우산 2차 설계 : 우산살의 최적화

Anti Stress

2004008125 박경식
2007004528 송정용

Contents

Problem Description

- 알루미늄으로 우산살을 만들려고 하는데 원가는 가장 싸면서 정면에서 바람이 불 때 **14m/s** 우산 끝이 **45**도 이상 휘지 않고 뒤로 받을때 우산이 항복되지 않는 범위에 들도록 우산살의 두께와 우산의 지지핀의 거리를 결정하여라.

  - **Design Variable : 'r' and 'd'**
  - **Object Function : cost**

Problem Statement

1. **Design Variable**
    1. 우산중심과 지지핀과의 거리
    2. 우산살의 두께(반지름)
2. **Constraint**
    1. 바람이 정면으로 불때 **slope**가 **45**도보다 작다.
    2. 바람이 반대로 볼때 항복응력 **280MPa**보다 작다.
    3. **Safe factor = 1 (**안전상에 큰 문제가 없는 물건이다.**)**

Problem Statement

3. Object

: 비를 맞지 않는 넓이의 최대

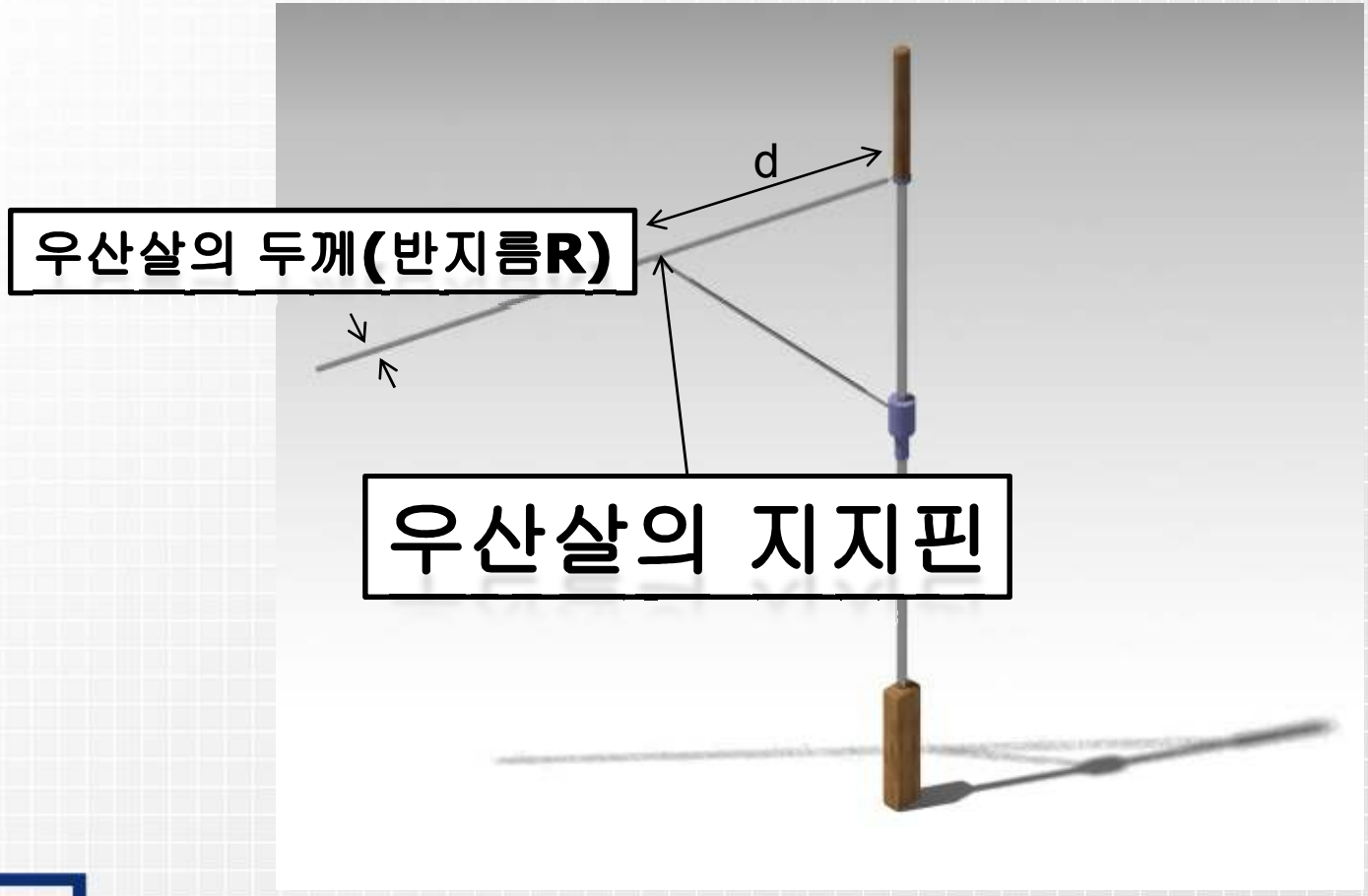
$$Cost = n\rho_1 c_1 \pi r^2 \left(l + \frac{k \sin\theta}{\sin 80^\circ} \right) + \alpha$$

