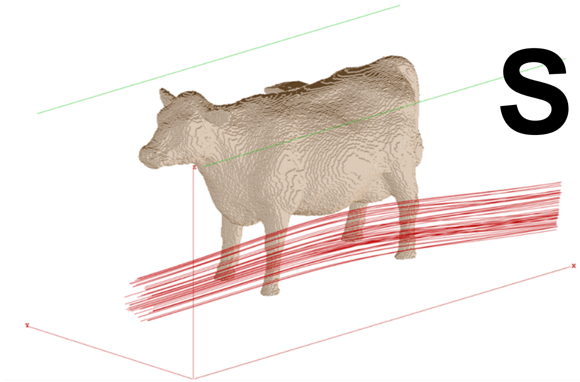
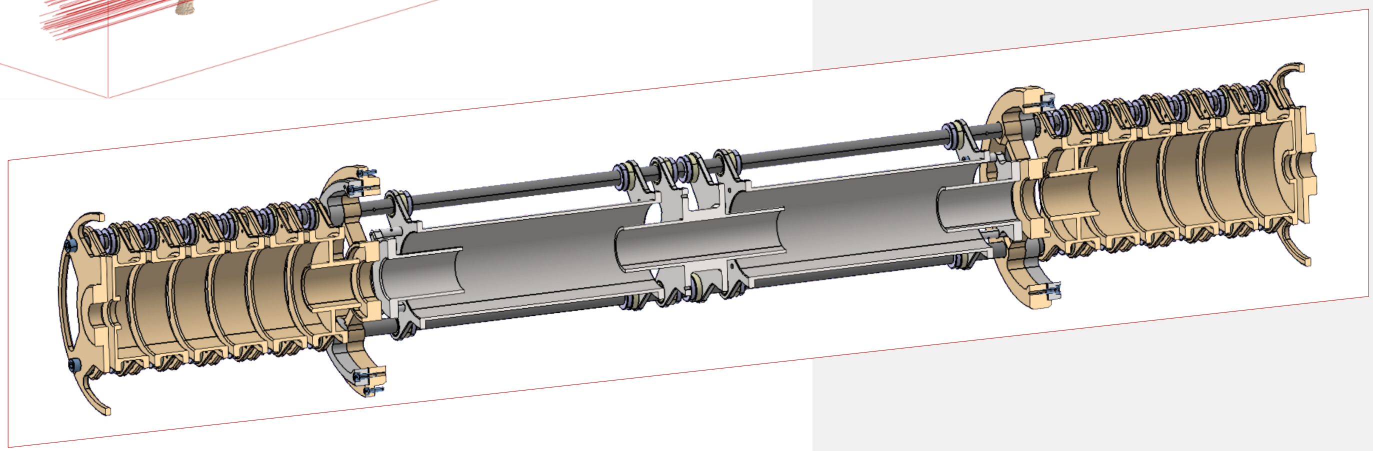
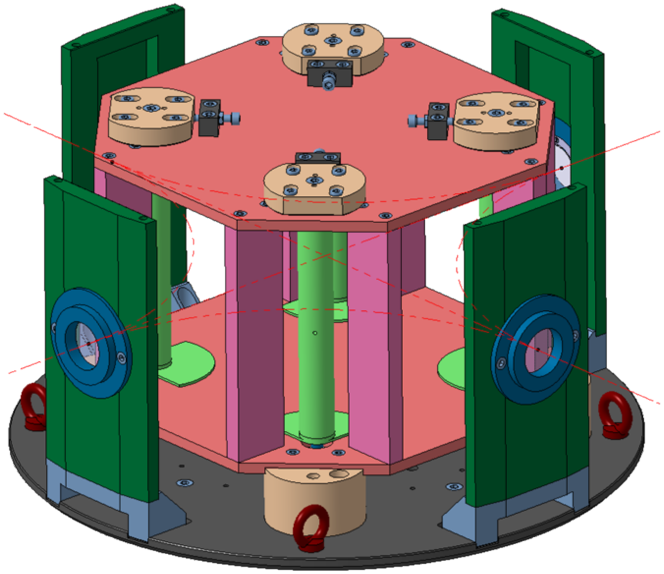
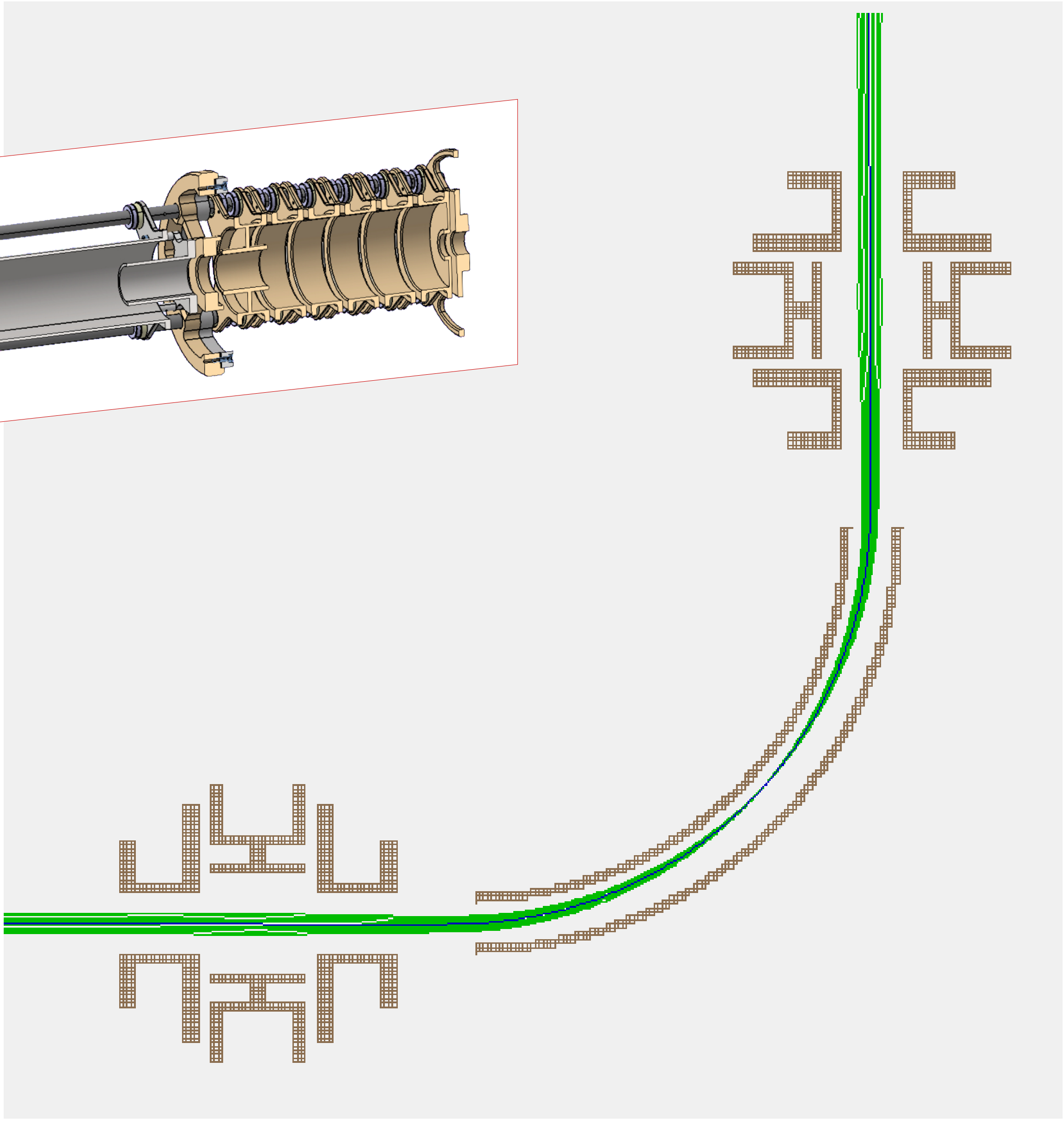
SIMION's Simplex optimizer









Introduction

- How to optimize an ion optics device ?

Introduction

- How to optimize an ion optics device ?

- Built in Simplex optimizer in SIMION.

Introduction

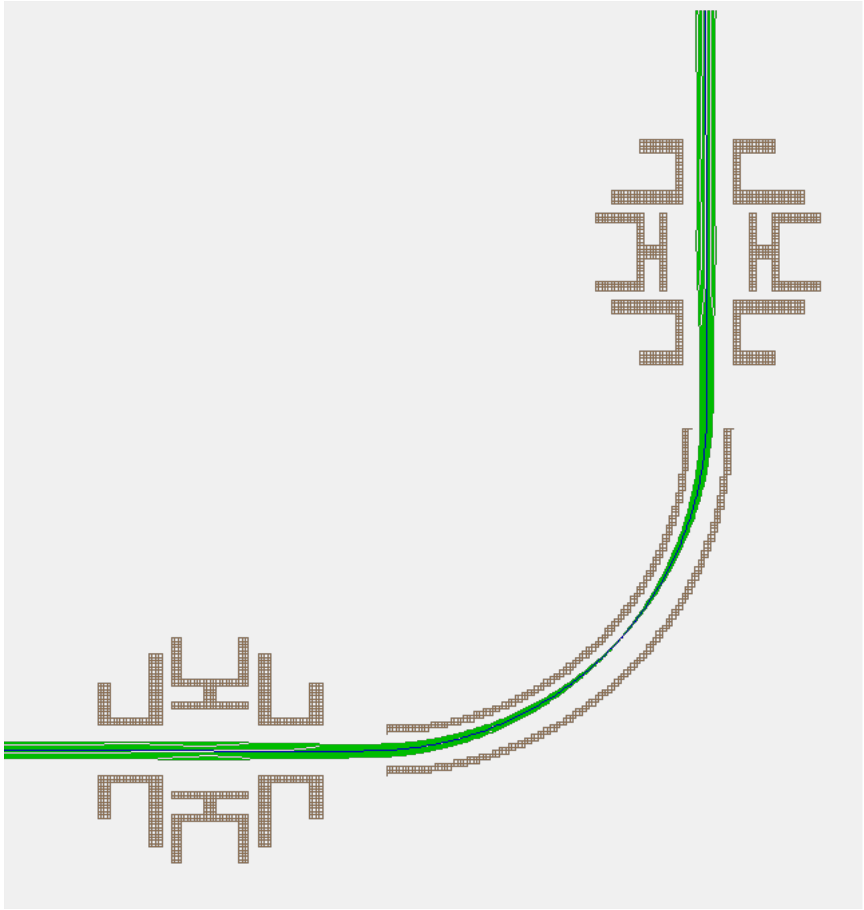
- How to optimize an ion optics device ?
- Built in Simplex optimizer in SIMION.
- Easy to use, can optimize anything

Introduction

- How to optimize an ion optics device ?
- Built in Simplex optimizer in SIMION.
- Easy to use, can optimize anything
- Some limitations, no ion optics theory

Introduction

- How to optimize an ion optics device ?
- Built in Simplex optimizer in SIMION.
- Easy to use, can optimize anything
- Some limitations, no ion optics theory
- Example of a 90° blade deflector



Outline

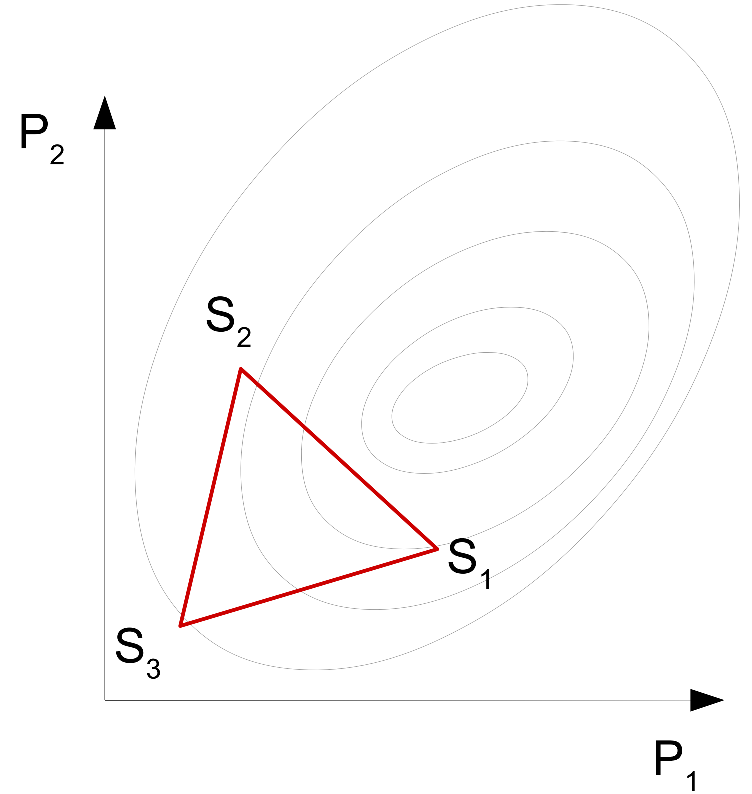
I. The Nelder-Mead Algorithm

II. Potential optimization

III. Geometric optimization

The Nelder-Mead algorithm

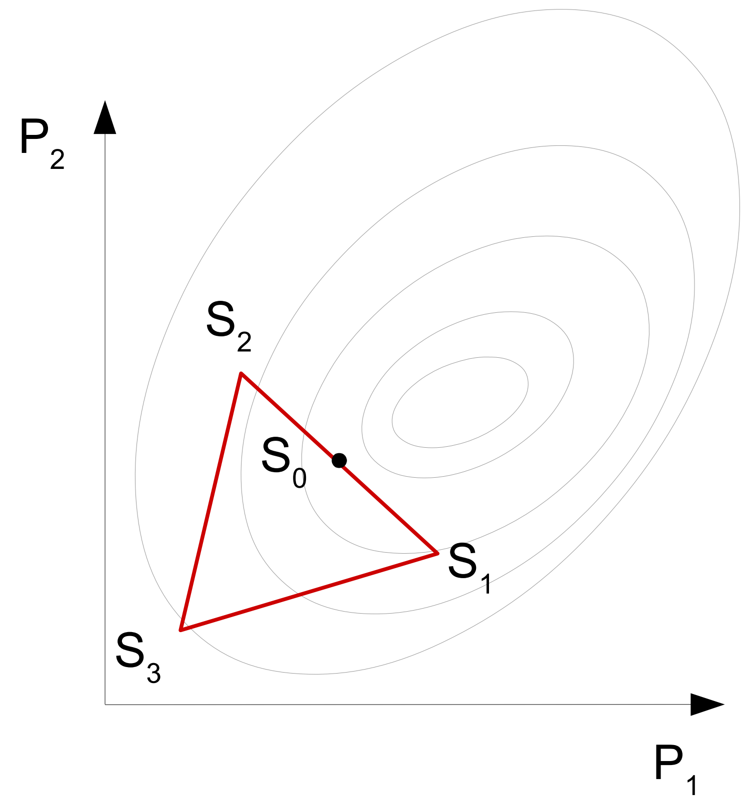
1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1)$ < … < (S_{n+1})



The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < \dots < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

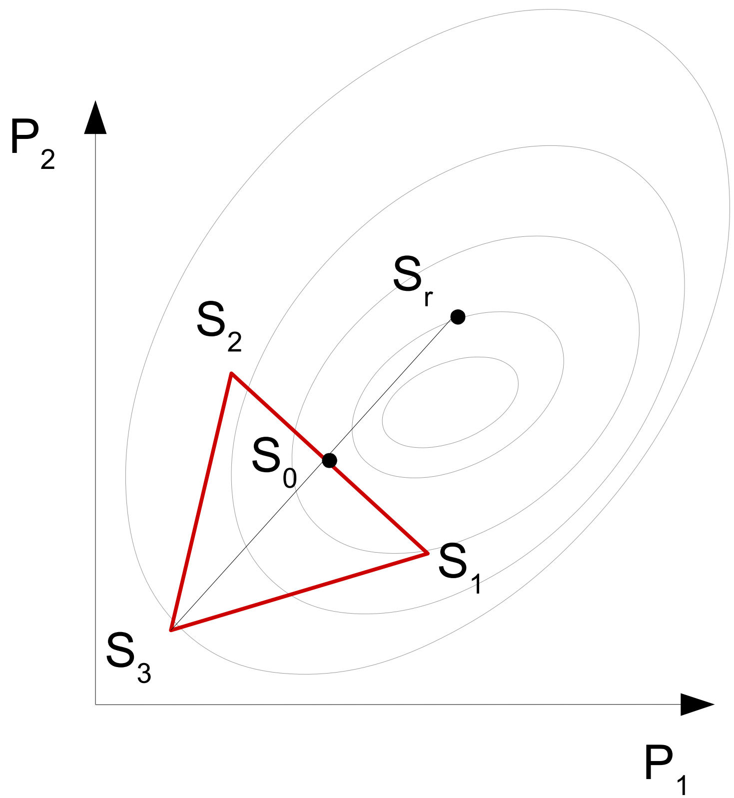


The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < … < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$



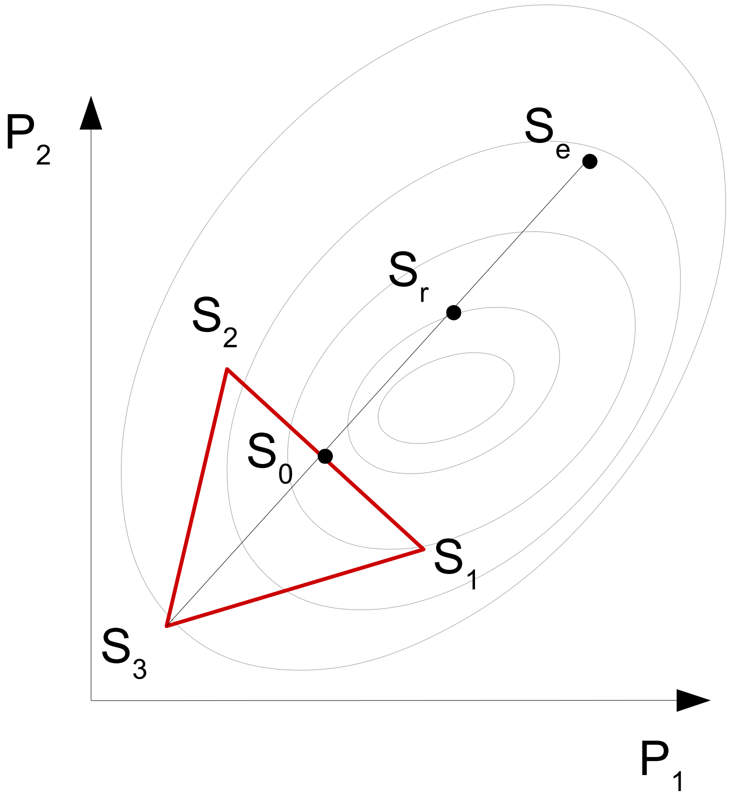
The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < \ldots < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$



