**Numerical Optimization**

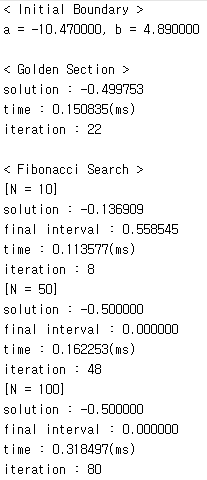
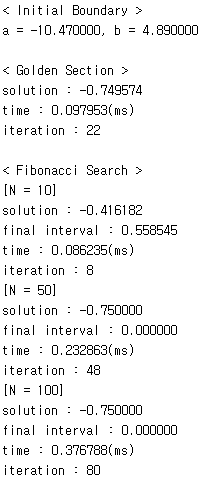
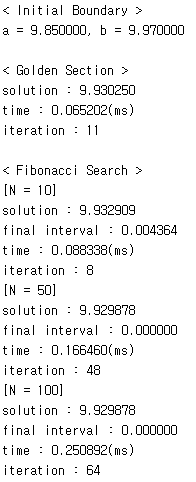
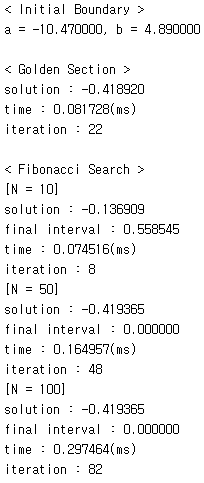
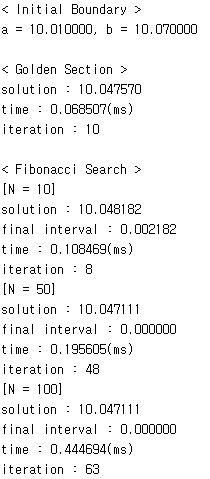
**Programming #2**

**Electrical Engineering & Computer Science**

**20171014 Young-Chul Yoon**

1. **Introduction  
   1) Goal of the assignment  
   -** Implement algorithm that seeks initial bound  
   - Implement Fibonacci search and Golden section search algorithms  
   - Discuss their comparative performance for at least five different problems

**2) Description about implementation and setting**I implemented my code using MATLAB. I implemented 3-algorithms in separate function to make the code looks better. For comparing performance, I used time and iteration as performance factor. Detailed settings are mentioned below.  
**1) Seeking initial bound function**  
- initial x : 10  
- d0 : 0.01  
**2) Golden section search**  
- rho : 0.6180  
- interval tolerance : 1.0e-3  
- return solution : b  
**3) Fibonacci search**  
- N : 10, 50, 100  
- Stop condition : (x1 == x2)  
- return solution : b  
**< 5 problems >**  
1)   
2)   
3)   
4)   
5)

1. **Result in program**
2. 
3. 
4. 
5. 
6. 
7. **Analysis**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem** | **Initial point** | **Method** | **Solution** | **interval** | **iteration** |
|  | [-10.47, 4.89] | Golden section | -0.499753 | < 10e-3 | 22 |
| Fibonacci (N=10) | -0.136909 | 0.558545 | 8 |
| Fibonacci (N=50) | -0.5 | 0 | 48 |
| Fibonacci (N=100) | -0.5 | 0 | 80 |
|  | [-10.47, 4.89] | Golden section | -0.749574 | < 10e-3 | 22 |
| Fibonacci (N=10) | -0.416182 | 0.558545 | 8 |
| Fibonacci (N=50) | -0.75 | 0 | 48 |
| Fibonacci (N=100) | -0.75 | 0 | 80 |
|  | [9.85, 9.97] | Golden section | 9.930250 | < 10e-3 | 11 |
| Fibonacci (N=10) | 9.932909 | 0.004364 | 8 |
| Fibonacci (N=50) | 9.929878 | 0 | 48 |
| Fibonacci (N=100) | 9.929878 | 0 | 64 |
|  | [-10.47, 4.89] | Golden section | -0.418920 | < 10e-3 | 22 |
| Fibonacci (N=10) | -0.136909 | 0.558545 | 8 |
| Fibonacci (N=50) | -0.419365 | 0 | 48 |
| Fibonacci (N=100) | -0.419365 | 0 | 82 |
|  | [10.01, 10.07] | Golden section | 10.047570 | < 10e-3 | 10 |
| Fibonacci (N=10) | 10.048182 | 0.002182 | 8 |
| Fibonacci (N=50) | 10.047111 | 0 | 48 |
| Fibonacci (N=100) | 10.047111 | 0 | 63 |

The result shows that selecting N is critical to performance of Fibonacci search..  
The process of convergence is different according to pre-defined N. So, it is hard to apply Fibonacci search in problems which requires specific stopping criterion. In other words, it is hard to determine which N is proper to find the solution of the problem. In my experiment, N = 10 is too small to find solution. And N = 100 is too big, which need redundant time to find solution. Also, there could exist N, which is smaller than 50 and have same accuracy with 50. But it is hard to determine such N without several trial.  
 Instead, Golden section search is applicable to problem which requires stopping criterion. It doesn’t need pre-defined sequences like Fibonacci search and the golden ration(rho) 0.6180 is invariable. So, it is much clear algorithm to estimate the number of iteration for finding solution.