ρ_1 = 알루미늄합금의 밀도

C_1 = 알루미늄합금의 가격

α=우산대,손잡이,방수천에 드는 비용

Project/Problem Symetric



Mathematical Model

1. Object Functon

$$Cost = n\rho_1 c_1 \pi r^2 \left(l + \frac{k \sin\theta}{\sin 80^\circ} \right) + \alpha$$

$n = 8$(우산살의 수)

α = 우산대(니켈)와 방수천,
우산손잡이(절연체)등의 예상가격
4249.5원

2. Constraint Function

$$g_1 = \frac{w_0 \left\{ \frac{l^2}{2}\left(\frac{1}{2} - \frac{1}{3d}\right) l^2 + \frac{l^2}{6d}(l-d)^2 - \frac{1}{24} l^4 + \left(\frac{l^2 d}{18} - \frac{l^2 d^2}{12} + \frac{1}{120} d^4 \right) \right\}}{EI} - \frac{\pi}{4} \le 0$$

$$g_2 = \frac{(l-d)p}{\pi r^2}\left(2 + \sqrt{\frac{4}{r^2} + \frac{1}{d^2}} \right) - 2.8e{+}08 \le 0$$

$l = 0.7m$

$\theta = 80^\circ$

L과 Θ는 1차때 구한
결과값으로 고정

The solution of Problem

- **Step of Problem's Solution**
  1. Find profit Algorithm for Solution
     1. Excel solve
     2. Genetic Algorithm
     3. Pattern Search
     4. Fmincon
  2. Solve
  3. analysis

1. Excel 해찾기

◆Find solution of excel (해 찾기)

Parameters	
n	8
ro1	2770
ro2	8900
c1	2048.15
c2	18290.75
c3	1481.48
l	0.7
seta	1.396263
seta2	1.396263
alpha	0.07
w0	103

해 찾기

constraint	g1	g2
	-42293250.3	-2.55432E-07

cost	5529.46962

design variables	초기값	해찾기 후
r	0.0015	0.002924
k	0.2	0.35

1. Excel solve

◆Find solution of excel (해 찾기 시나리오)

$r = 0.002999$ $k = 0.35$ 으로 수렴한 해찾기 시나리오

1	2	3	4	5	6	7
0.1	0.05	0.046072	0.023086	0.007943	0.003588	0.002999
0.3	0.2975	0.35	0.35	0.35	0.35	0.35

$r = 0.003565$ $k = 0.1$ 으로 수렴한 해찾기 시나리오

1	2	3	4	5	6	7
0.002	0.052	0.027	0.010616	0.004174	0.003857	0.003565
0.1	0.1	0.1	0.1	0.1	0.1	0.1