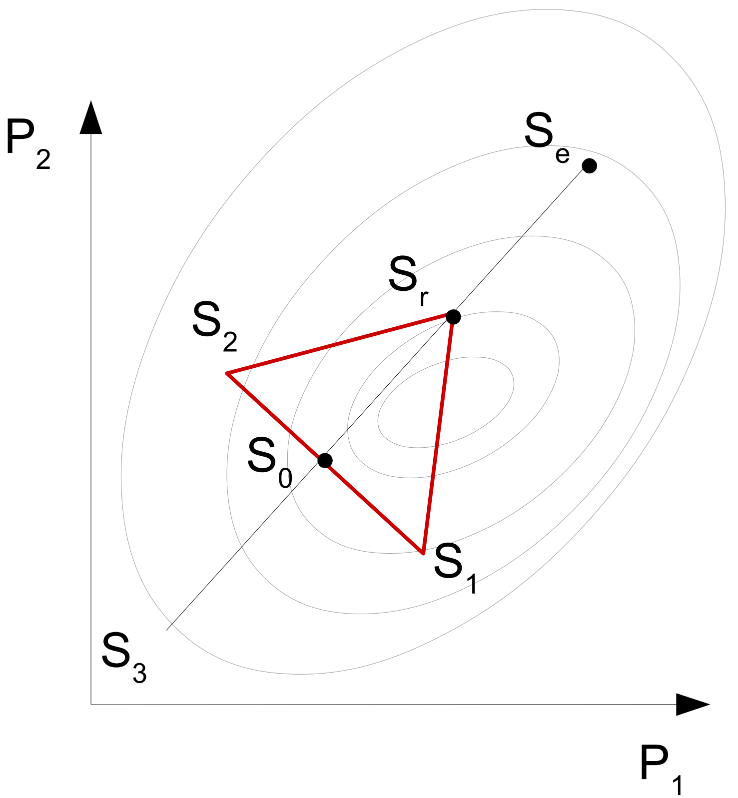
The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1)$ < … < (S_{n+1})

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e



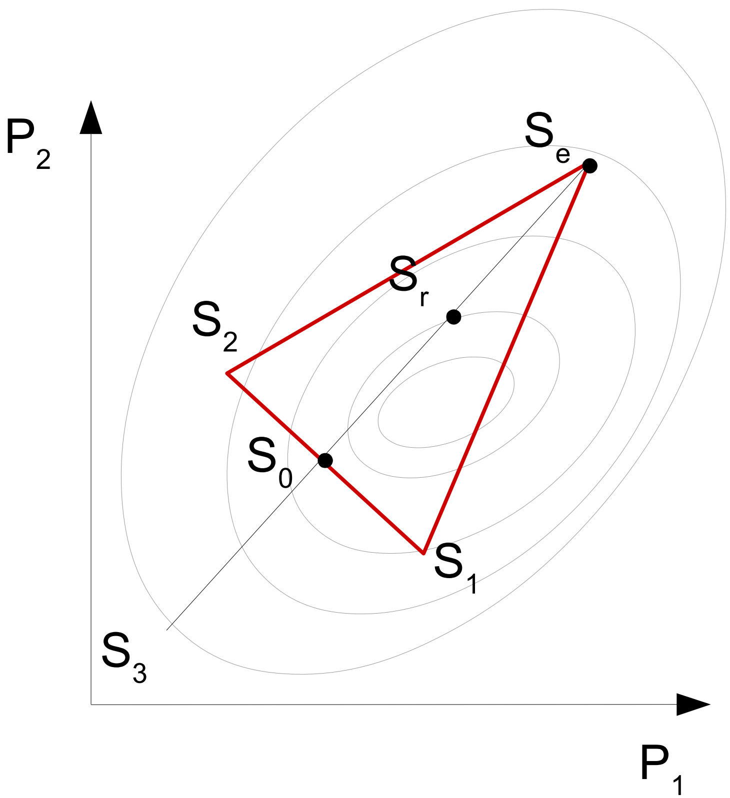
The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < \ldots < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e



The Nelder-Mead algorithm

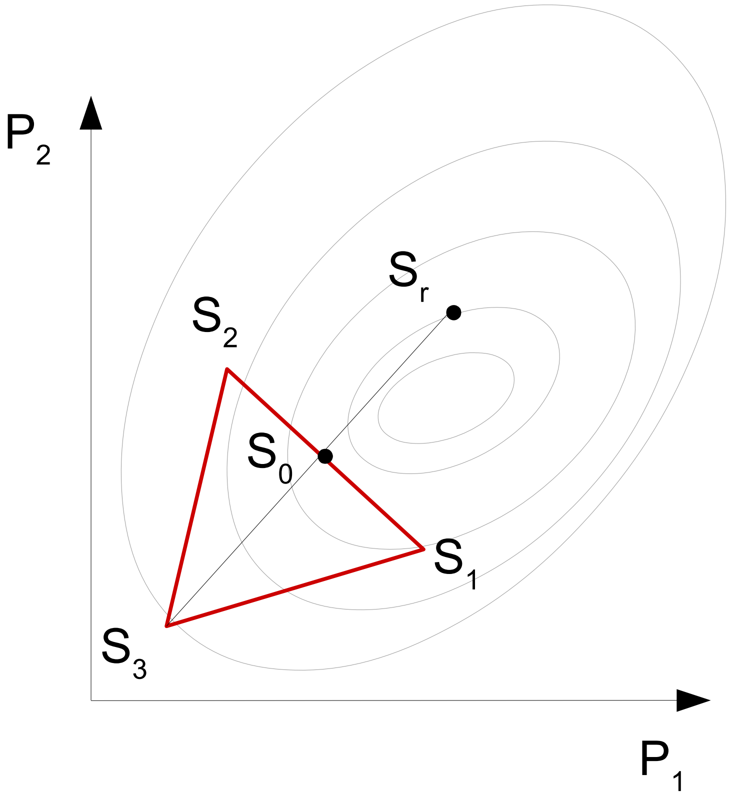
1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1)$ < … < (S_{n+1})

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r



The Nelder-Mead algorithm

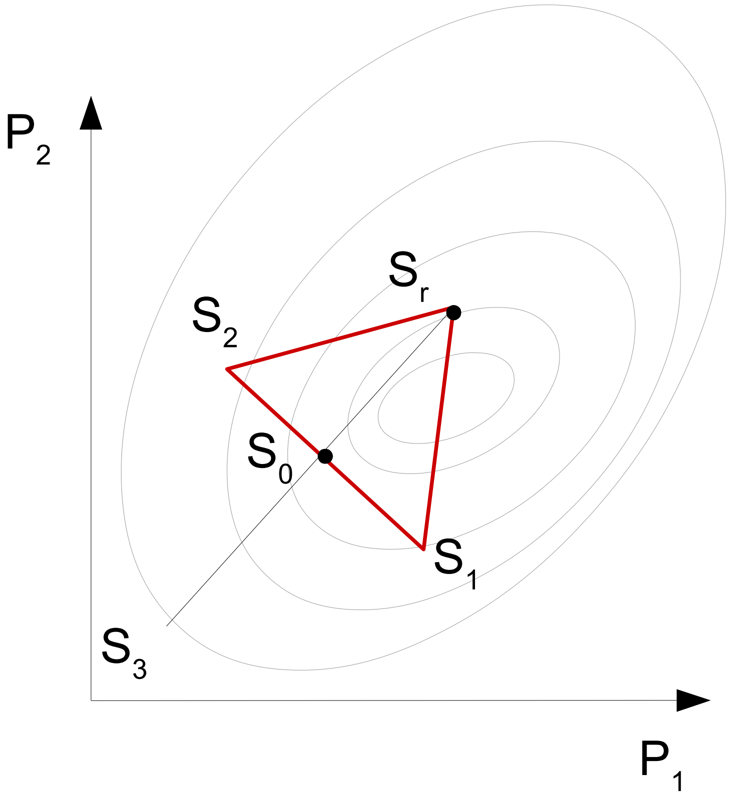
1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < \dots < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r



The Nelder-Mead algorithm

1. Sort the parameter sets $S_1, \dots, S_{n+1}$ so $f(S_1) < \dots < (S_{n+1})$

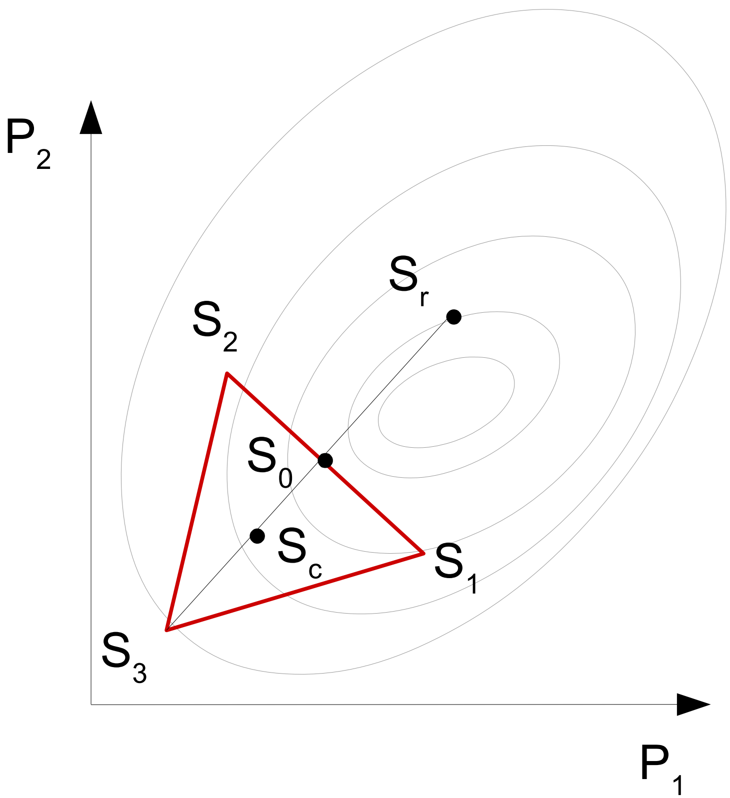
2. Calculate S_0 the barycenter of $S_1, \dots, S_n$

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r

6. If $f(S_r) > f(S_n)$ calculate $S_c = S_0 + (S_0 – S_{n+1})/2$



The Nelder-Mead algorithm

1. Sort the parameter sets $S_1, \dots, S_{n+1}$ so $f(S_1) < \dots < (S_{n+1})$

2. Calculate S_0 the barycenter of $S_1, \dots, S_n$

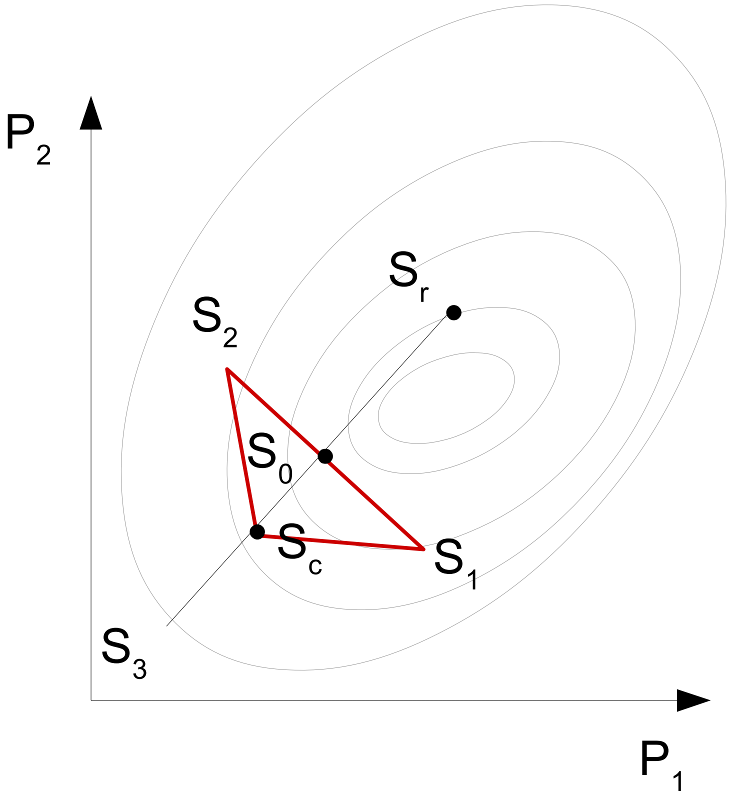
3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r

6. If $f(S_r) > f(S_n)$ calculate $S_c = S_0 + (S_0 – S_{n+1})/2$

7. If $S_c < S_{n+1}$ replace S_{n+1} with S_c



The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < … < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

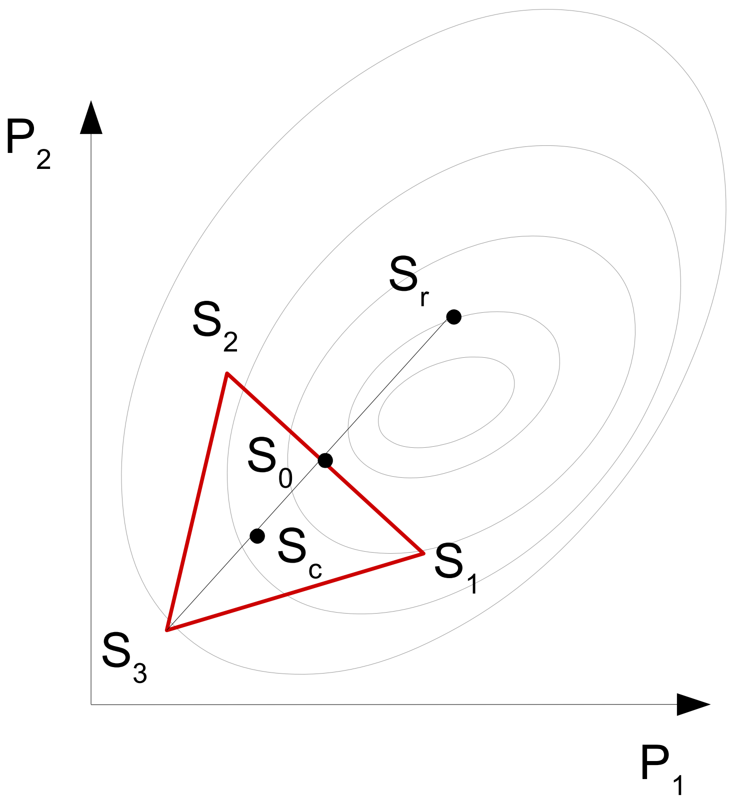
4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r

6. If $f(S_r) > f(S_n)$ calculate $S_c = S_0 + (S_0 – S_{n+1})/2$

7. If $S_c < S_{n+1}$ replace S_{n+1} with S_c

8. Else replace all S_i , i>1 with $S_1 + (S_i – S_1)/2$



The Nelder-Mead algorithm

1. Sort the parameter sets S_1 , … , S_{n+1} so $f(S_1) < … < (S_{n+1})$

2. Calculate S_0 the barycenter of S_1 , … , S_n

3. Calculate $S_r = S_0 + (S_0 - S_{n+1})$

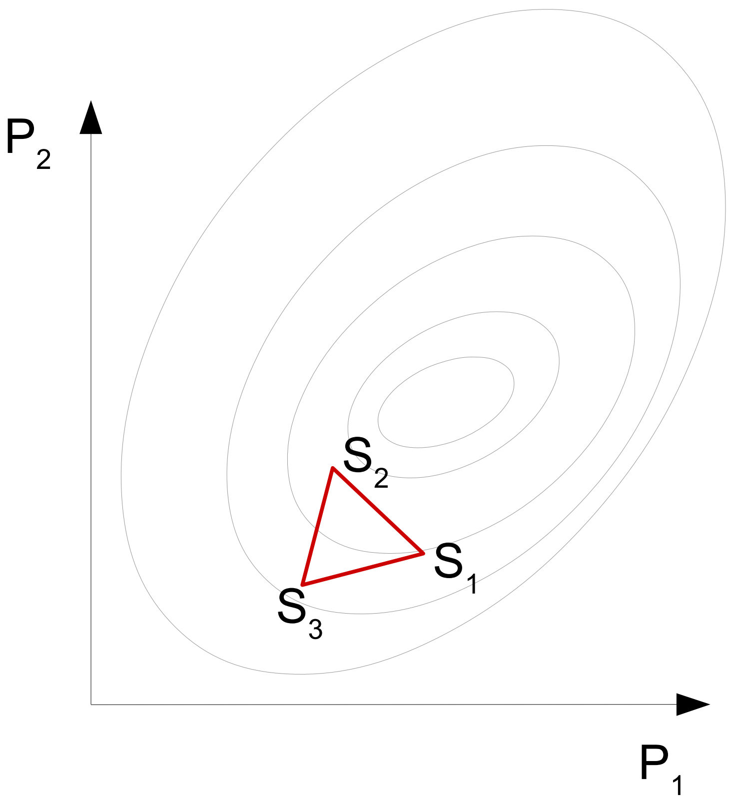
4. If $f(S_r) < f(S_1)$ calculate $S_e = S_0 + 2(S_0 - S_{n+1})$ and replace S_{n+1} with the best of S_r and S_e

