

Testing optimization tools

Task 1:

Unconstrained functions:

For the unconstrained problems the Gekko library for python were used.

Beal function

```
*****
Steady State Optimization with Interior Point Solver
*****

show more (open the raw output data in a text editor) ...

Solution time : 1.290000000153668E-002 sec
Objective      : 6.352767068812760E-023
Successful solution
-----

x.value
✓ 0.6s
[3.0]

y.value
✓ 0.5s
[0.5]
```

Correct! Solved with Steady State Optimization with Interior Point Solver. $f(3, 0.5) = 0$.

The solution (Objective in the solver) shows a small, this is due to a rounding errors in code, it can be rounded down to 0.

Himmelbaas

```

-----
Steady State Optimization with APOPT Solver
-----

show more (open the raw output data in a text editor) ...

Solution time   :   1.270000000658911E-002 sec
Objective       :   1.872974929247235E-013
Successful solution
-----

x.VALUE
✓ 0.7s
[2.9999999655]

y.VALUE
✓ 0.3s
[1.9999999262]

```

Correct! The solver found the solution $f(3, 2) = 0$. This is one of 4 solutions to the problem. The solver did not recognize this. The program had the same rounding error as the previous task

Constrained Optimization:

Rosenbrock, cubic and line constrained

```

if __name__ == '__main__': ...
    fun: 1.3412162831045345e-10
    jac: array([ 0.0004698 , -0.00022984])
    message: 'Optimization terminated successfully'
    nfev: 31
    nit: 9
    njev: 9
    status: 0
    success: True
    x: array([1.00000058, 1.          ])

```

Correct answer! The SciPy library in python is used in this problem. It uses Sequential Least Squares Programming to solve the problem. $f(1,1) = 0$

RosenBrock, disk constraints

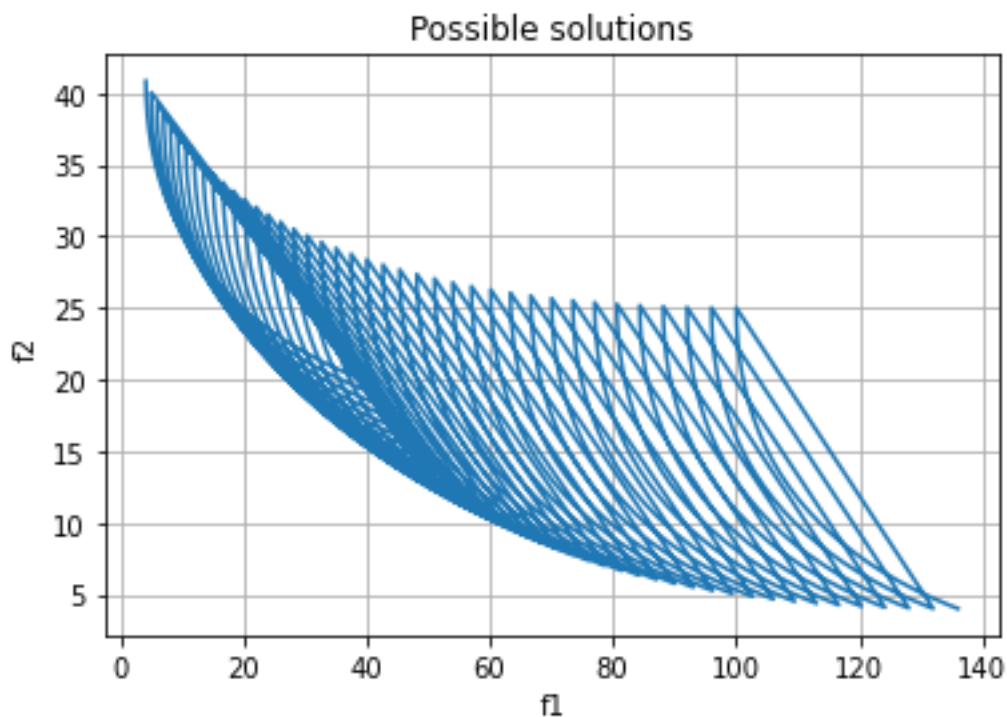
```
if __name__ == '__main__': ...  
    fun: 3.608907291128452e-08  
    jac: array([ 0.00229613, -0.0013219 ])  
    message: 'Optimization terminated successfully'  
    nfev: 71  
    nit: 22  
    njev: 22  
    status: 0  
    success: True  
    x: array([0.99982193, 0.99963726])
```

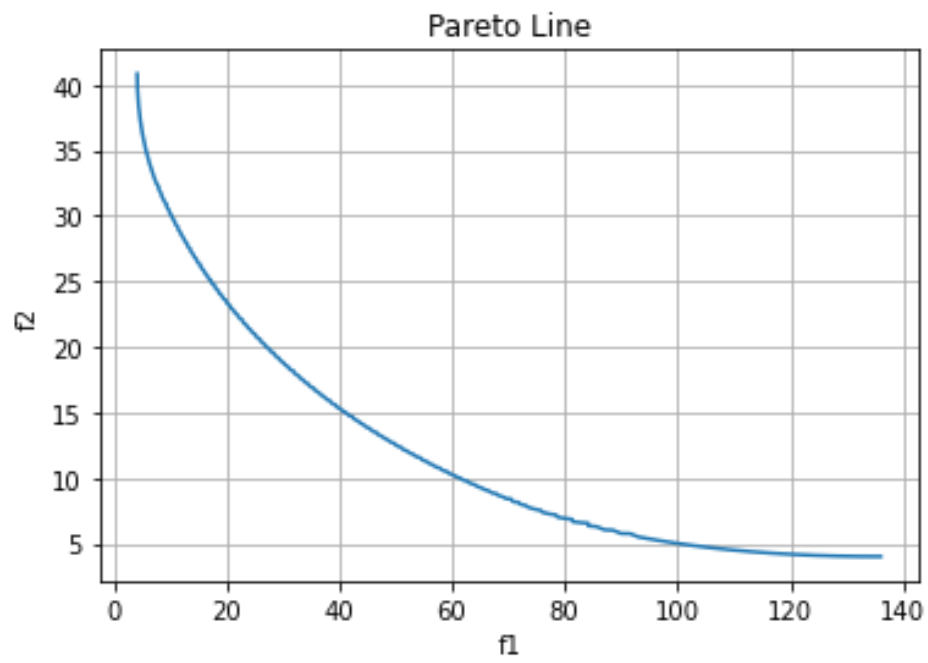
The same method is used in this problem as the previous one. It finds the right answer, although with a rounding error. $f(0,0) = 1$

