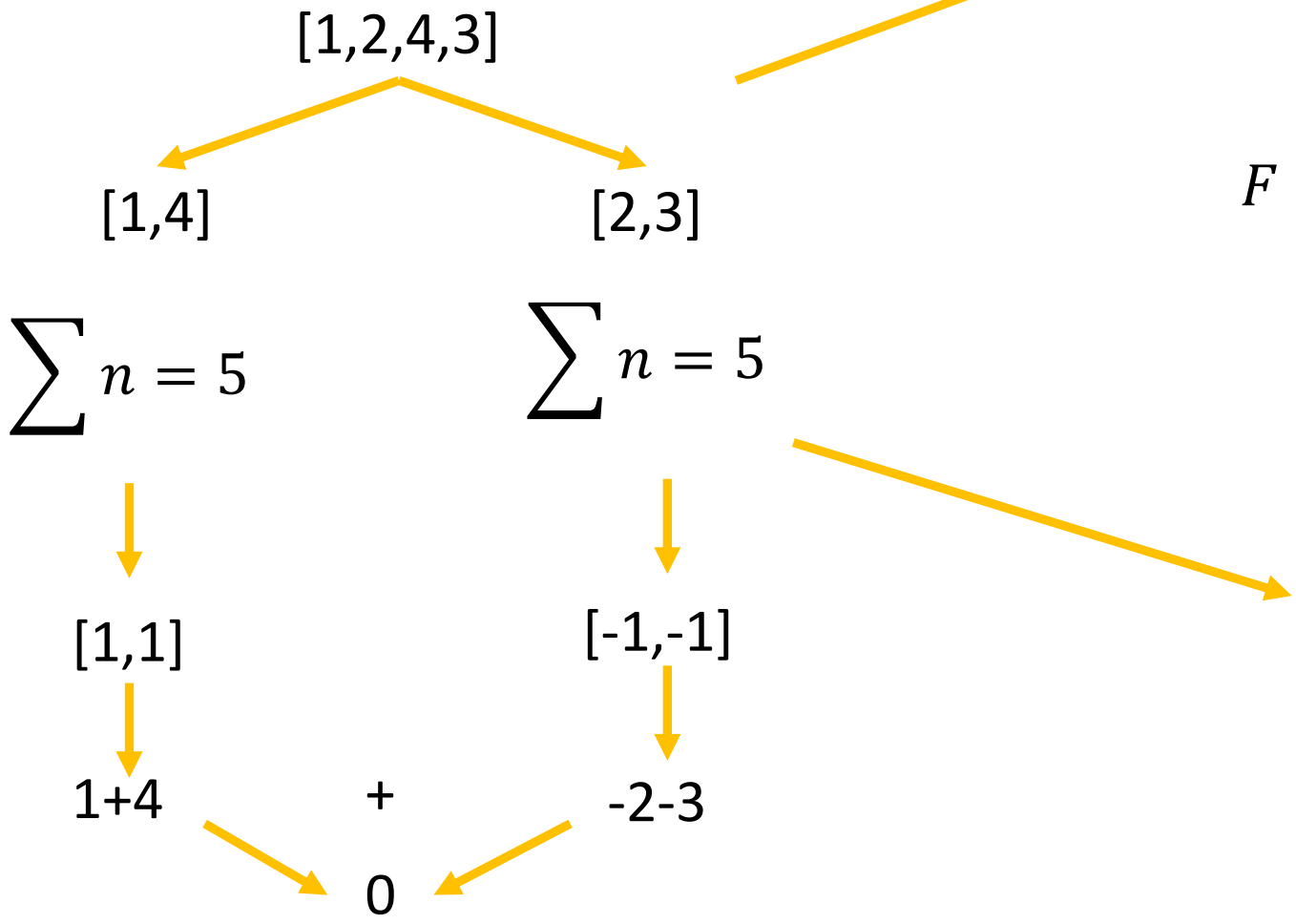
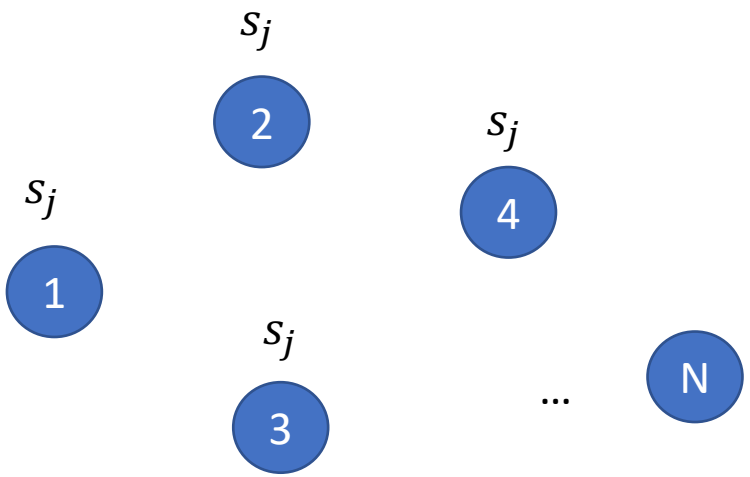
Number partitioning problem