5. If $f(S_1) < f(S_r) < f(S_n)$ replace S_{n+1} with S_r

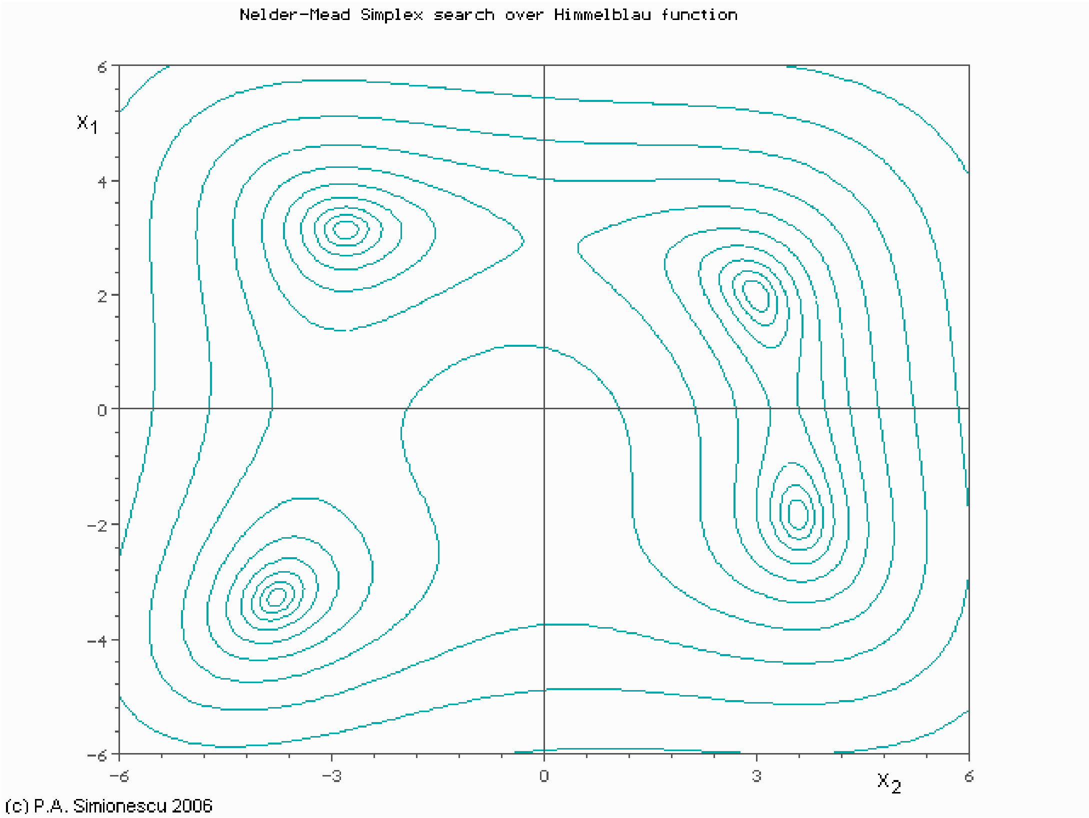
6. If $f(S_r) > f(S_n)$ calculate $S_c = S_0 + (S_0 – S_{n+1})/2$

7. If $S_c < S_{n+1}$ replace S_{n+1} with S_c

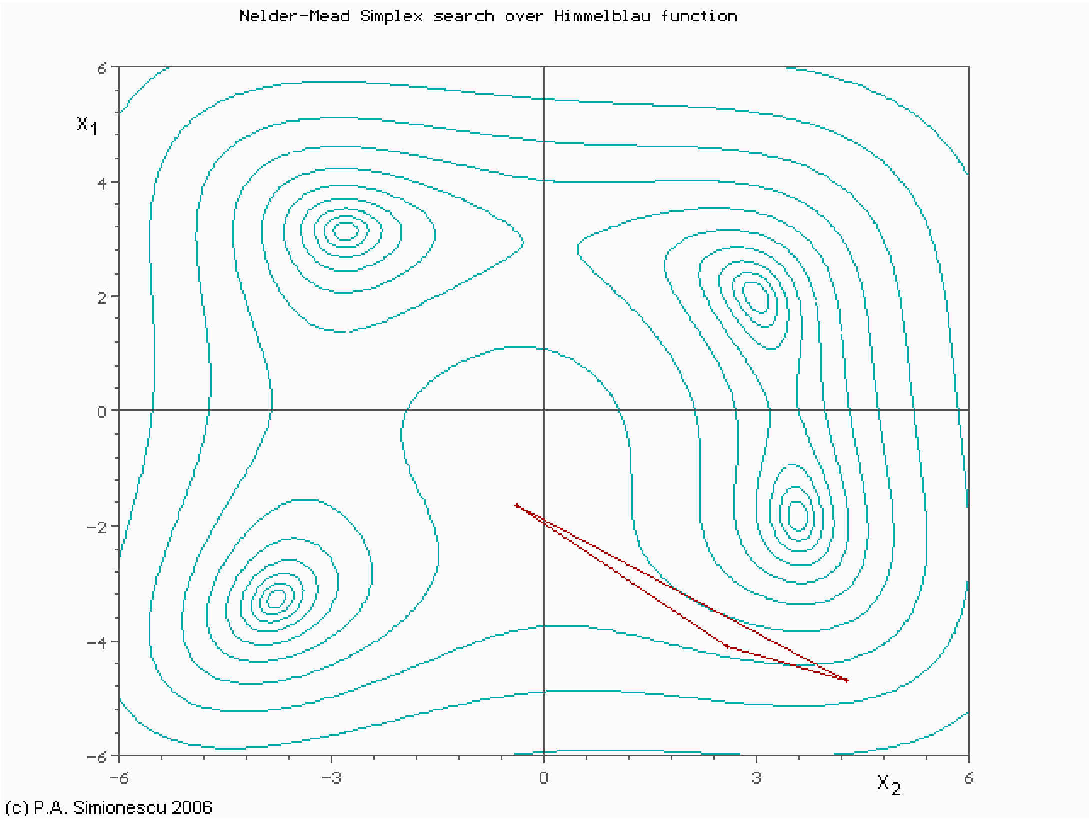
8. Else replace all S_i , i>1 with $S_1 + (S_i – S_1)/2$