Task 2:

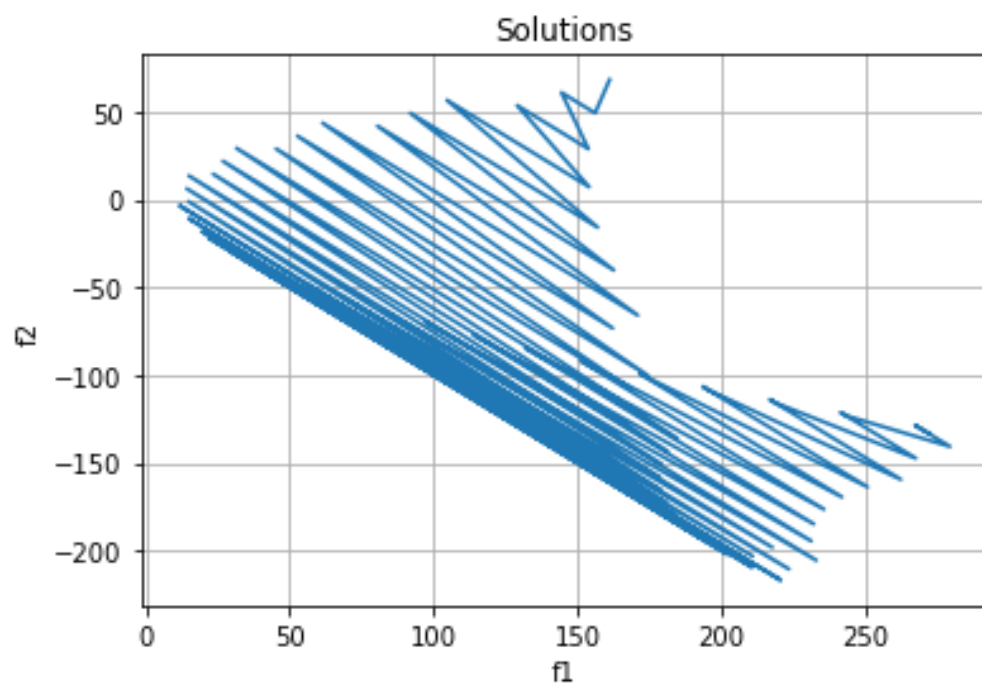
With the possible solutions plotted, it's clear where the pareto line should go. A python function for generating a pareto line was found at <https://code.activestate.com/recipes/578230-pareto-front/>

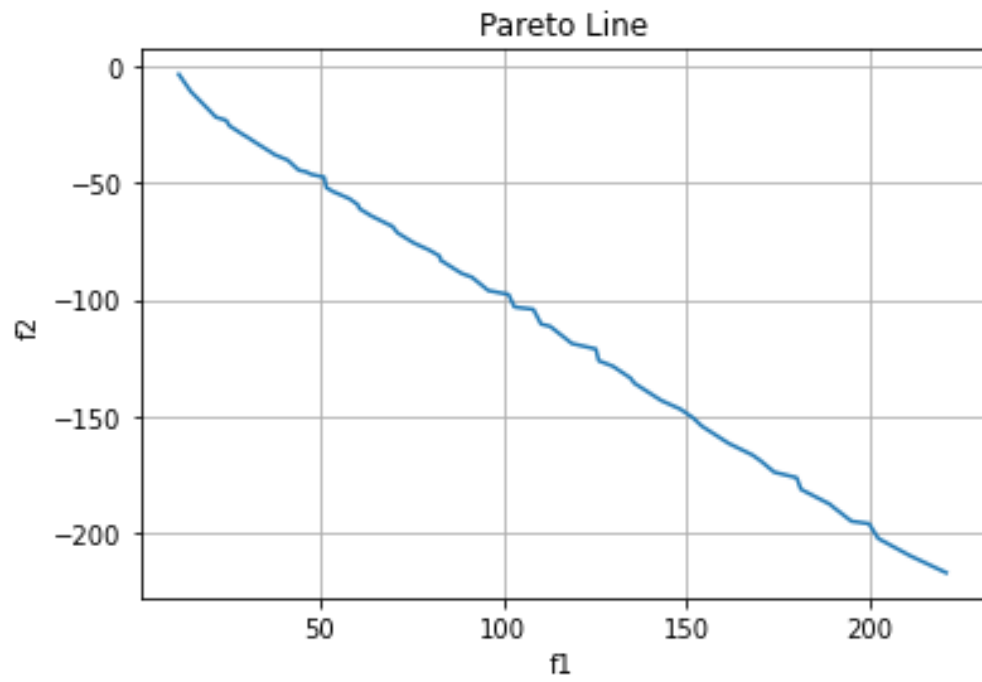
Binh and Korn function





Chankong and Haimes:





For both of the problems we can easily see that the correct pareto line is plotted. However, the solution is not perfect. Due to errors in the calculation at the start. The first 5% of the data had to be excluded-