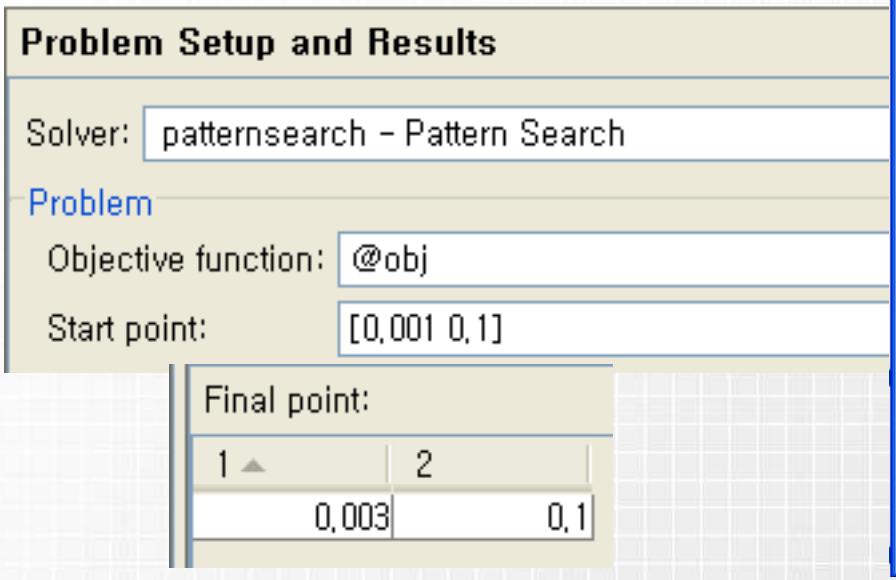
2. Ganetic Algorithm

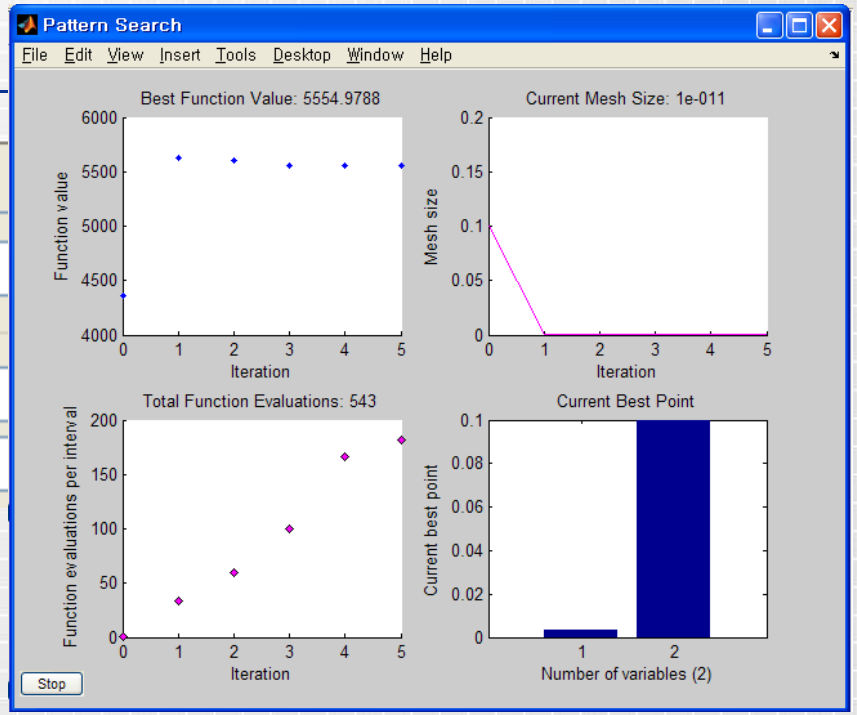
iteration	f	x	y		
1	5576.468	0.00329	0.140569		
2	5557.979	0.003346	0.105051		
3	5554.992	0.003354	0.1		
Best	max	Stall			
Generatio	f-count	f(x)	constraint	Generations	
	1	5312	2.59E+10	6.23E-09	0
	2	10824	23937.5	6.54E-11	0
	3	16024	8347.94	6.54E-11	0
4	5555.003	0.003354	0.1		

❖ Constraint 를 적용한 genetic algorithm

- **Constraint**의 사용 가능한 범위내에서 인구**50**으로 찾았으며 값을 찾는데 횟수는 적었으나 그 값이 할 때 마다 바뀌었다.
- 인구수와 횟수를 늘렸지만 값이 잘나오지 않아서 **Pattern search** 와 **hybrid** 시켜서 계산함