1. **Code Implementation**

**< Setting and Main function >**

|  |
| --- |
| %% Copyright (C) 2017 Young-Chul Yoon  %% All rights reserved.  %% Optimization Homework  %% Fibonacci, Golden section search  %% 5 functions  %f = @(x) x^2 + x;  %f = @(x) x^4 + x^3 + 1;  %f = @(x) 3\*sin(x^2) + x^2 + x;  %f = @(x) exp(x^2) + x;  f = @(x) 5\*sin(x^4) + 3\*cos(x^3) + x;  %% Seeking bound algorithm  [a, b] = bound\_seeking(f);  if f(a) == f(b)  return;  end  fprintf('< Initial Boundary >\n');  fprintf('a = %f, b = %f\n\n', a, b);  %% Golden section search  [x\_gol, t\_gol, it\_gol] = golden\_section(f, a, b);  fprintf('< Golden Section >\n');  fprintf('solution : %f\n', x\_gol);  fprintf('time : %f(ms)\n',t\_gol\*1000);  fprintf('iteration : %d\n\n', it\_gol);  %% Fibonacci search for N = 10, 50, 100  N = [10, 50, 100];  fprintf('< Fibonacci Search >\n');  for i=1:length(N)  [x\_fib, interv, t\_fib, it\_fib] = fibonacci\_search(f, a, b, N(i));  fprintf('[N = %d]\n', N(i));  fprintf('solution : %f\n', x\_fib);  fprintf('final interval : %f\n', interv);  fprintf('time : %f(ms)\n', t\_fib\*1000);  fprintf('iteration : %d\n', it\_fib);  end |

**< Bound seeking >**

|  |
| --- |
| function [a, b] = bound\_seeking(f)  %% Copyright (C) 2017 Young-Chul Yoon  %% All rights reserved.  %% Initial point and d0  init\_x = 10;  d0 = 0.01;  f\_m = f(init\_x - d0);  f0 = f(init\_x);  f\_p = f(init\_x + d0);  d = 0;  x = [0, init\_x, 0];  %% Set the sign of d  if f\_m > f0 & f0 > f\_p  d = d0;  x(1) = init\_x - d0;  x(3) = init\_x + d0;  elseif f\_m < f0 & f0 < f\_p  d = -d0;  x(1) = init\_x + d0;  x(3) = init\_x - d0;  elseif f\_m > f0 & f0 < f\_p  a = init\_x - d0;  b = init\_x + d0;  return;  elseif f\_m == f\_p  fprintf('f\_m == f\_p == f0\n');  a = init\_x - d0;  b = init\_x + d0;  return;  else  fprintf('init\_x is maximum point\n');  return;  end  k = 1;  while true  if f(x(3))>f(x(2))  if d>0  a = x(1);  b = x(3);  else  a = x(3);  b = x(1);  end  return;  end  x\_sav = x;  %% Expand bound range using power of 2  x(3) = x\_sav(3)+(2^k)\*d;  x(2) = x\_sav(3);  x(1) = x\_sav(2);  k = k+1;  end  end |

**< Golden section search >**

|  |
| --- |
| function [x, t, it] = golden\_section(f, a, b)  %% Copyright (C) 2017 Young-Chul Yoon  %% All rights reserved.  tic;  rho = 0.6180;  tol = 1.0e-3;  %% Iterate until interval of two points is less than tolerance  it = 1;  while true  if (abs(a-b) < tol)  break;  end  x1 = a + (1-rho)\*(b-a);  x2 = a + rho\*(b-a);  if f(x2)>f(x1)  b = x2;  else  a = x1;  end  it = it+1;  end  t = toc;  x = b;  end |

**< Fibonacci search >**

|  |
| --- |
| function [x, interv, t, it] = fibonacci\_search(f, a, b, N)  %% Copyright (C) 2017 Young-Chul Yoon  %% All rights reserved.  tic;  %% Generate N fibonacci numbers  f\_num = [];  f\_num = [f\_num, 1];  f\_num = [f\_num, 1];  for i=3:N  f\_num = [f\_num, (f\_num(i-1) + f\_num(i-2))];  end  %% Iterate maximum N epoch  it = 1;  for i=N:-1:3  L = b-a;  x1 = a+(f\_num(i-2)/f\_num(i))\*L;  x2 = b-(f\_num(i-2)/f\_num(i))\*L;  if x1 == x2  break;  elseif f(x1)>f(x2)  a = x1;  else  b = x2;  end  it = it+1;  end  t = toc;  x = b;  interv = b-a;  end |