac{x_3 - x_1}{x_2 - x_1} = \frac{x_2 - x_4}{x_2 - x_1} = \tau$$

$$x_3 = (1 - \tau)x_1 + \tau x_2$$

$$x_4 = \tau x_1 + (1 - \tau)x_2$$

$$N = \frac{\ln(\varepsilon)}{\ln(1 - \tau)} + 3$$

1D Optimization—Golden Section

1. Select initial bounds x_1 and x_2, and compute $F(x_1)$ and $F(x_2)$
2. Initialize x_3 and x_4 using previous formulae, and compute $F(x_3)$ and $F(x_4)$
3. Based on required tolerance ε, determine total number of iterations N; initialize counter K = 3

1D Optimization—Golden Section

4. Loop while $K < N$:
   1. If $F(x_3) > F(x_4)$
      1. $x_1 \leftarrow x_3$
      2. $F(x_1) \leftarrow F(x_3)$
      3. $x_3 \leftarrow x_4$
      4. $F(x_3) \leftarrow F(x_4)$
      5. $x_4 \leftarrow tx_1 + (1-t)x_2$
      6. Compute $F(x_4)$
      7. $K \leftarrow K+1$
   2. Else
      1. $x_2 \leftarrow x_4$
      2. $F(x_2) \leftarrow F(x_4)$
      3. $x_4 \leftarrow x_3$
      4. $F(x_4) \leftarrow F(x_3)$
      5. $x_3 \leftarrow (1-t)x_1 + tx_2$
      6. Compute $F(x_3)$
      7. $K \leftarrow K+1$

Multi-dimensional Optimization

- $y = f(\mathbf{x})$; $\mathbf{x}$ is a vector of inputs
- More complex, with no direct analogue to derivative method
- Three classes of approaches
  - Zero-order: No derivative/gradient information available
  - First-order: Gradient information (analytic or numerical)
  - Second-order: Hessian (second derivative) information