3. Pattern Search

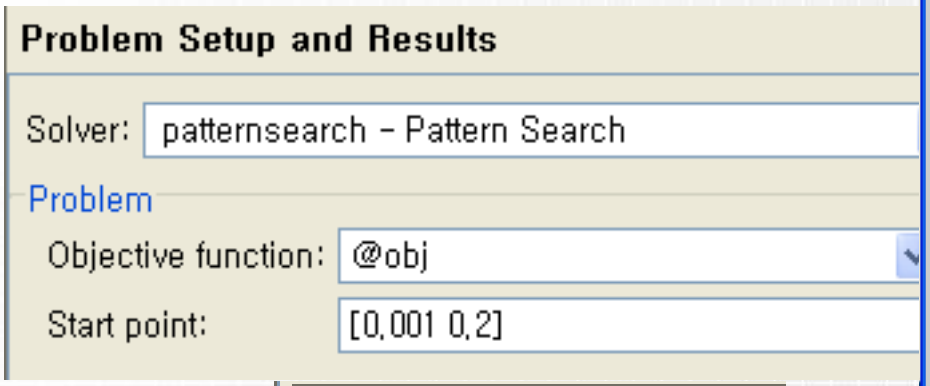


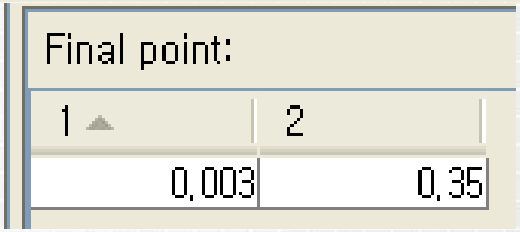


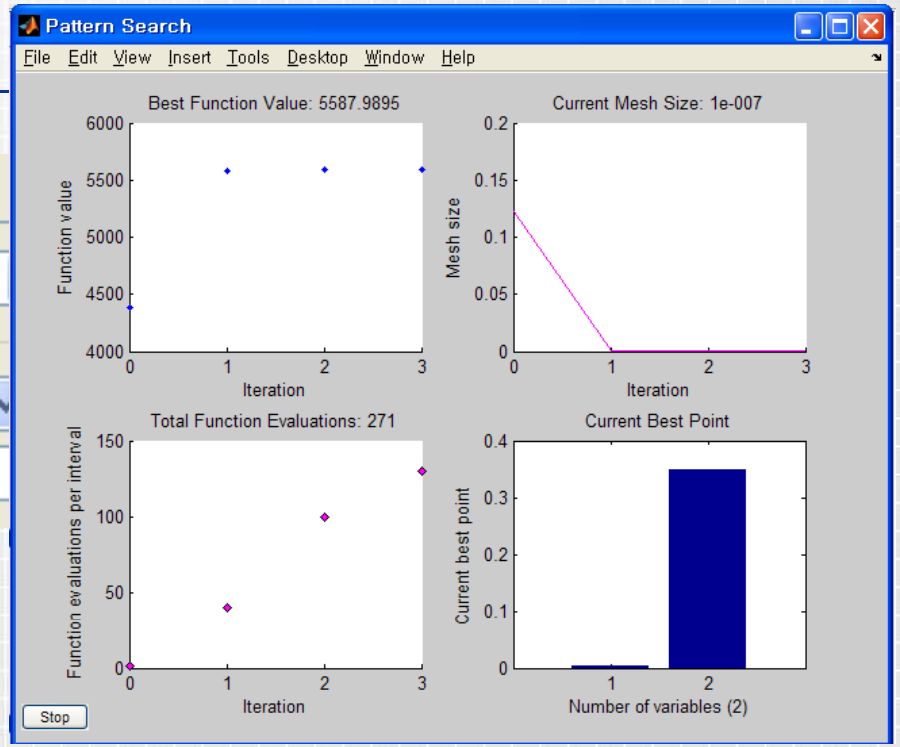
❖ Constraint 를 적용한 Pattern Search

- 예상 설계변수의 크기가 매우 작기 때문에 시작 **mesh**의 크기를 작게 설정하고 값을 찾았다.
- **D**값을 **0.1**로 초기값을 주었을 때 설계 값은 **r=0.003, d=0.1**이 나왔다.