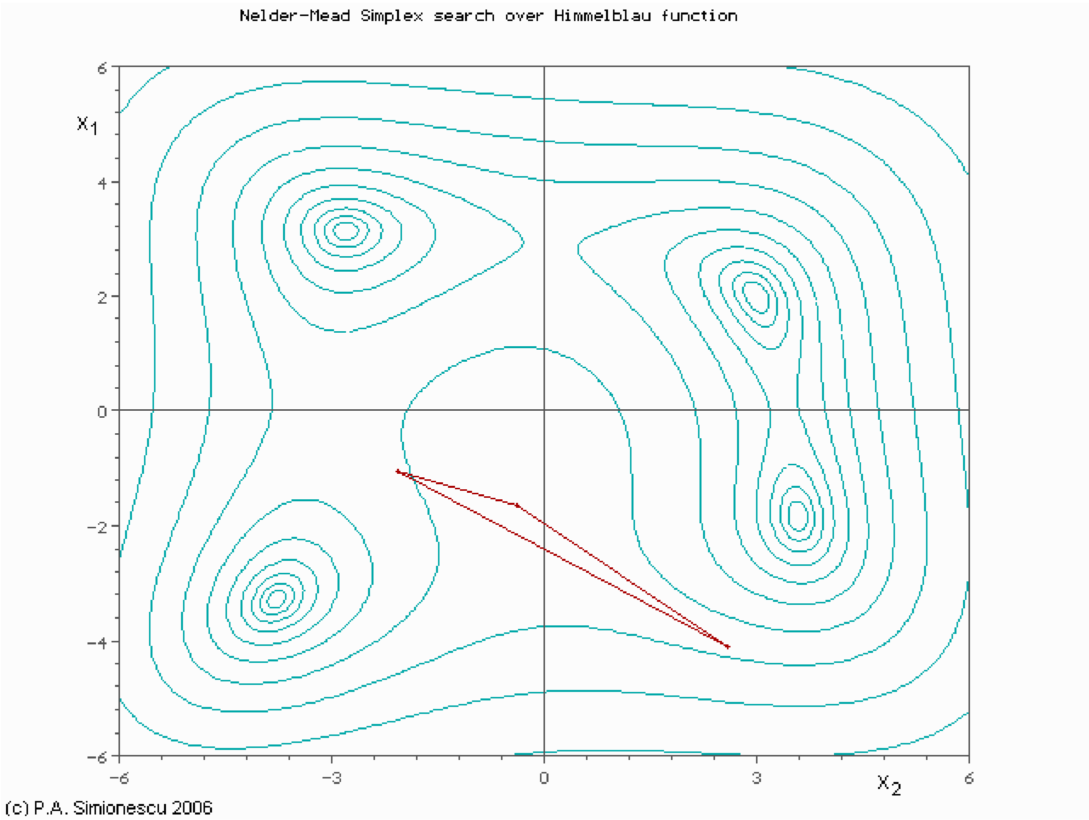
The Nelder-Mead algorithm



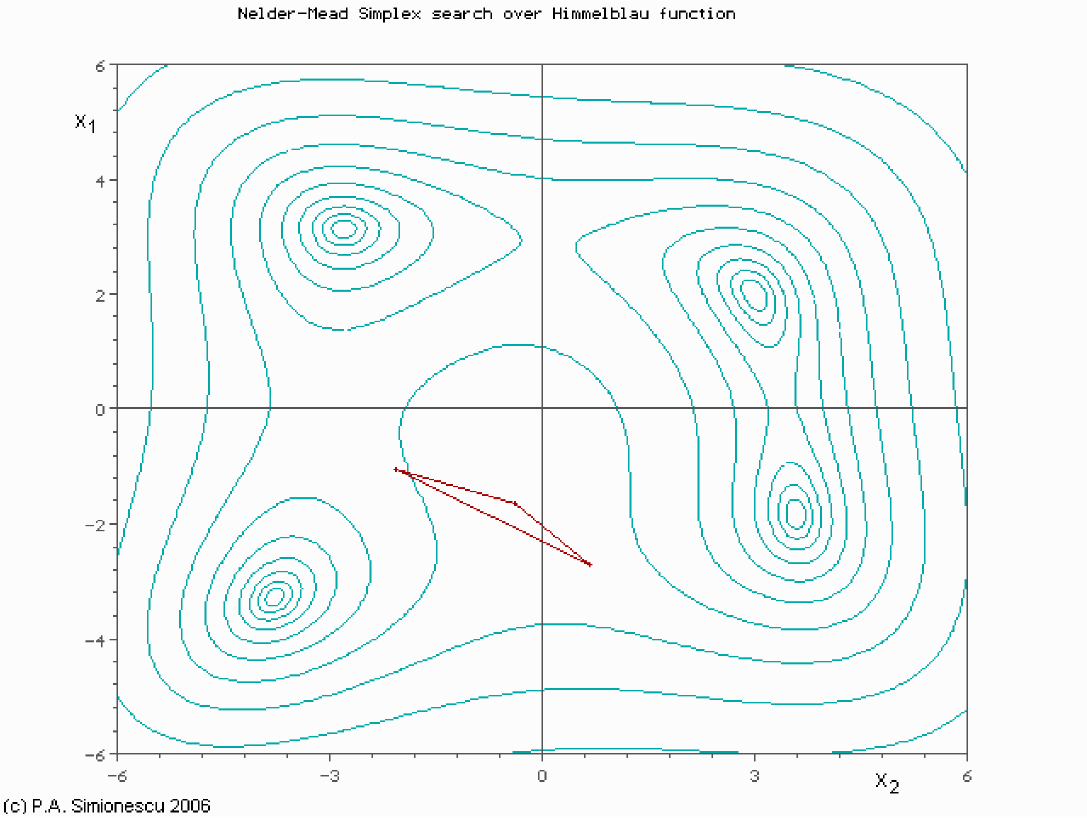
The Nelder-Mead algorithm



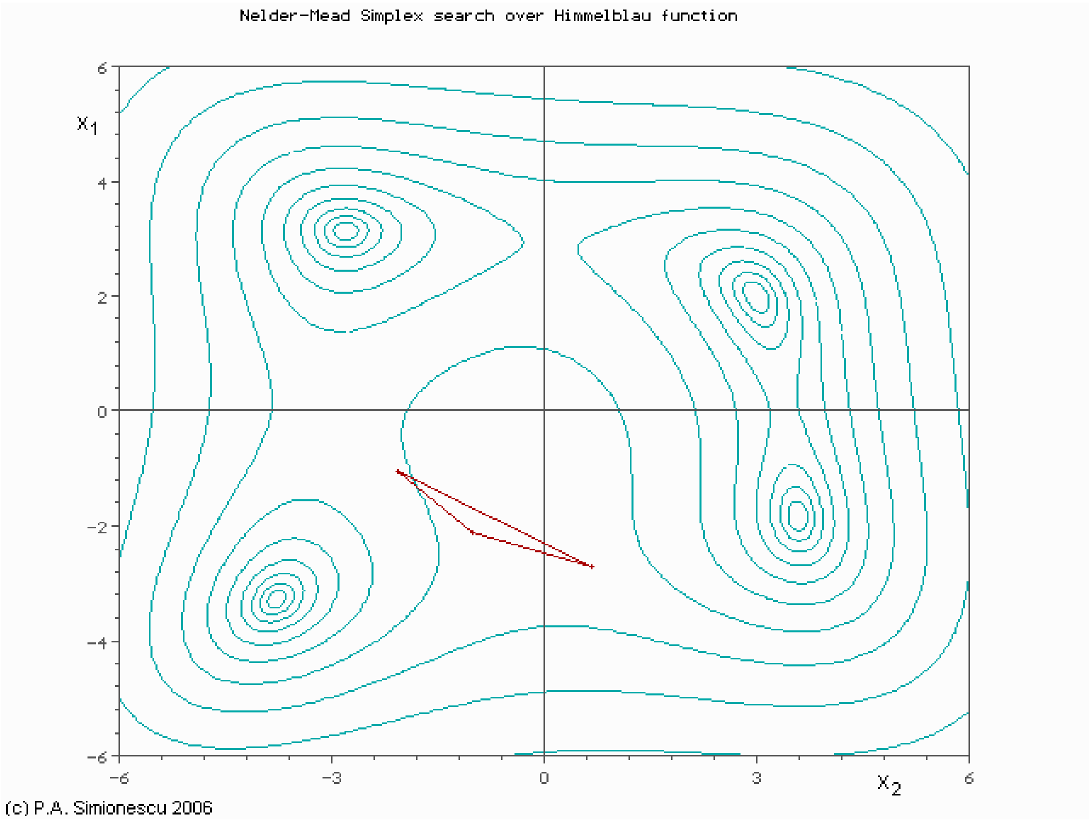
The Nelder-Mead algorithm



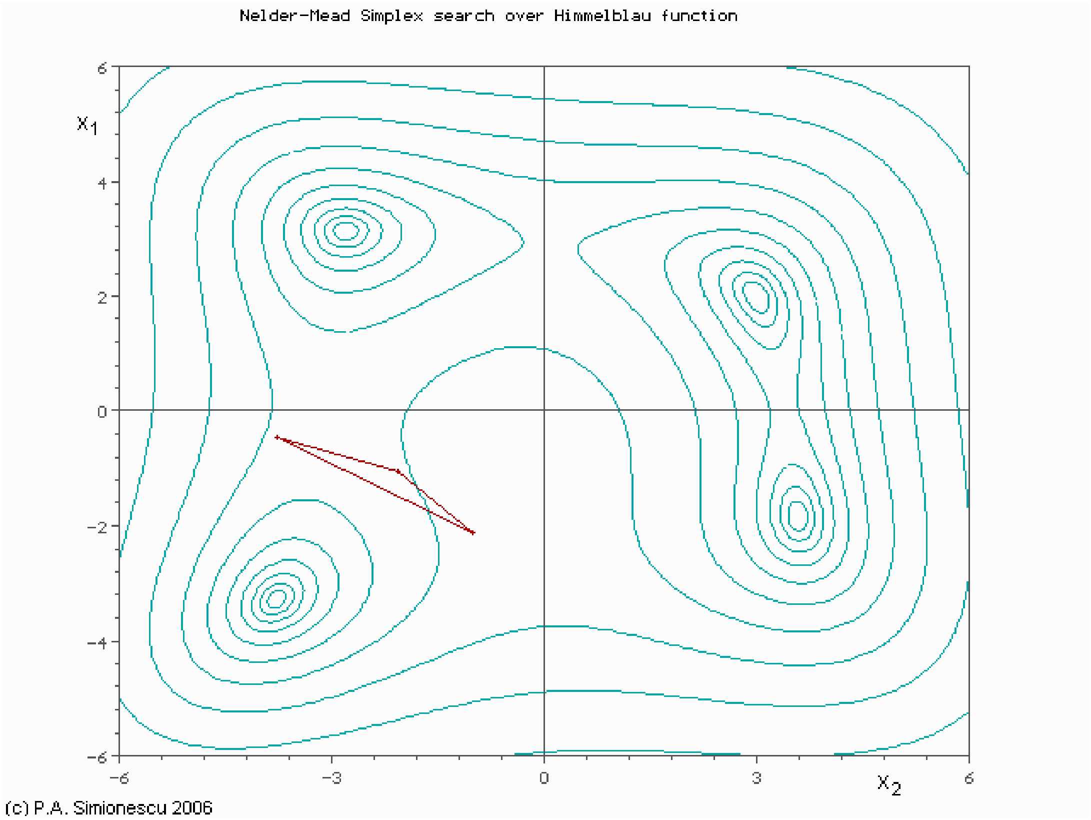
The Nelder-Mead algorithm



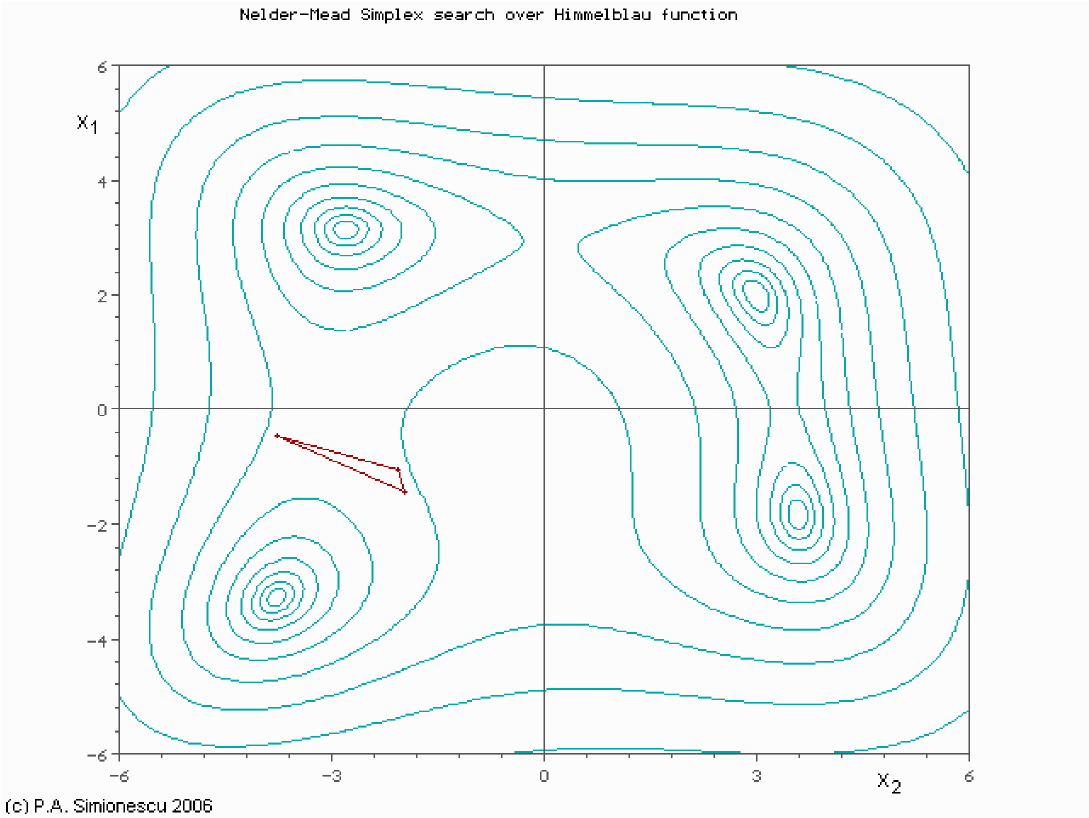
The Nelder-Mead algorithm



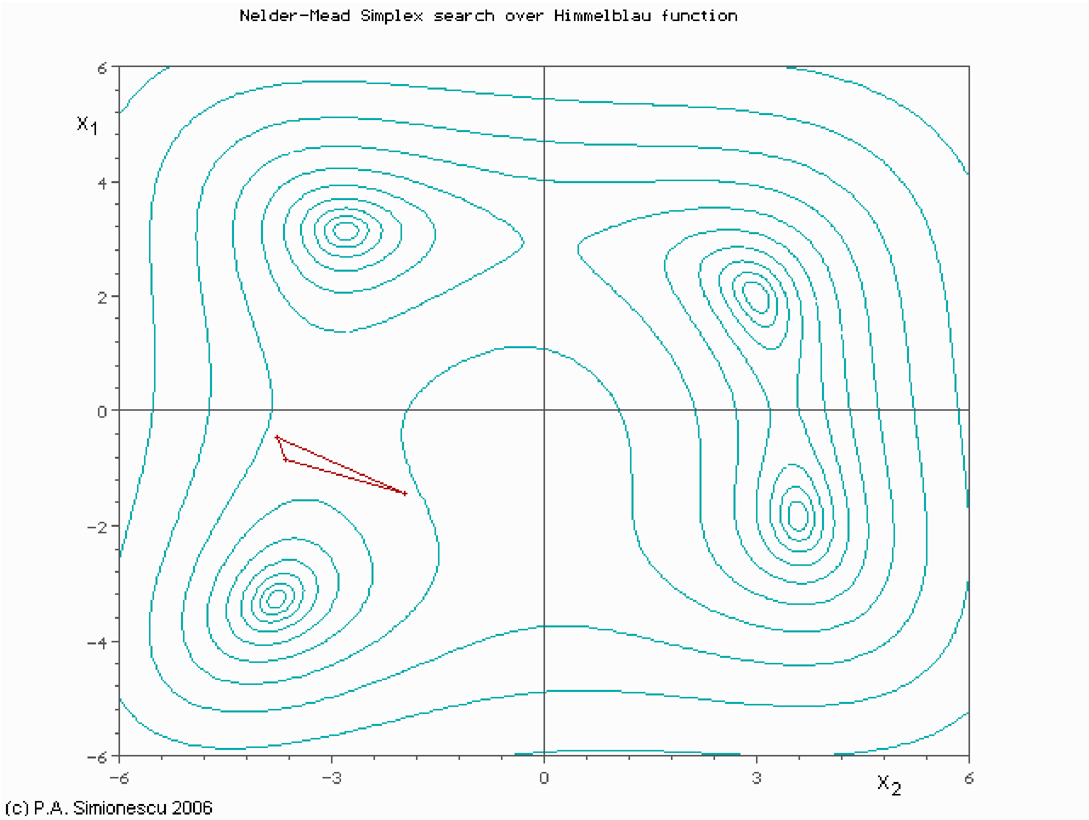
The Nelder-Mead algorithm



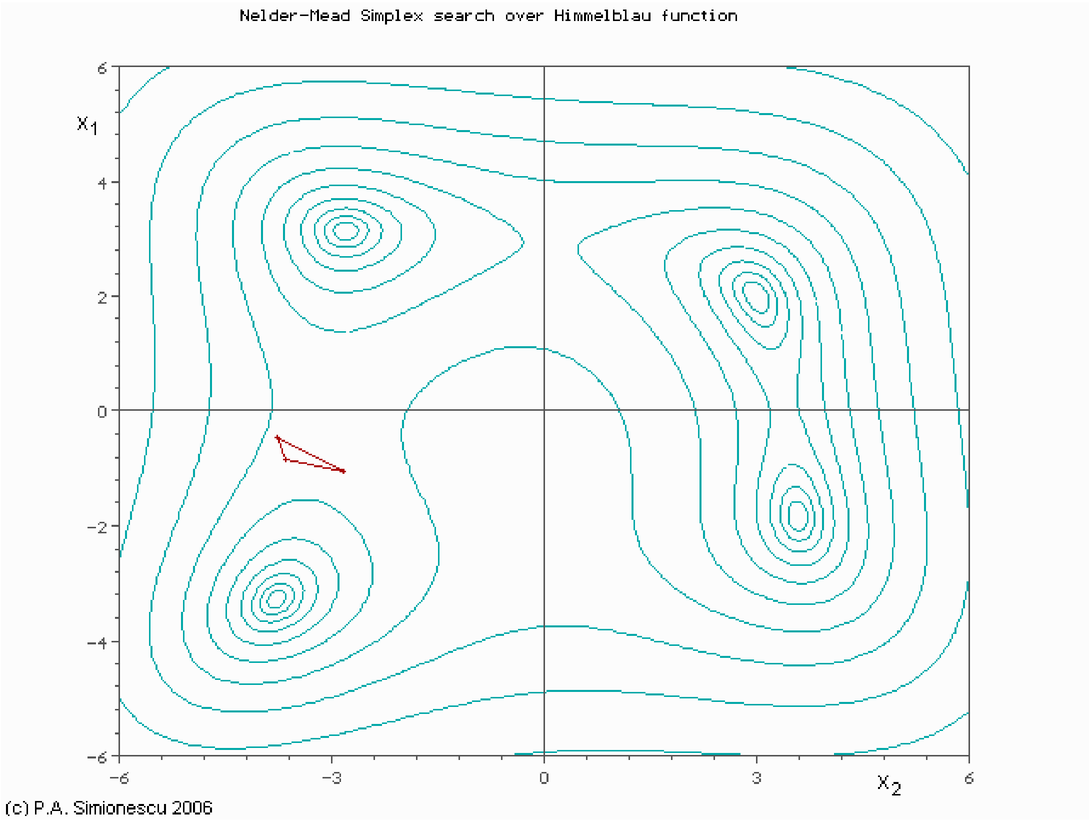
The Nelder-Mead algorithm



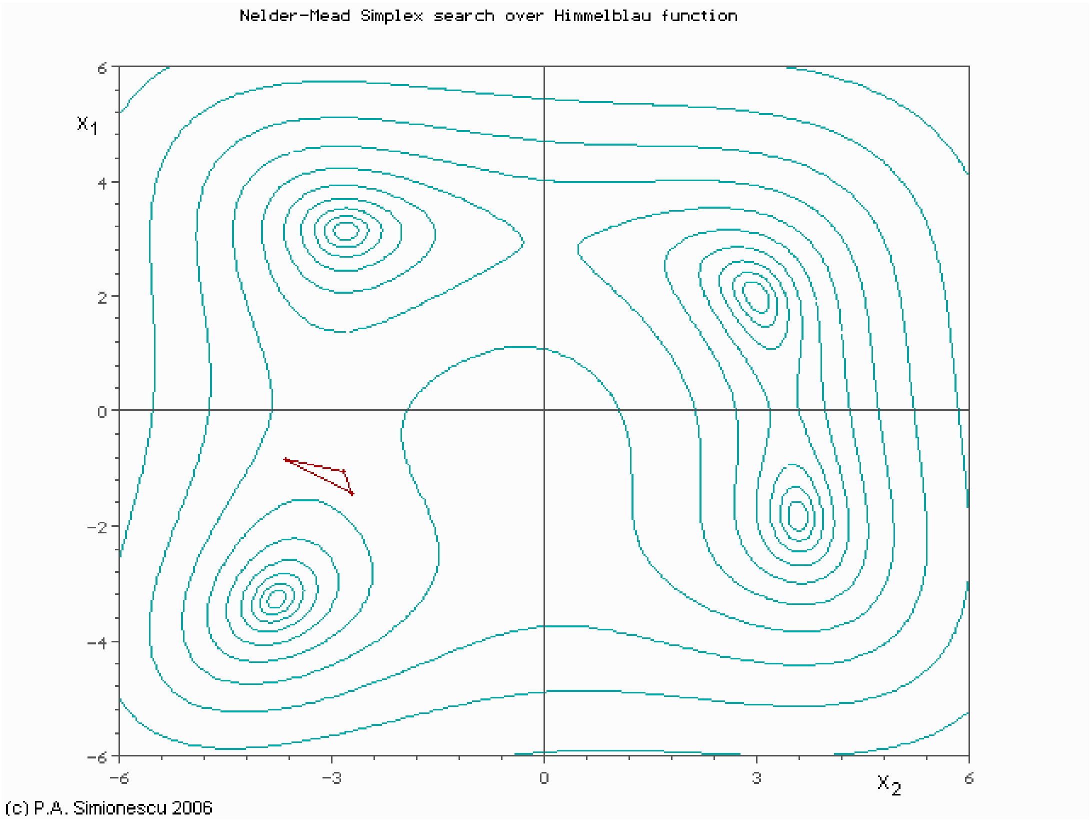
The Nelder-Mead algorithm



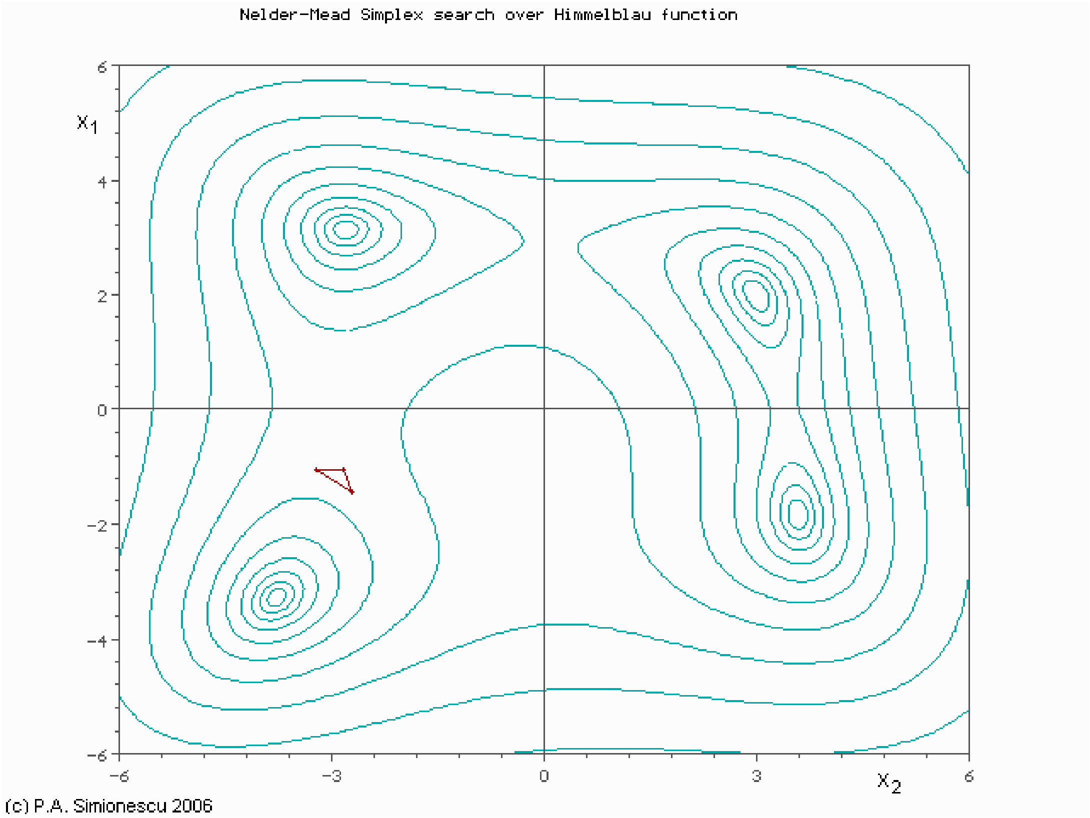
The Nelder-Mead algorithm



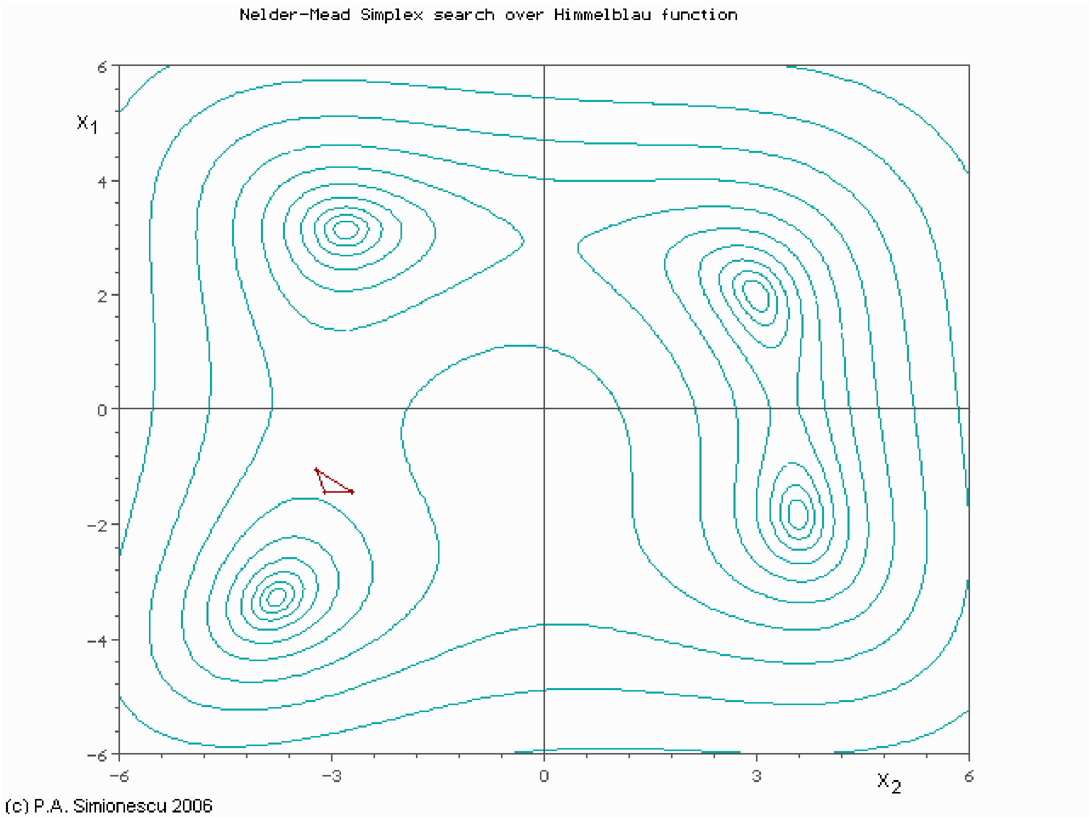
The Nelder-Mead algorithm



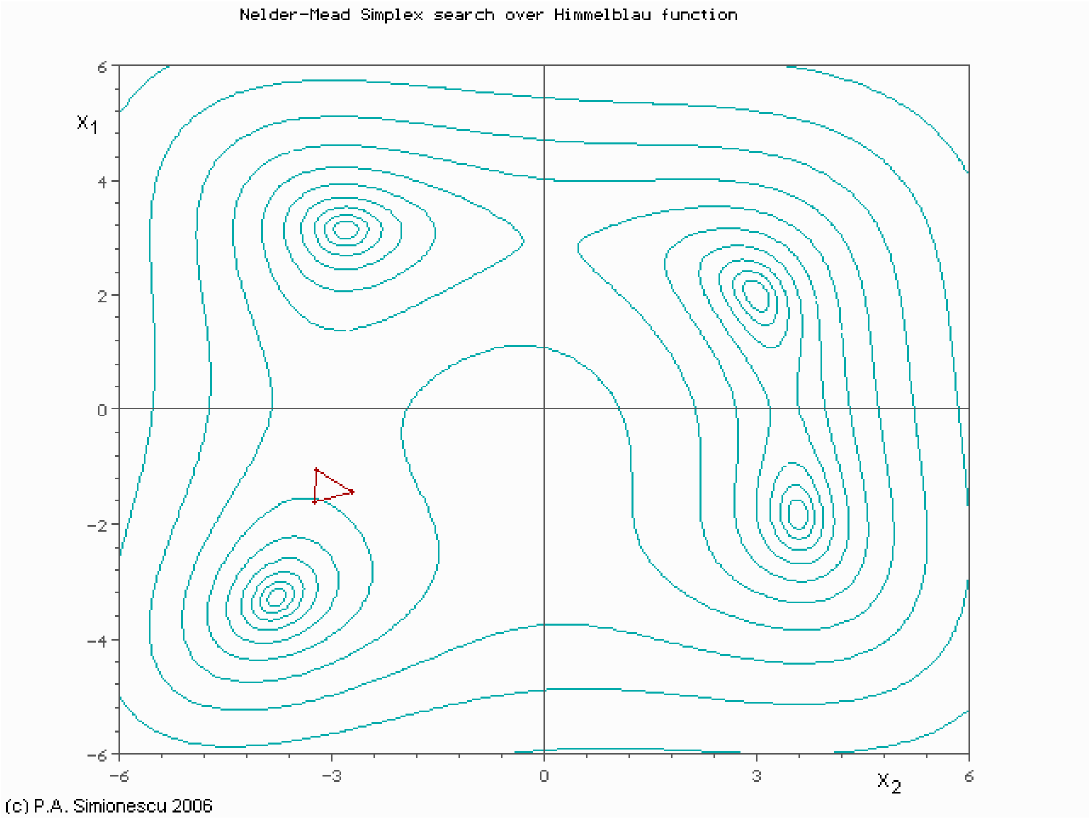
The Nelder-Mead algorithm



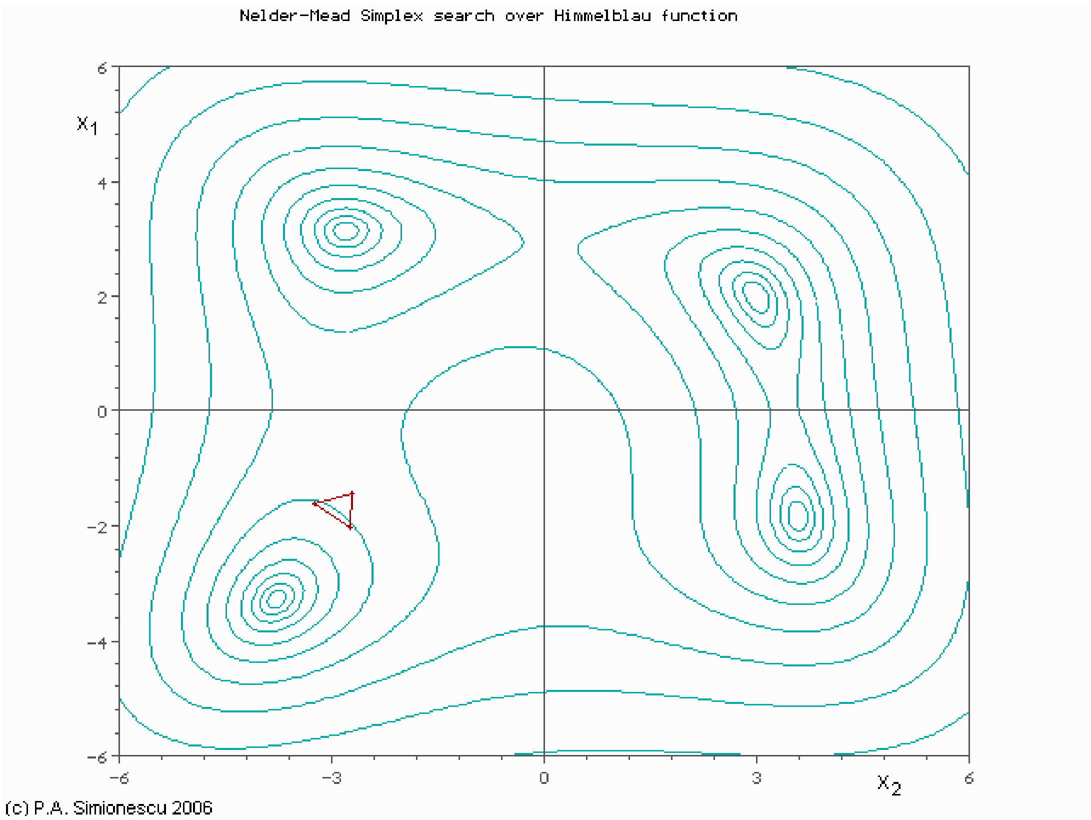