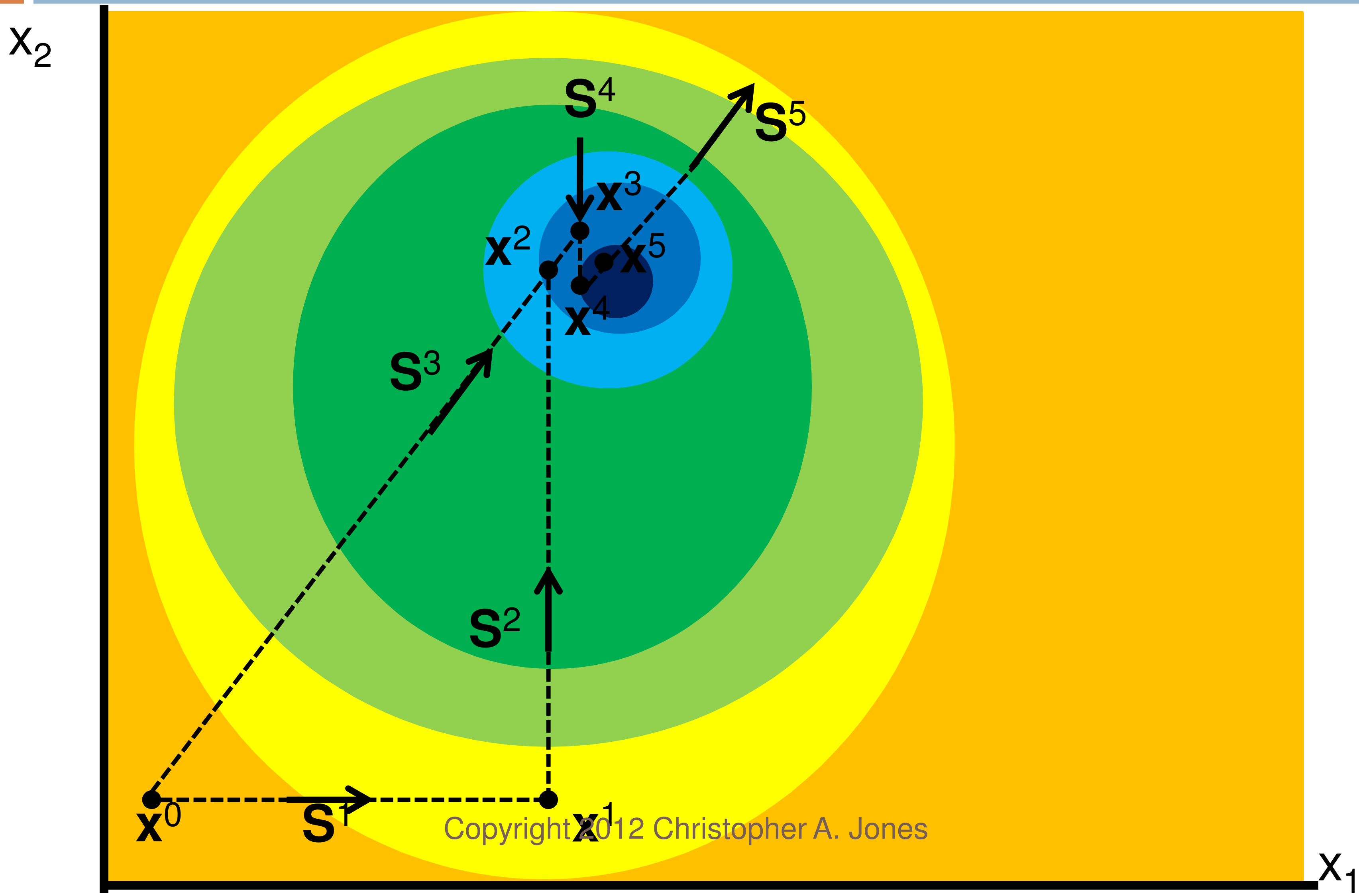
Random Approaches

- Most inefficient, but easiest to implement
- Monte Carlo methods
  - Sample from distributions for each input variable and evaluate f(**x**)
  - Select optimal result from list
- Simple random search
  - Randomly select values for each of xi between lower and upper bounds
  - Evaluate f(**x**), keep if better than current best f(**x**), repeat
  - Continue until maximum number of iterations reached

Deterministic Optimization

- General equation: $\mathbf{X}^q = \mathbf{X}^{q-1} + \alpha^* \mathbf{S}^q$
  - $\mathbf{X}^q$ = New point to evaluate
  - $\mathbf{X}^{q-1}$ = Previous point to evaluate
  - α^* = Scalar governing step size
  - $\mathbf{S}^q$ = Search direction
  - q = count of current iteration
- Optimization strategies consist of choosing the methods to pick α^* (what line search to use) and $\mathbf{S}^q$ (how to determine what direction to search)

Powell's Method



Powell's Method

1. Select an initial point $\mathbf{x}^0$ (vector of length n)
2. Perform a line search sequentially in each of n unit directions (i.e. [1,0] followed by [0,1]); store each direction as a column in n-by-n matrix $\mathbf{H}$
3. Determine the conjugate direction $\mathbf{S}^{n+1}$ by computing $\mathbf{x}^n - \mathbf{x}^0$ and performing a line search to find $\mathbf{x}^{n+1}$
4. Set $\mathbf{x}^0$ equal to $\mathbf{x}^{n+1}$ and append $\mathbf{S}^{n+1}$ to the right side of $\mathbf{H}$, then delete the leftmost column of $\mathbf{H}$
5. Repeat steps 2—4 until convergence is met

Powell Pseudo-code from MATLAB

1. Initialize x_0, **y**, and n-by-n identity matrix **H**
2. Outer Loop (while outer counter < max iterations)
    1. Set q = 1
    2. Inner Loop (for q = 1 to n)
        1. Line search with Golden Section in direction **H**(:,q)
        2. $\mathbf{x} = \mathbf{x} + \alpha^* \mathbf{H}(:,q)$
    3. $\mathbf{S}_{new} = \mathbf{x} - \mathbf{y}$
    4. Line search with Golden Section in direction $\mathbf{S}_{new}$
    5. Check convergence, break if abs(norm($\mathbf{S}_{new}$)) < tolerance
    6. Append $\mathbf{S}_{new}$ to **H** and delete **H**(:,1)