3. Pattern Search







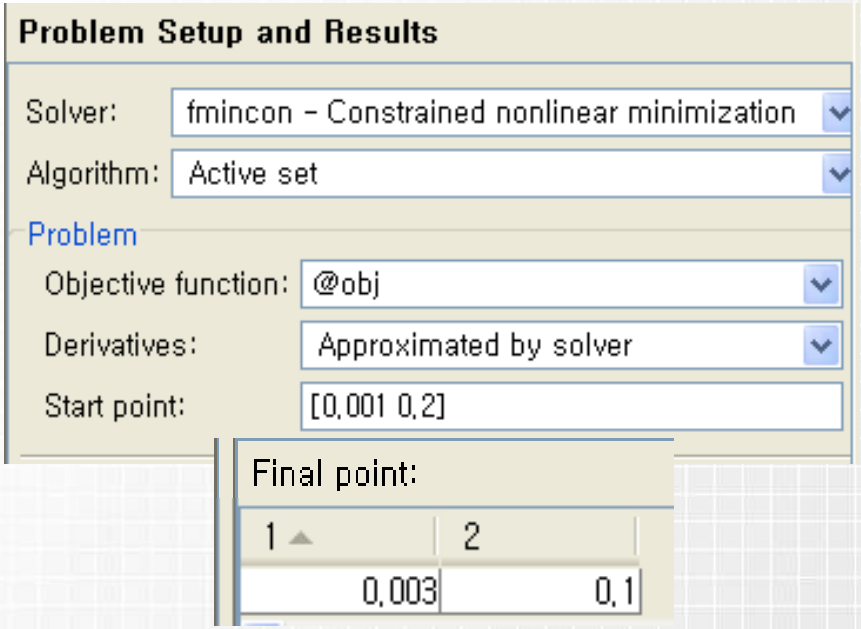
❖ Constraint 를 적용한 Pattern Search

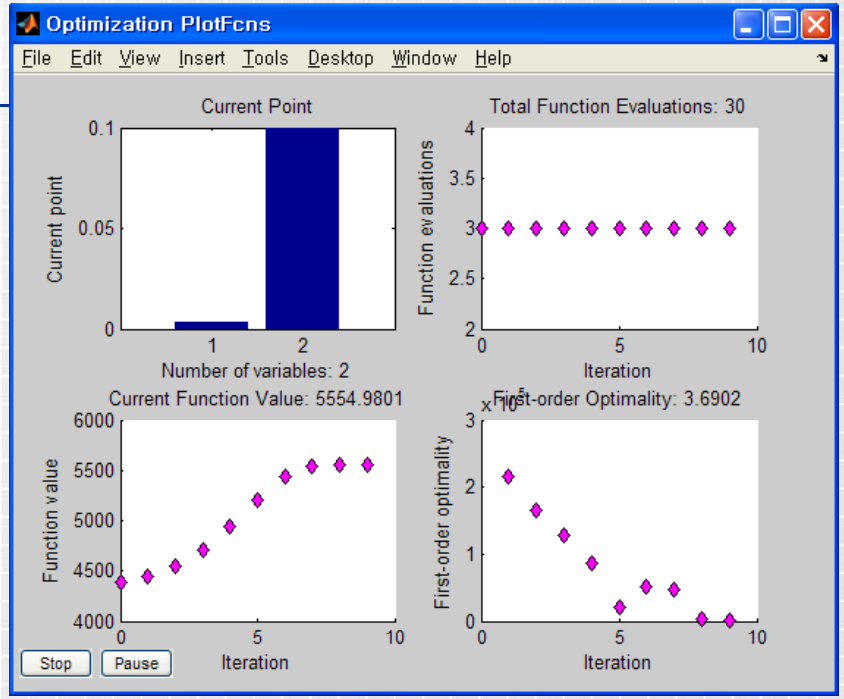
- 초기값이 **0.001, 0.2**로 시작하였을 때
  결과 값이 **0.003, 0.35**가 나왔다.