The Nelder-Mead algorithm



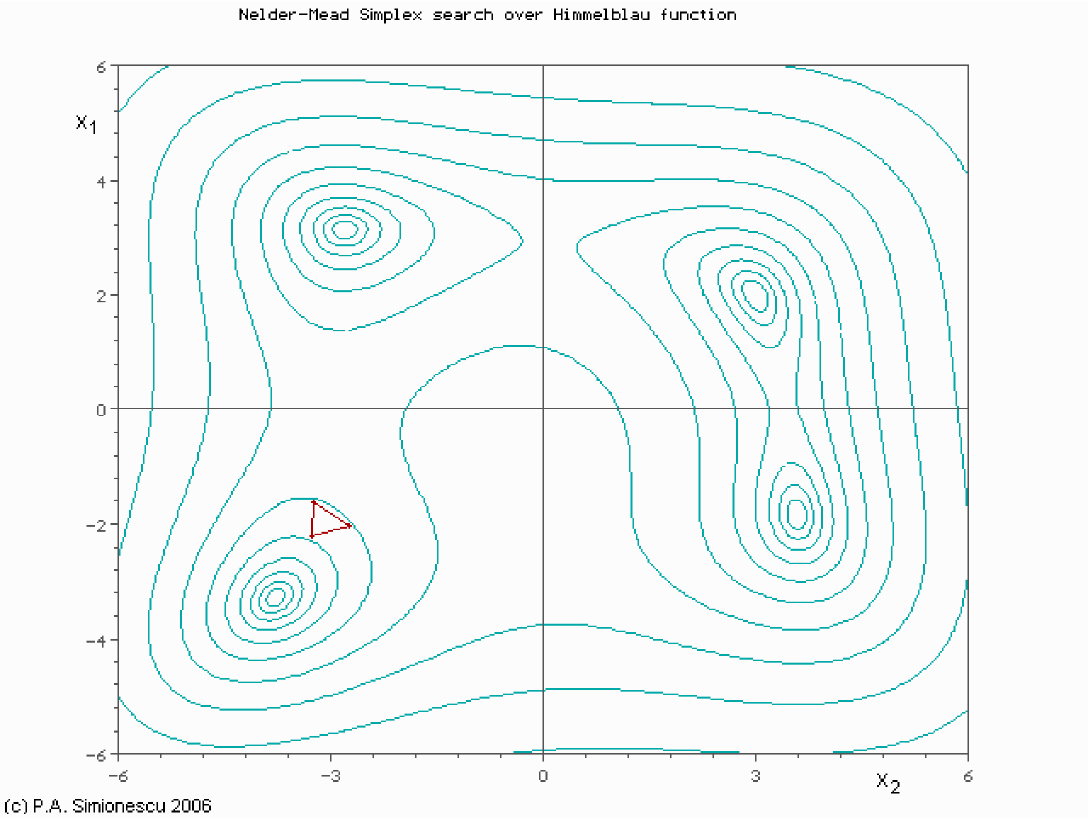
The Nelder-Mead algorithm



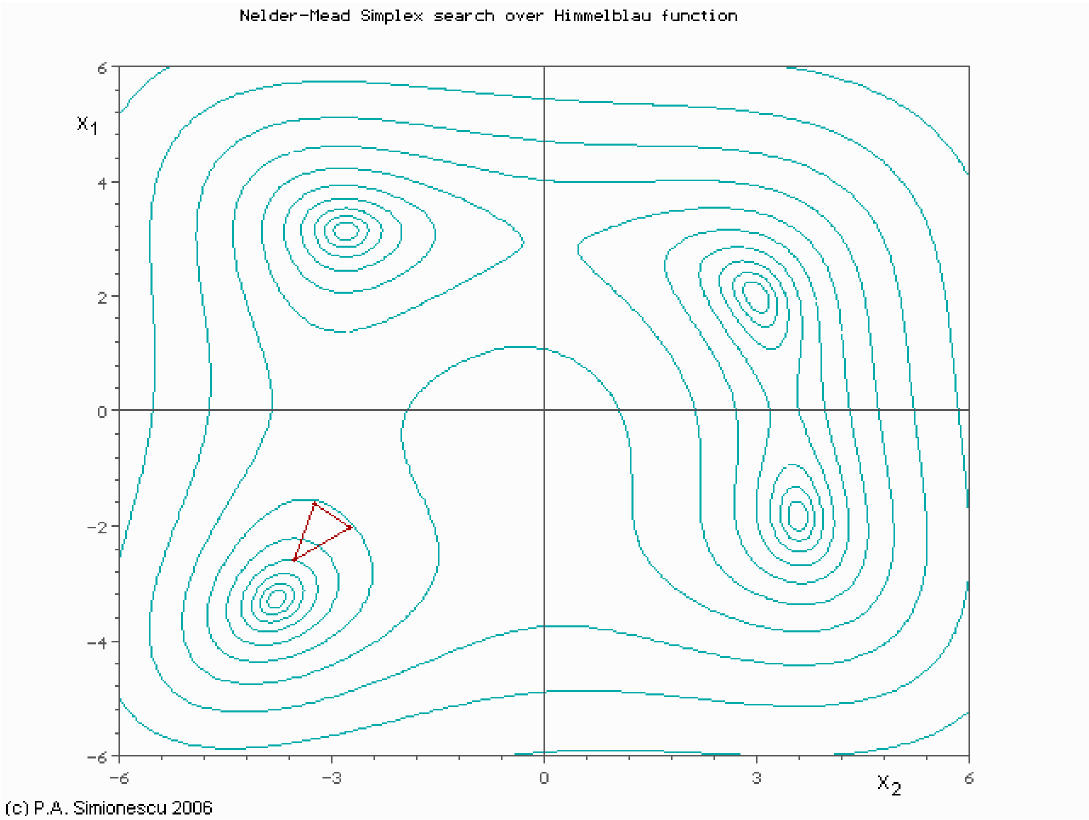
The Nelder-Mead algorithm



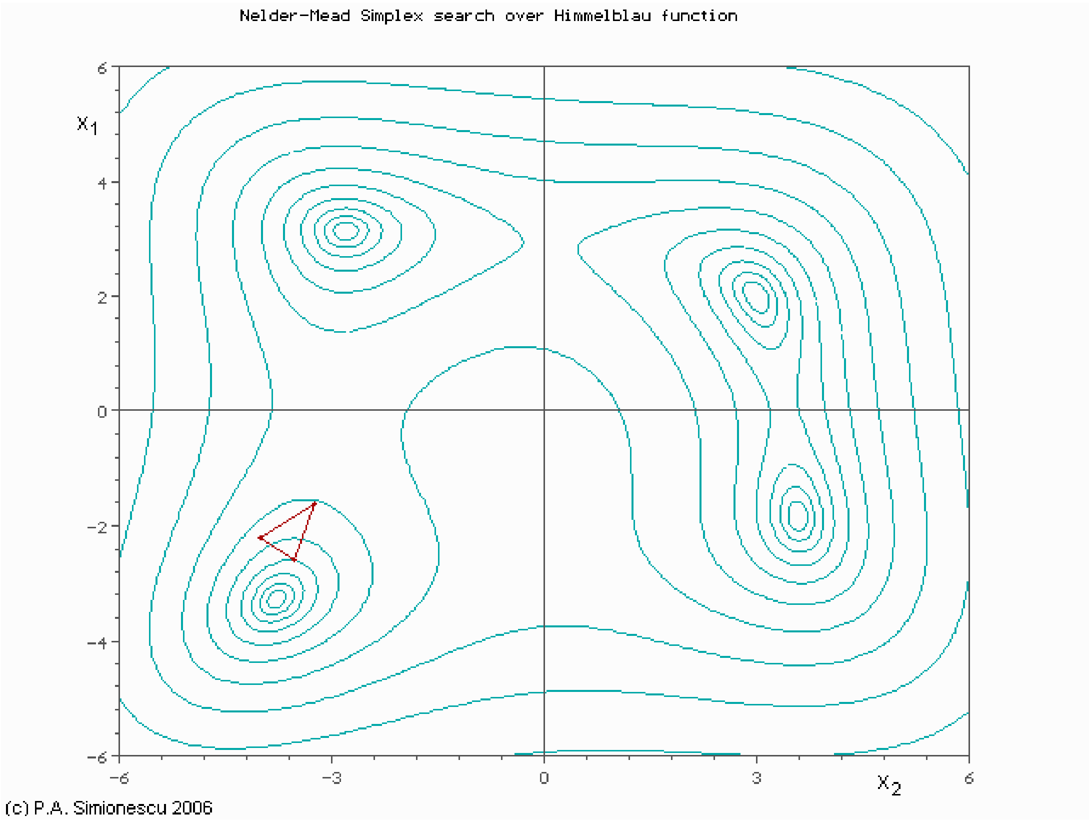
The Nelder-Mead algorithm



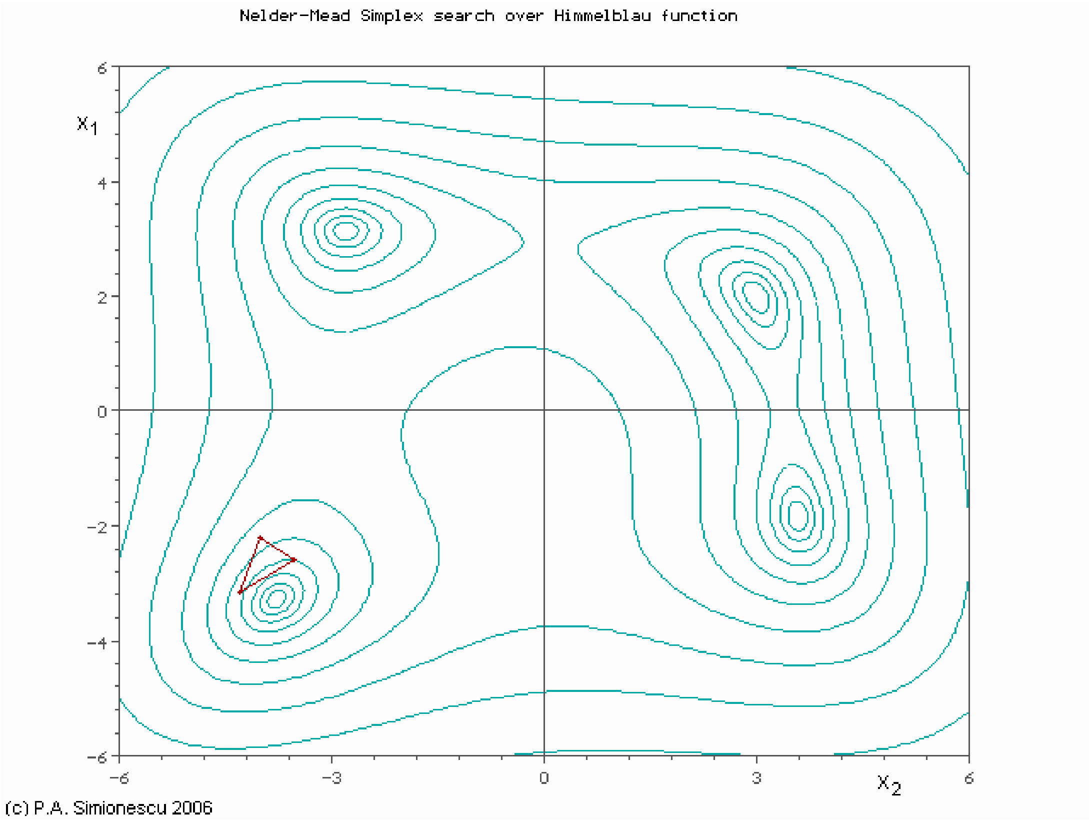
The Nelder-Mead algorithm



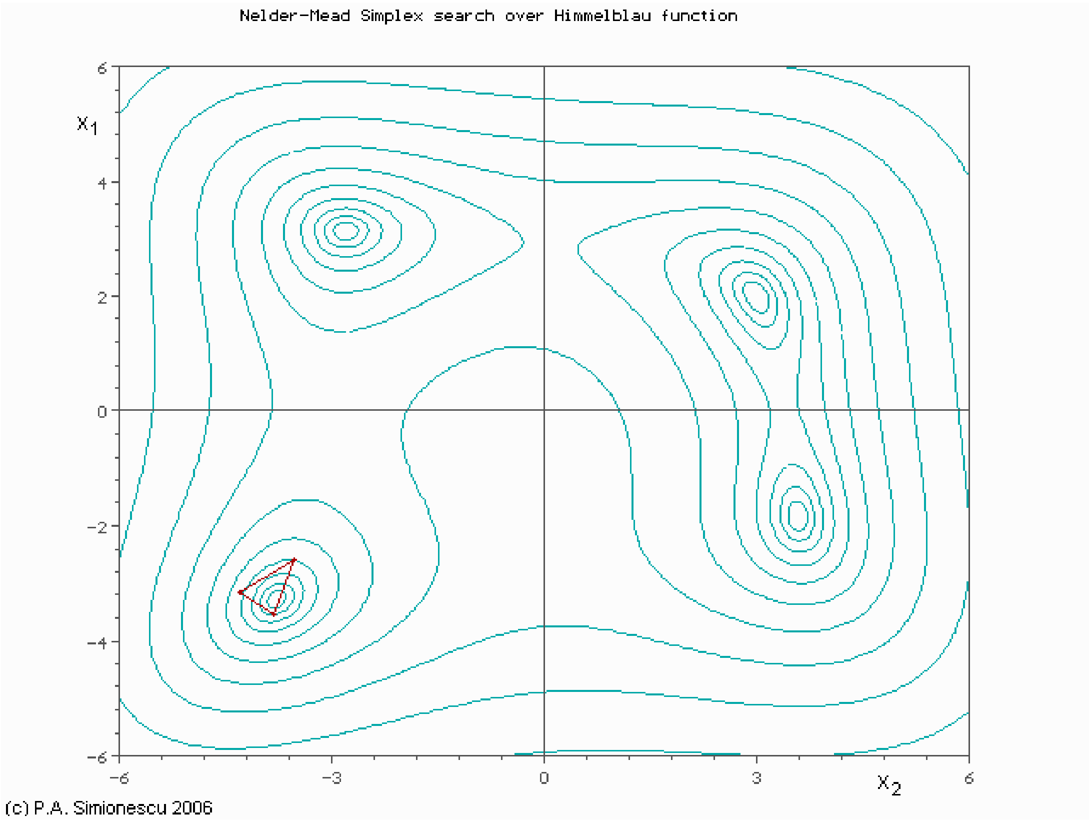
The Nelder-Mead algorithm



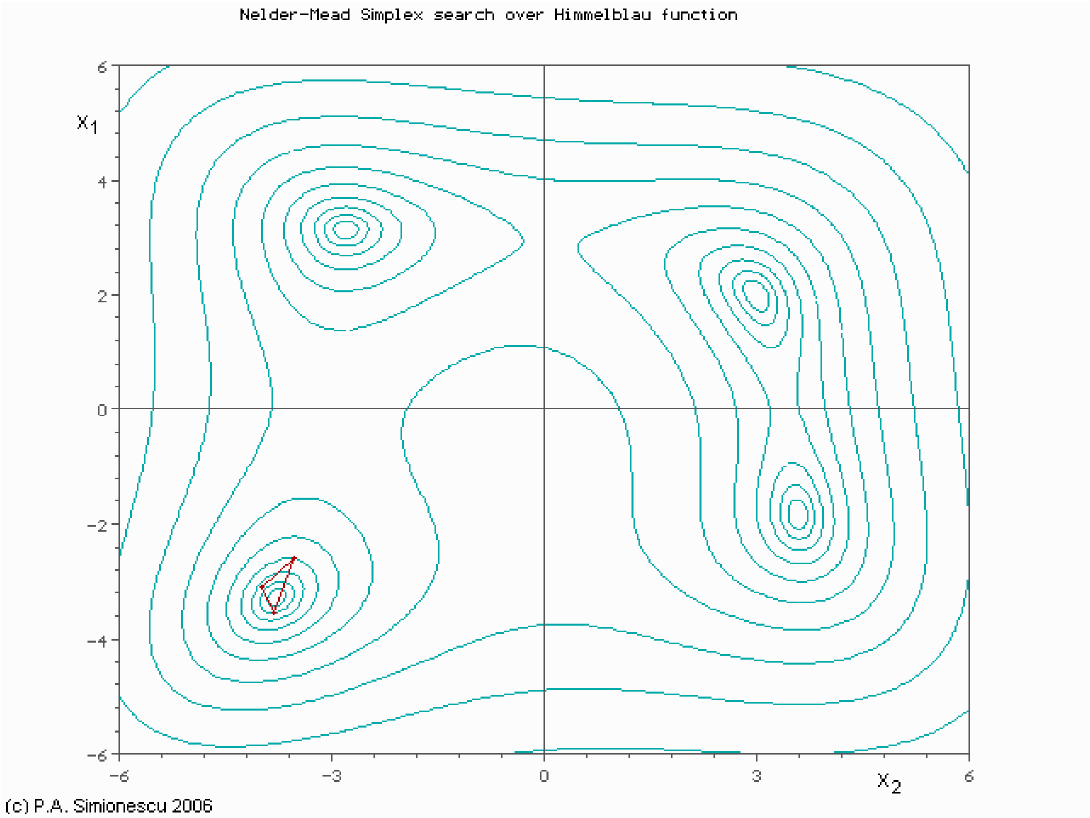
The Nelder-Mead algorithm



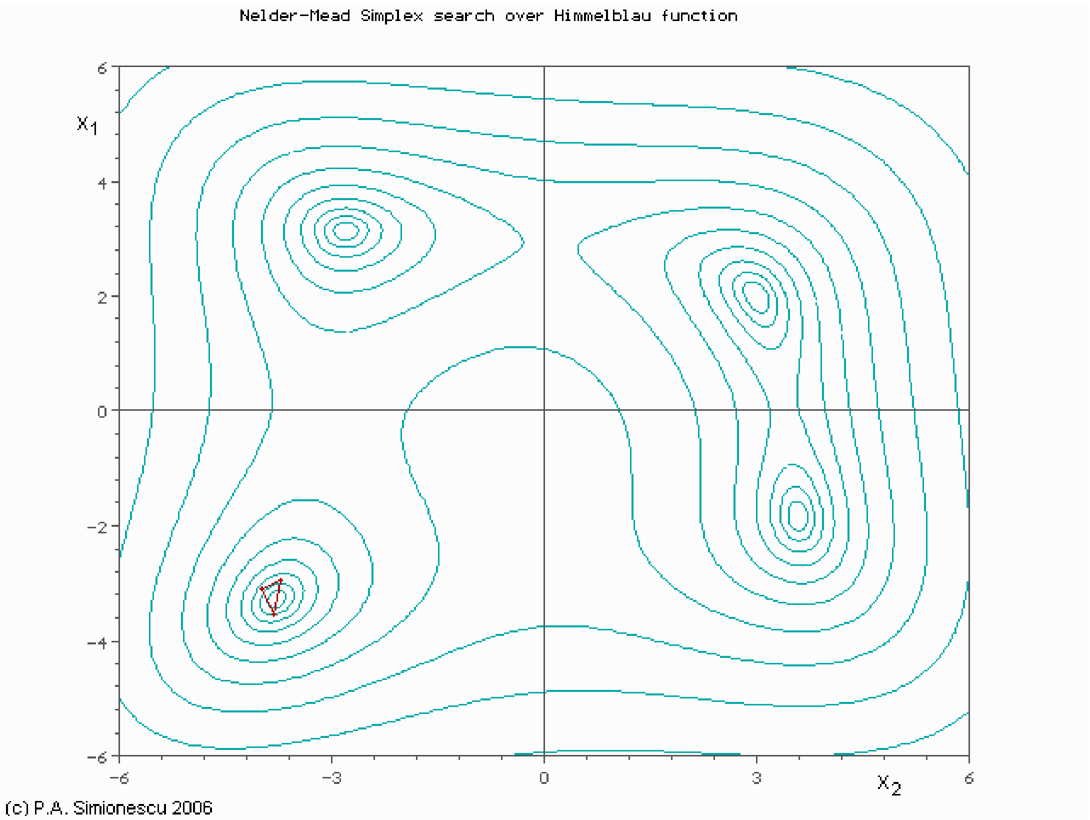
The Nelder-Mead algorithm



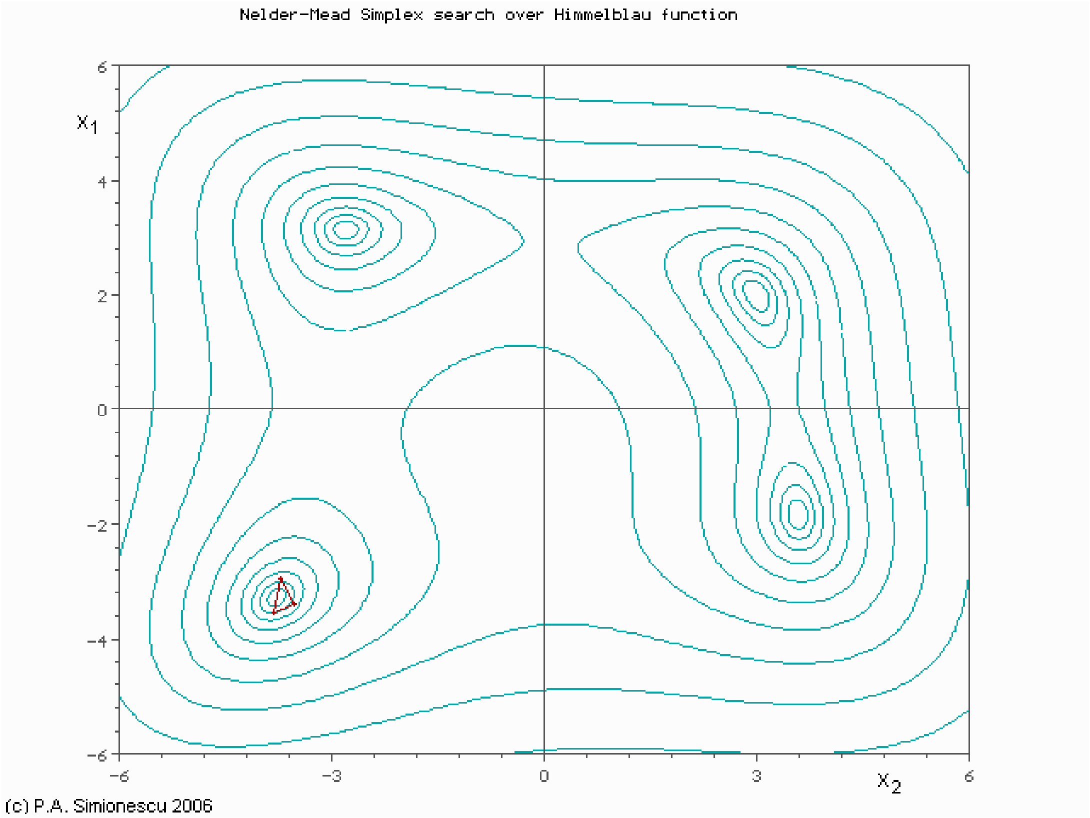
The Nelder-Mead algorithm



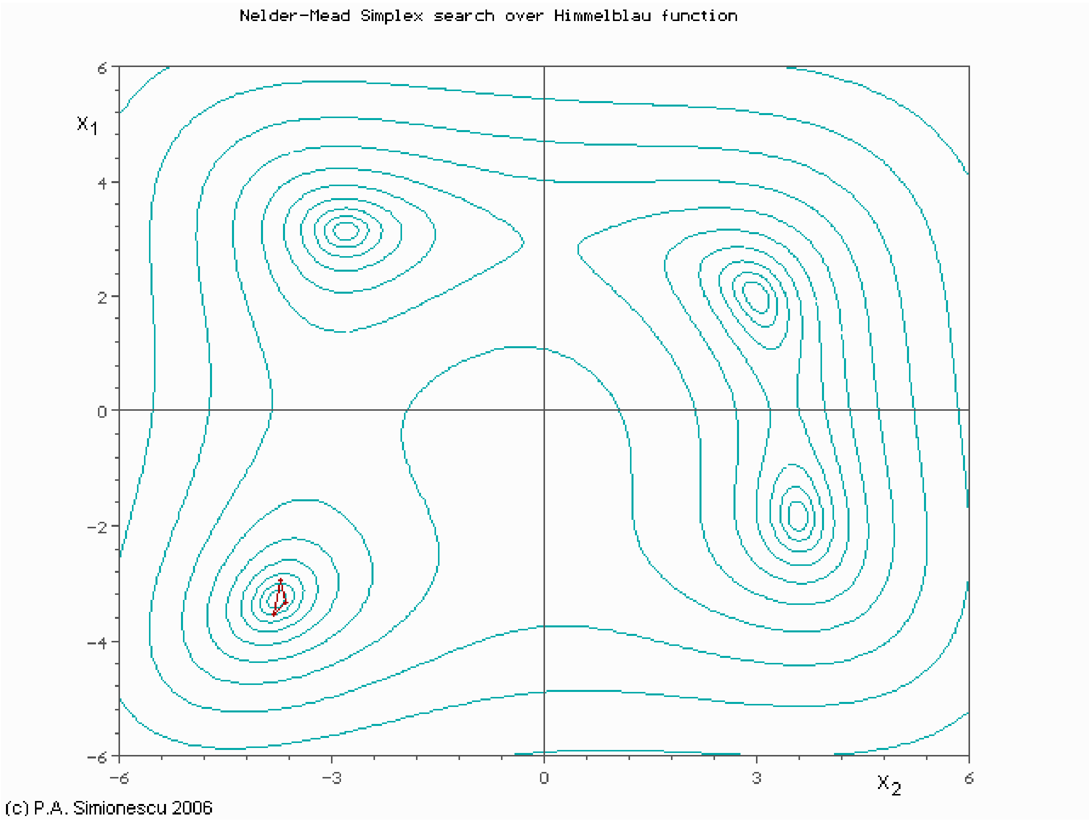
The Nelder-Mead algorithm



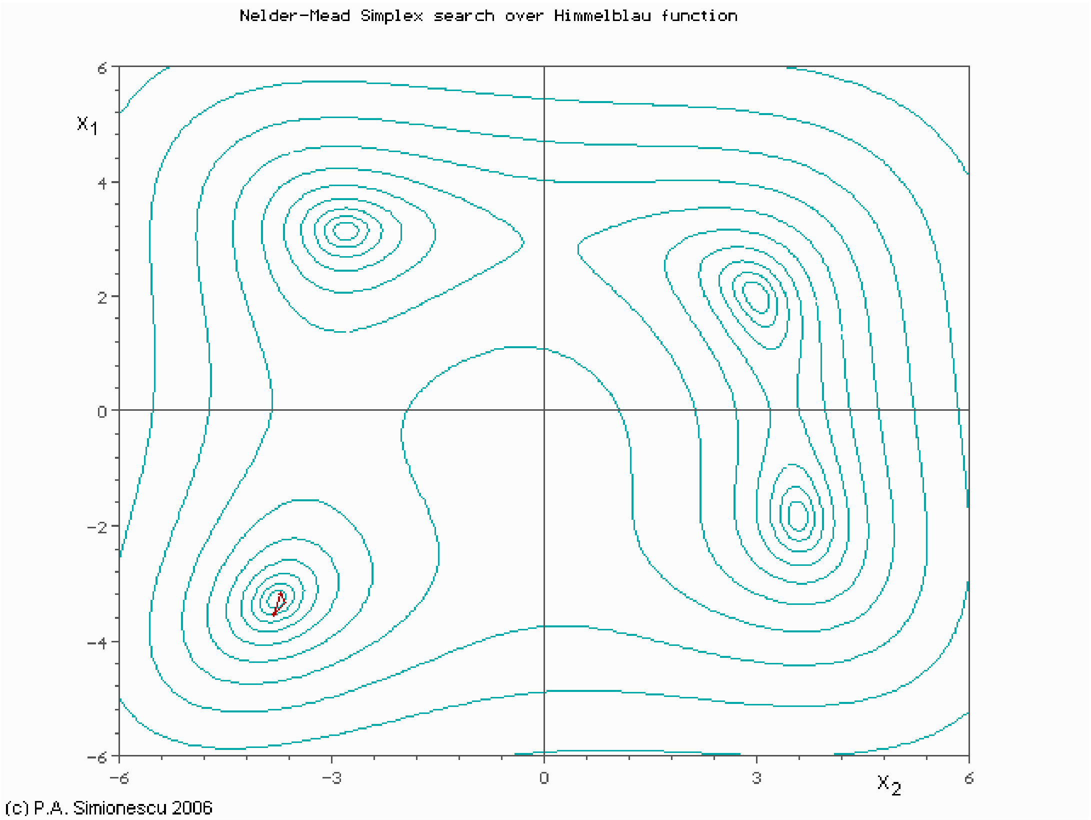
The Nelder-Mead algorithm



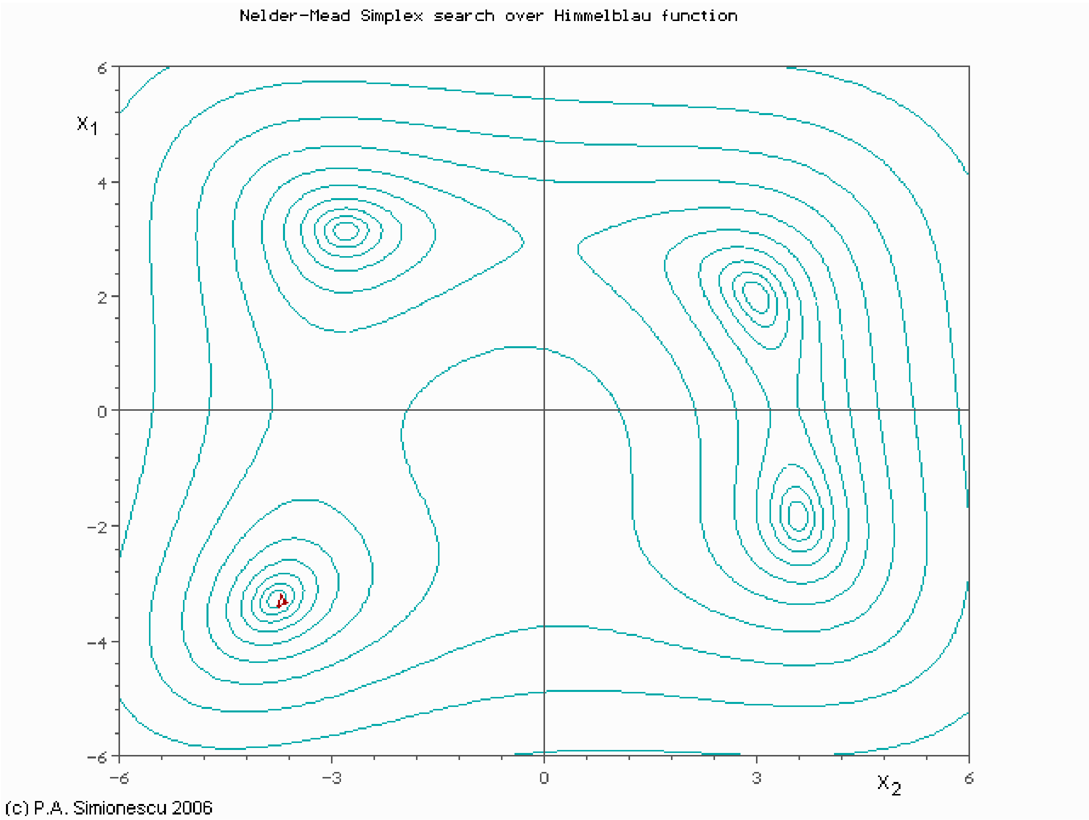
The Nelder-Mead algorithm