3. Pattern Search 시나리오

[0.001 0.1]에서 시작하여 [0.003 0.1]에 수렴						
Iter	f-count	f(x)	constraint	MeshSize	Method	
0	1	4365.53	98.65	0.0001		
1	120	5631.36	0	0.001	Increase	penalty
2	124	5631.04	0	1.00E-05	Increase	penalty
3	260	5556.5	0	1.00E-07	Increase	penalty
4	362	5554.99	9.77E-07	1.00E-09	Increase	penalty
5	511	5554.98	1.00E-06	1.00E-11	Increase	penalty
[0.001 0.3]에서 시작하여 [0.003 0.1]에 수렴 (mesh 0.0001 시작)						
Iter	f-count	f(x)	constraint	MeshSize	Method	
0	1	4406.06	56.62	0.0001		
1	187	5554.98	2.88E-07	1.00E-03	Increase	penalty
2	191	5554.68	0.000366	1.00E-05	Increase	penalty
3	250	5554.98	0.00E+00	1.00E-07	Increase	penalty
[0.001 0.3]에서 시작하여 [0.003 0.3]에 수렴 (mesh 1 시작)						
Iter	f-count	f(x)	constraint	MeshSize	Method	
0	1	4406.06	56.62	0.1733		
1	39	5261.35	0.522	0.001	Increase	penalty
2	183	5587.99	1.14E-09	1.00E-05	Increase	penalty
3	331	5587.99	0	1.00E-07	Increase	penalty

4. Fmincon

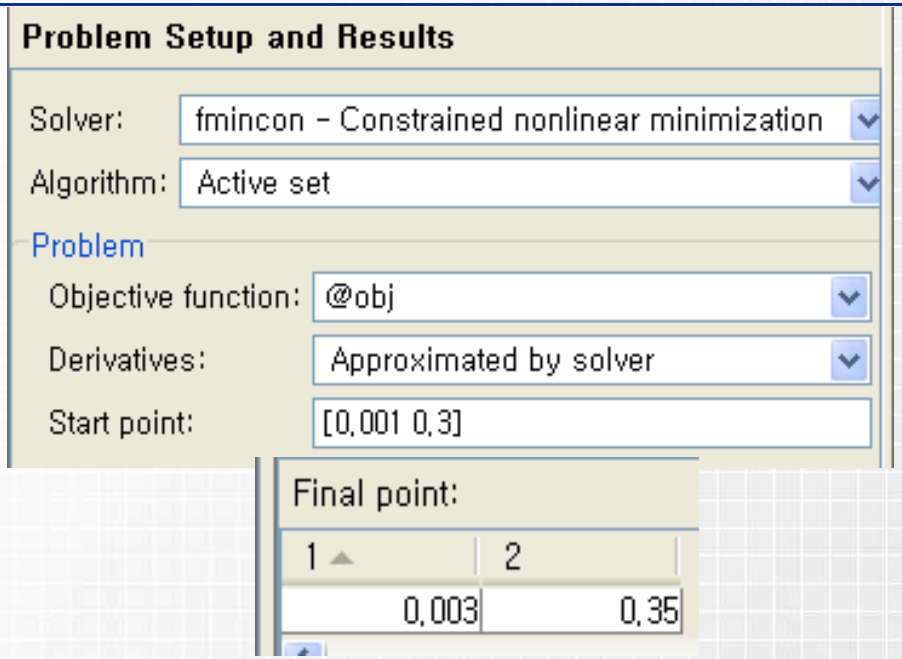


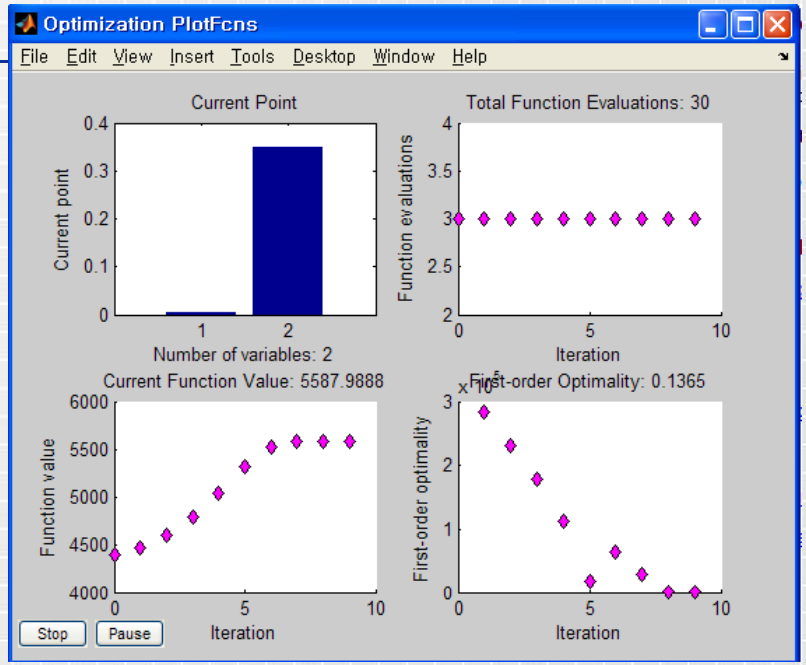


❖ Fmincon을 이용한 최적해 찾기

- 시작점 **[0.001 0.2], nonlinearconstraint**로 인해 **Approximated by solver**이용
- 결과값 **[0.003 0.1]**

4. Fmincon





❖ Fmincon을 이용한 최적해 찾기

- 시작점 **[0.001 0.3]** 앞의 결과와 달리 계산된 결과값 **[0.003 0.1]** 이다.
- **D**의 시작값을 **0.01**단위로 바꾸면서 해본결과인데 **Pettern Search**보다 **0.5**더 큰 **D**값에서 **0.35**를 찾아 갔다.

4. Fmincon

[0.001 0.1]에서 시작하여 [0.003 0.1]에 수렴							
Max	Line	search	Directiona	First-order			
Iter	F-count	f(x)	constraint	steplength	derivative	optimality	Procedure
0	3	4365.53	98.65	Infeasible	start	point	
1	6	4430.22	40.2	1	2.32E+05	2.13E+05	
2	9	4529.72	16.26	1	2.90E+05	1.76E+05	
3	12	4679.31	6.46	1	3.61E+05	1.40E+05	
4	15	4892.28	2.454	1	4.47E+05	9.85E+04	
5	18	5158.82	0.8334	1	5.46E+05	3.76E+04	
6	21	5407.96	0.212	1	6.50E+05	3.97E+04	
7	24	5534.35	0.02542	1	7.33E+05	5.52E+04	
8	27	5554.57	0.00049	1	7.72E+05	6.73E+03	
9	30	5554.98	1.86E-07	1	7.78E+05	19.5	
[0.001 0.3]에서 시작하여 [0.003 0.3]에 수렴							
Max	Line	search	Directiona	First-order			
Iter	F-count	f(x)	constraint	steplength	derivative	optimality	Procedure
0	3	4397.95	64.33	Infeasible	start	point	
1	6	4481.29	25.4	1	1.45E+03	2.83E+05	
2	9	4607.34	10.2	1	3.81E+05	2.31E+05	
3	12	4792.76	3.982	1	4.73E+05	1.79E+05	
4	15	5043.34	1.447	1	5.83E+05	1.13E+05	
5	18	5321.49	0.439	1	7.05E+05	1.79E+04	
6	21	5522.3	0.08316	1	8.19E+05	6.37E+04	
7	24	5583.96	0.00475	1	8.93E+05	3.06E+04	
8	27	5587.97	1.77E-05	1	9.14E+05	719	
9	30	5587.99	2.40E-11	1	9.16E+05	0.136	

5.analysis

- 초기 값에 따라서 최적화 툴이 찾은 설계 변수가 달라졌다.
- **Pattern Search**의 경우 **mesh**의 크기에 따라서 찾는 값이 달라졌다.

	Excel		GA		PS		Fmincon	
	R	D	R	D	R	D	R	D
설계변수	0.002924	0.35	0.3354	0.1	0.003(0.003)	0.1(0.35)	0.003(0.003)	0.1(0.35)
목적함수	5529.469245		5554.992		5554.98(5587.99)		5554.98(5587.99)	
iteration			3		6(4)		10(10)	

Conclusion

- ✓ 설계 값을 찾는 데에 초기값 선정의 중요성과 알고리즘의 특성을 알고 잘 선택하는 것이 중요하다.
- ✓ 초기값에 따라서 더 나은 목표 값을 찾을 수 있으므로 그냥 찾기보단 초기값을 변화 시키면서 찾아보는 과정이 필요하다.
- ✓ 중간발표 때 적용하지 못한 우산살의 내구력을 적용하였다.(설계의 확장)

Thank You !

From Anti Stress