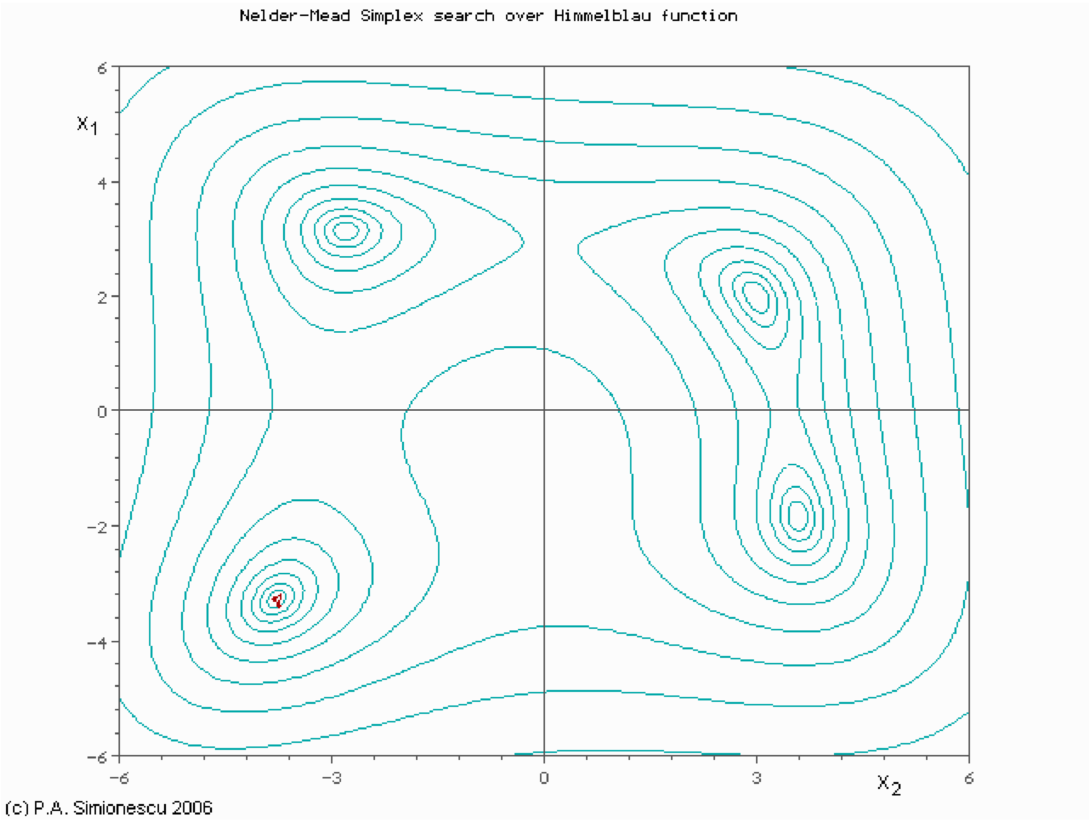
The Nelder-Mead algorithm



The Nelder-Mead algorithm



The Nelder-Mead algorithm



Outline

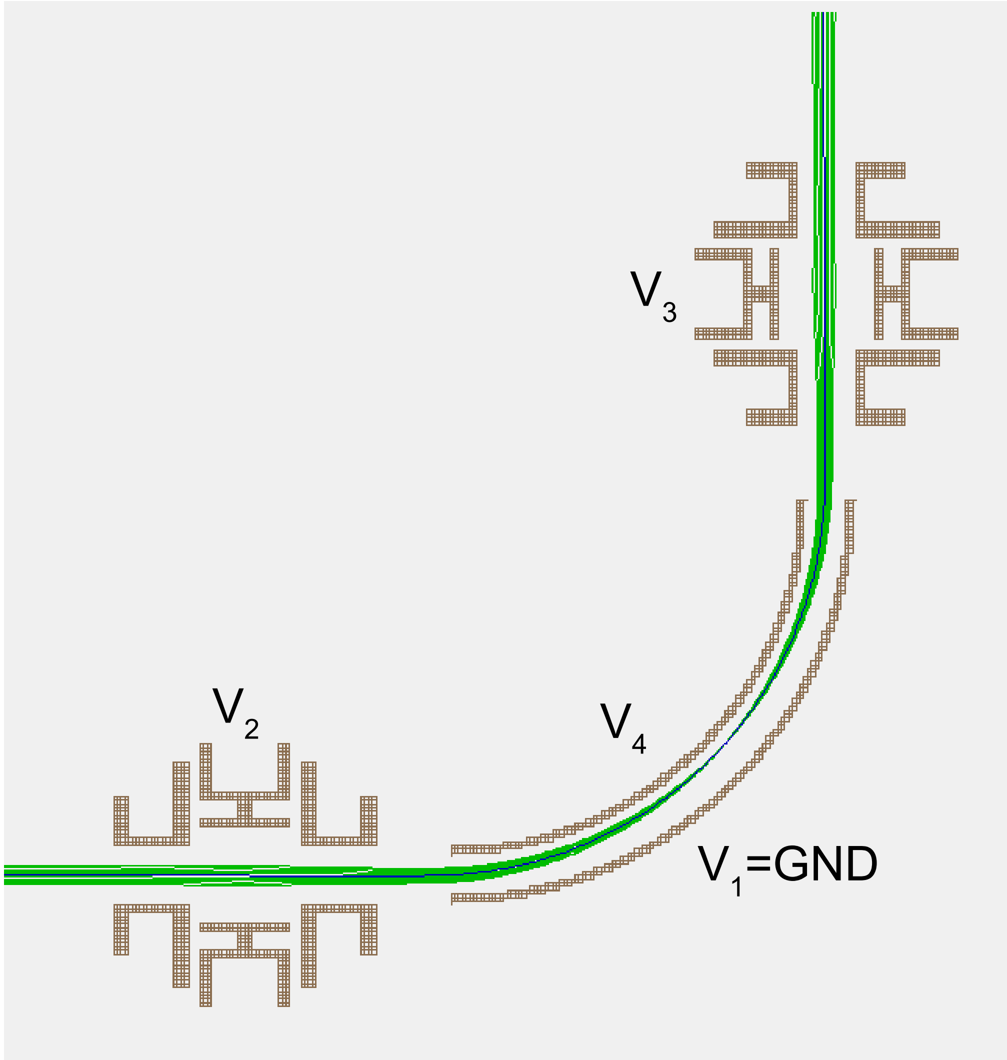
I. The Nelder-Mead Algorithm

II. Potential optimization

III. Geometric optimization

Potential optimization

Potentials:



Potential optimization

Goal function:

```
s = abs(means[2]-EL2x)*1E1 + abs(means[5])*1E1 + (bad_splat/number_of_ions)*1E2
```

position — angle — transmission

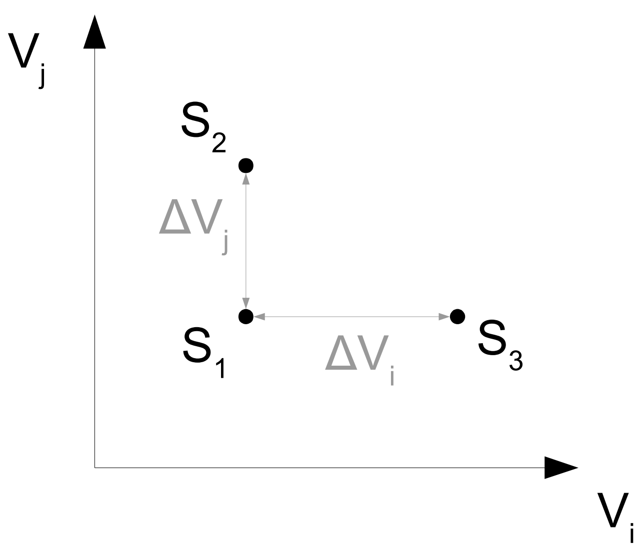
Potential optimization

Goal function:

s = abs(means[2]-EL2x)*1E1 + abs(means[5])*1E1 + (bad_splat/number_of_ions)*1E2

position — angle — transmission

Starting the optimizer:



Optimizer =

Starting point $(V_1, \ldots, V_n)$

+

Variations $(\Delta V_1, \ldots, \Delta v_n)$

+

Convergence radius

Potential optimization

Input beam:

Potential optimization

Input beam:

- Can be gaussian

Potential optimization

Input beam:

- Can be gaussian

or

- Can be made with arithmetic sequences

Potential optimization

Input beam:

- Can be gaussian

or

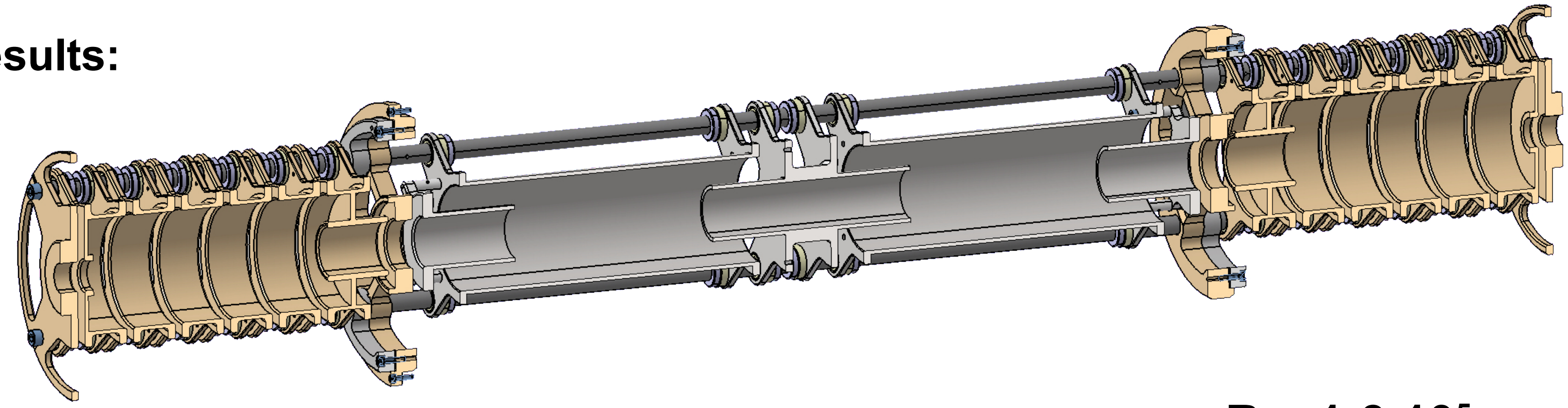
- Can be made with arithmetic sequences

but

It shouldn't be random !

Potential optimization

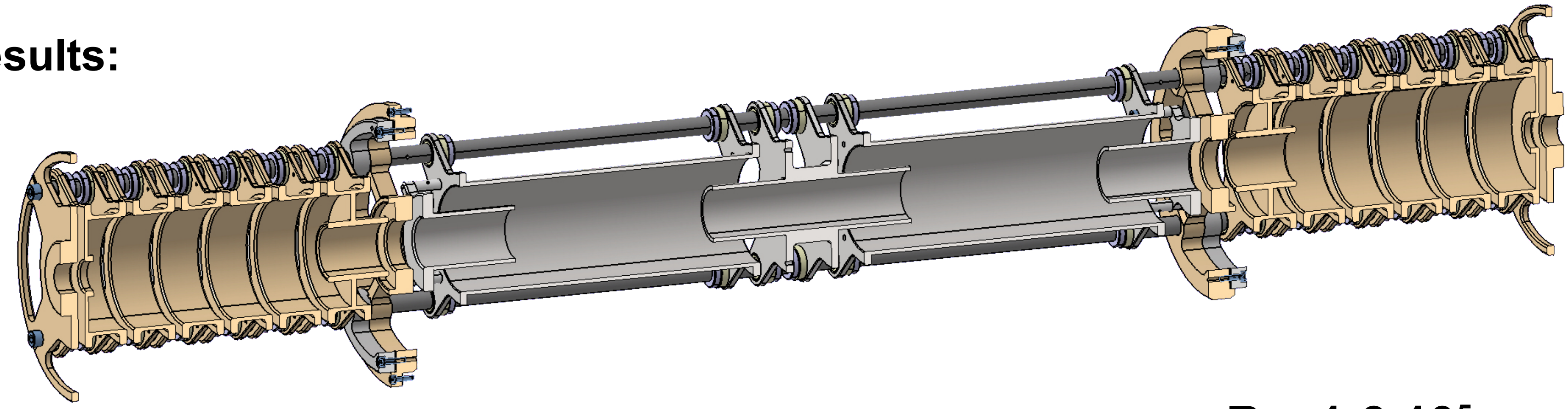
Results:



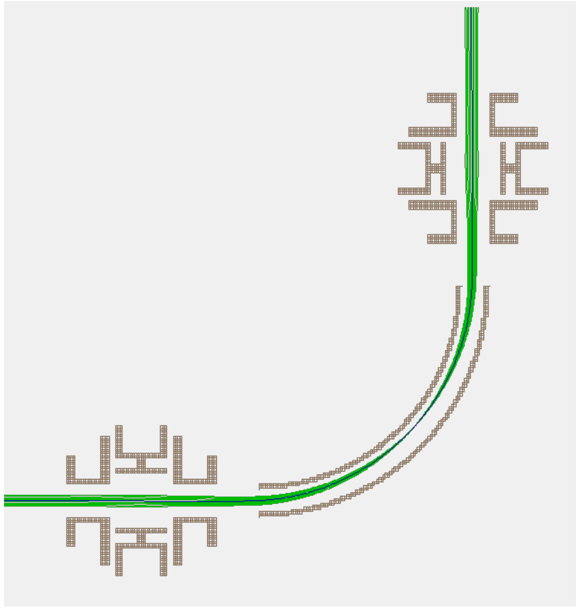
R = 1-3.10^5

Potential optimization

Results:



R = 1-3.10^5



s=25.9411 ➡ s=0.8322

Outline

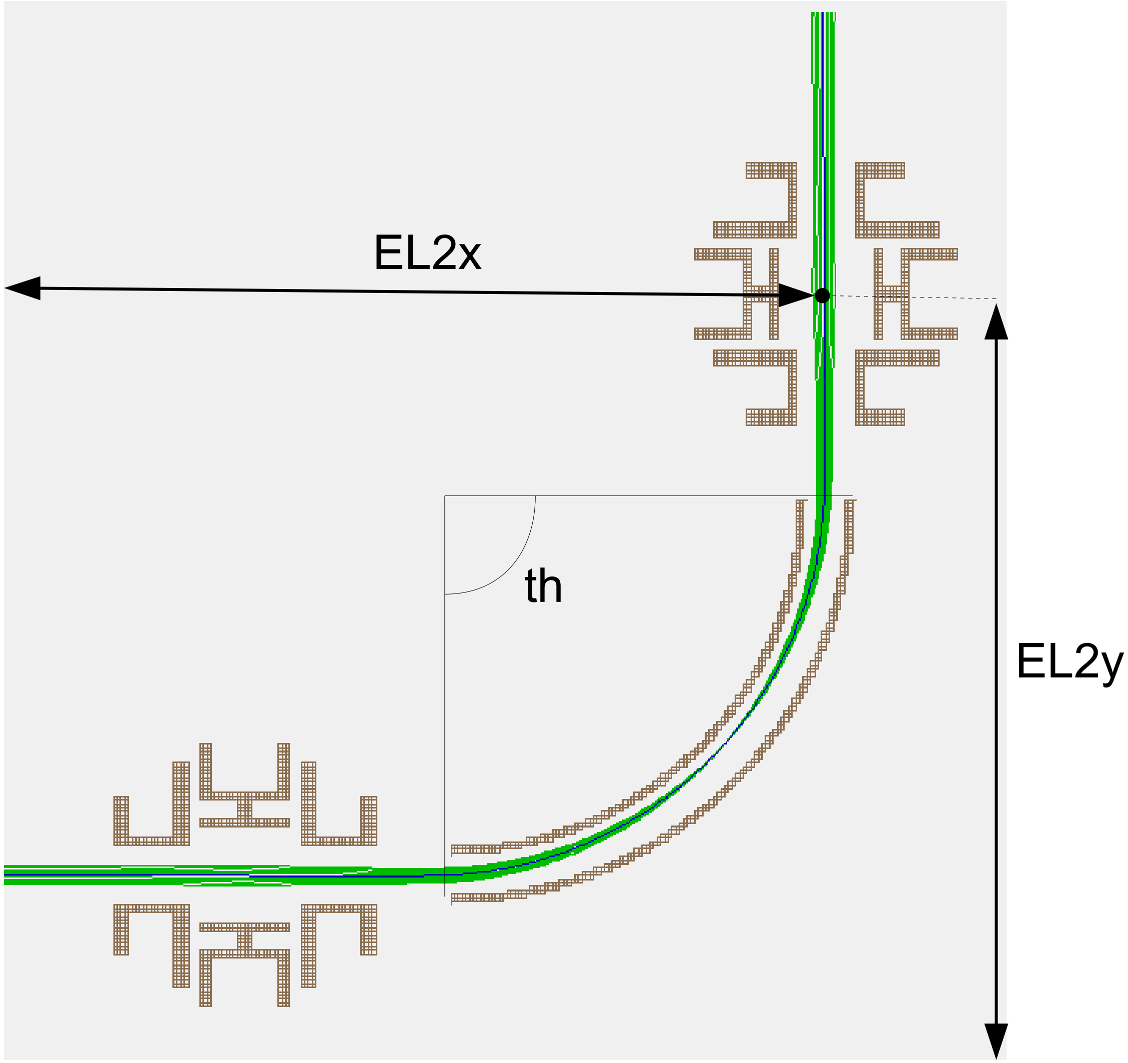
I. The Nelder-Mead Algorithm

II. Potential optimization

III. Geometric optimization

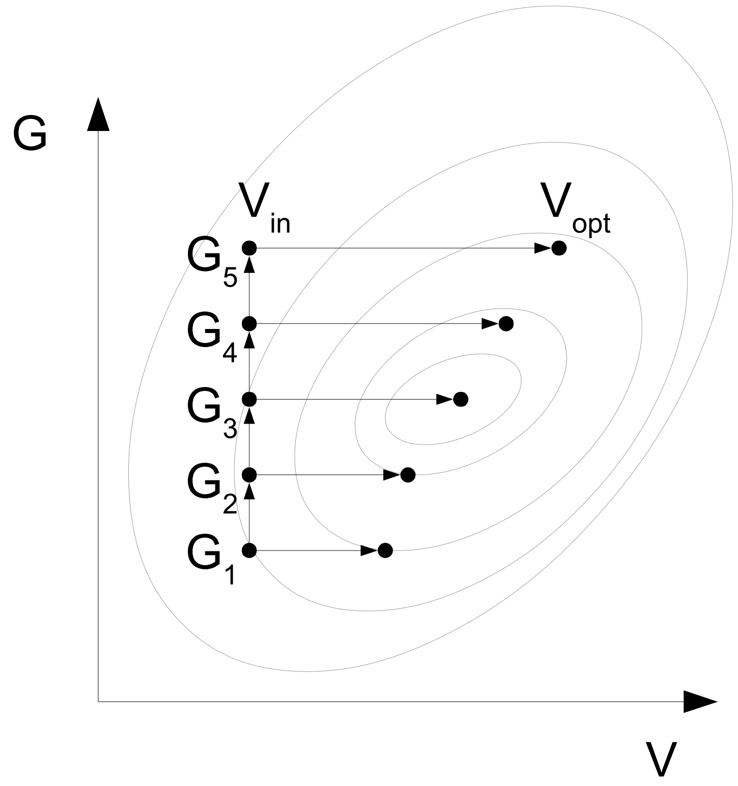
Geometry optimization

Geometry:



Geometry optimization

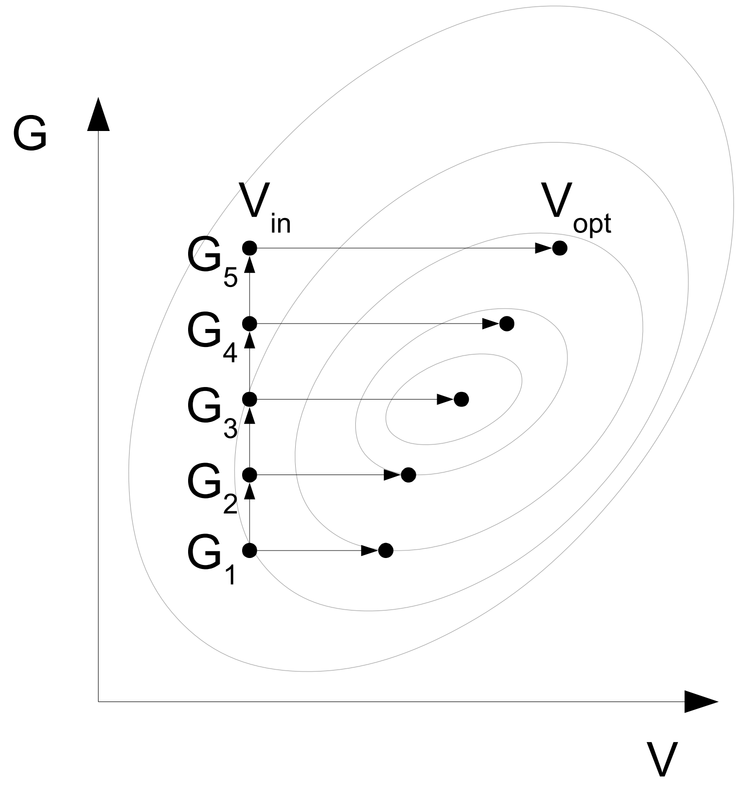
G sweep + V optimization



- Easy to find local minima
- Fast

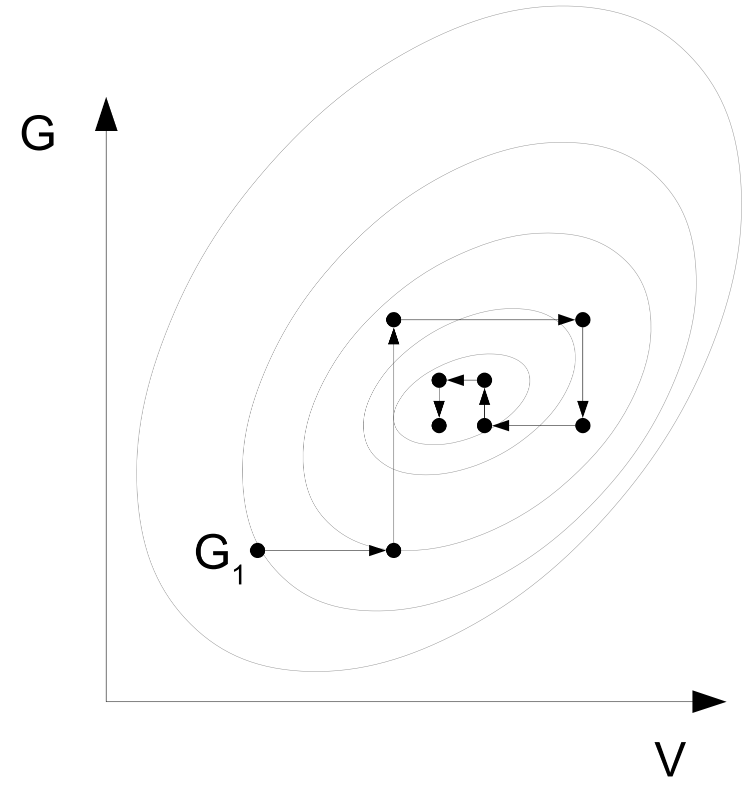
Geometry optimization

G sweep + V optimization



- Easy to find local minima
- Fast

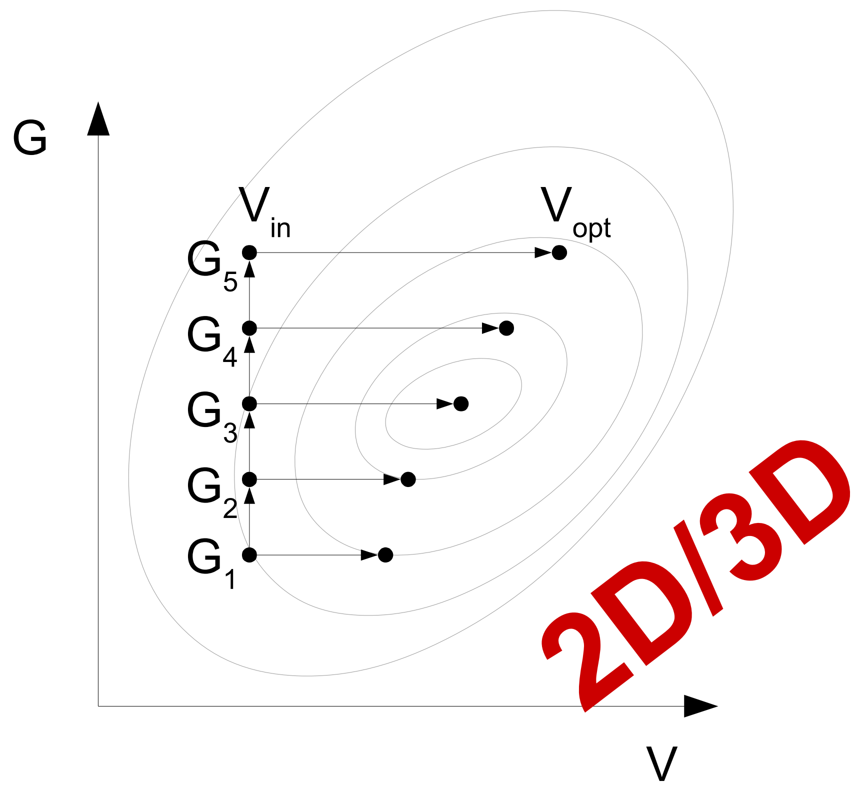
Intricated simplexes



- Finds better, close to absolute minimas
- Slow

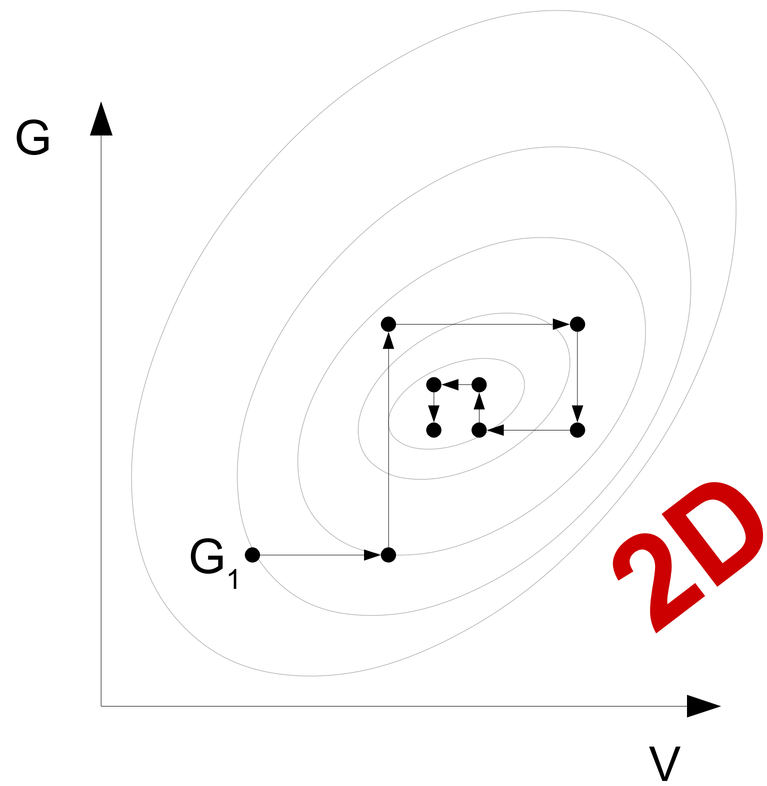
Geometry optimization

G sweep + V optimization



- Easy to find local minima
- Fast

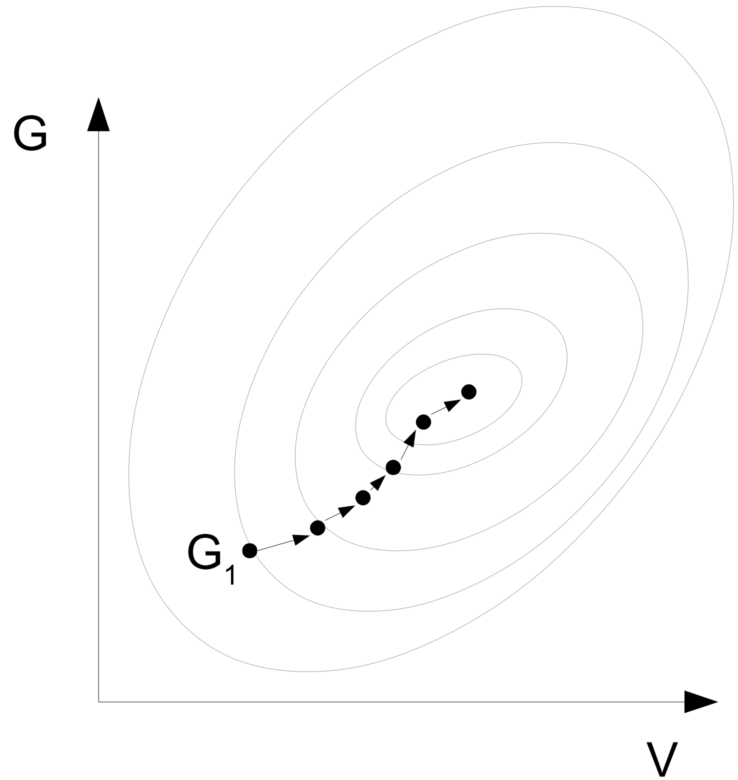
Intricated simplexes



- Finds better, close to absolute minimas
- Slow

Geometry optimization

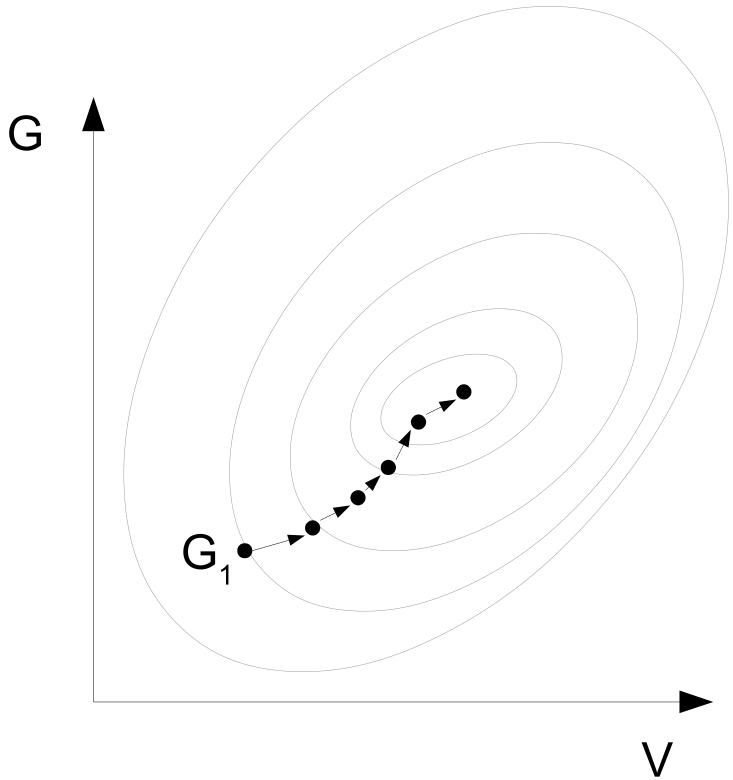
Simplex on G & V



- Can be faster or slower than intricated simplexes
- Still slow
- No feedback

Geometry optimization

Simplex on G & V



2D

- Can be faster or slower than intricated simplexes
- Still slow
- No feedback

Geometry optimization

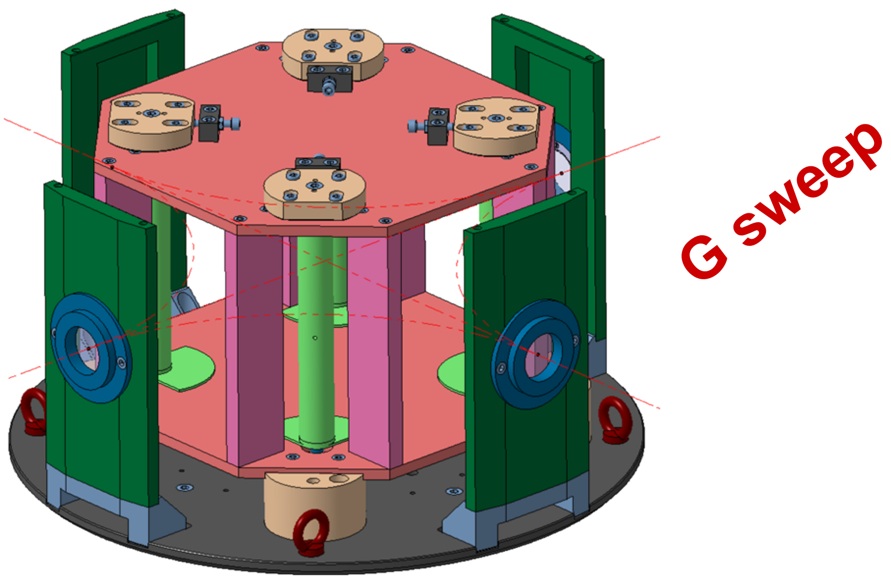
Results:

- Parallelism conservation :

$$\alpha = 0.0017°$$

- ToF spread conservation :

$$ToF_{FWHM} = 0.59\text{ns}$$



Geometry optimization

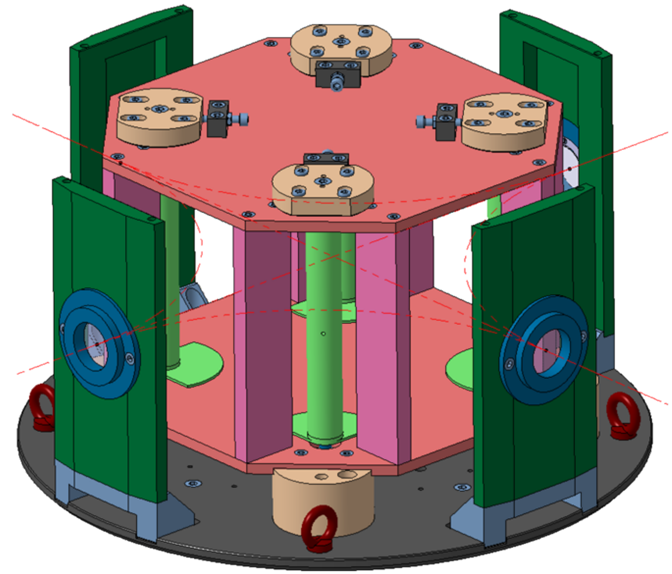
Results:

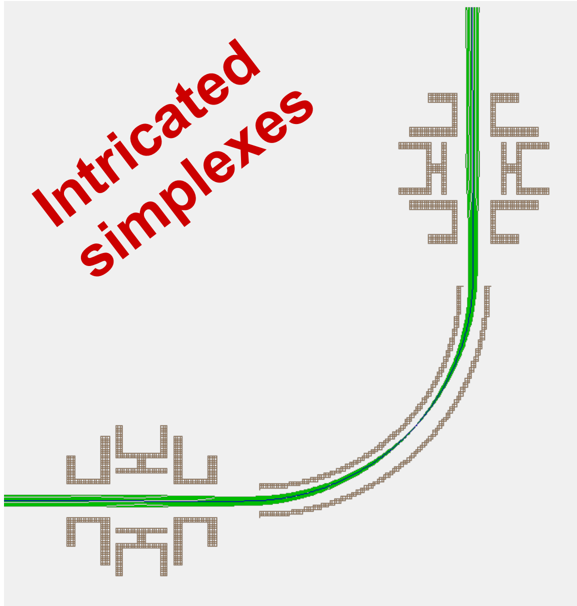
- Parallelism conservation :

$$\alpha = 0.0017°$$

- ToF spread conservation :

$$ToF_{FWHM} = 0.59\text{ns}$$





s=25.9411 → **s=0.8322**

→ **s=0.00001**

Conclusion

- The Simplex optimizer is a very powerfull tool

  → Easy V opt

  → Possible G opt / Beam opt

- Some of its drawbacks can be compensated (restart, randomize&restart, simulated annealing)

- Why is it working ? Not working ?