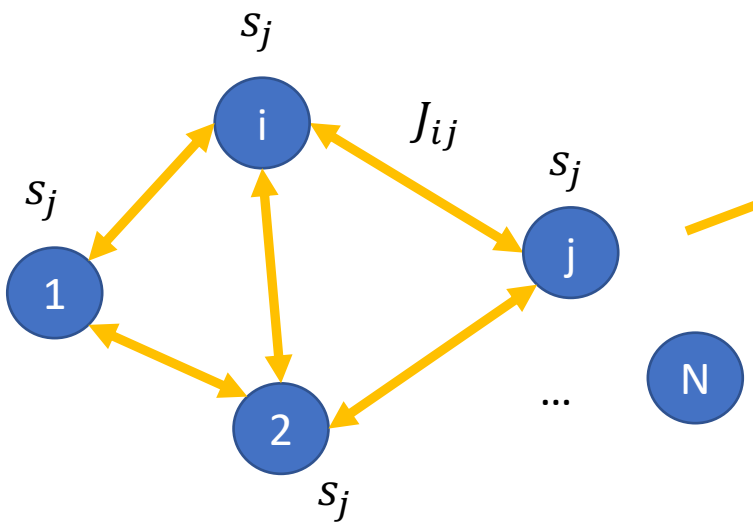
$$F = \sum_{j=1}^{N} n_j * s_j , \qquad s_j = [1, -1]$$

$$F = \left(\sum_{j=1}^{N} n_j * s_j \right)^2 , \qquad s_j = [1, -1]$$





Ising formulation



$$H = -\sum_{\substack{j=1 \\ i<j}}^{N} J_{ij} * s_i s_j - \sum_{i=1}^{N} h_i s_i \,, \qquad s_{i,j} = [1, -1]$$

$$F = \left(\sum_{j=1}^{N} n_j * s_j \right)^2 , \qquad s_j = [1, -1]$$

$$\mathrm{F} = (n_1 s_1)^2 + (n_2 s_2)^2 + 2 n_1 n_2 s_1 s_2$$

$$J_{ij} = 2 * n_i n_j$$

$$F = \sum_{\substack{j=1 \\ i<j}}^{N} 2 n_i n_j * s_i s_j + \sum_{i=1}^{N} (n_i s_i)^2$$

Const

1

Gradient descent

$$F = \left(\sum_{j=1}^{N} n_j * s_j \right)^2, \qquad s_j = [1, -1]$$

$F = 0 \rightarrow set\ can\ be\ devided$

$Find\ optimal\ [s_j]$

$$s_j = s_j - \lambda \nabla F$$

$s_j = [-1, -1, 1, 1]$

One shot learning

```
s_loop = np.array(s) - lamb * 2 *
np.dot(n,s) * np.array(n)

s_loop = np.sign(s_loop )
```

```
Number set is: [2 3 4 1]

Optimal s: [-1. -1.  1.  1.]
H(s) = 0.0
The partioning is: R1 = [4 1], R2 = [2 3]

Additional: number of random initializations: 19, lambda = 0.0001

Calculation time: 0.015455722808837891
```

```
Number set is: [68.42802071 58.47154548 56.66436102 56.48909859 14.51242583 68.52094558
 73.66676465 25.04797913 75.80112407 38.51513211 15.52501584 45.85045232
 17.6535574  88.02315769 45.63895676  1.33618909 13.37848647 40.99404662
 96.35439474 18.01969741 14.35154552 20.57371445 81.763313   53.21350435
 18.03460714 47.63648885 78.65793986 86.49330755 27.9567458  60.75346262
 35.3262304  45.24026311 68.75622667 37.04450078 16.60620128 14.14243637
  2.15011946 73.70459677 65.12994648 96.98161034 99.49446244 31.33267952
 63.56113804 53.26611235 22.77601576 23.35417552 46.25060642 92.75512529
 68.1819234  49.12683609 27.2889032  26.61841768  3.64107945 20.55402584
 96.22209884 95.22877039 61.03453455 87.77504459 57.04023275 91.15994168
 51.25525216 18.76288769 44.43329654 12.97332087 72.33895199 52.86283578
 60.60562594 26.040787   58.25282562  0.44037819 43.10743067 35.43427232
 93.18683573 29.70675712 20.73440319 17.644898   78.08774641 18.64744513
  3.095893    3.83793712 65.9597135  86.5196912  63.25631448 74.57852294
 38.69507607 18.20124025 59.21300869 97.38828093  0.36306191 91.84046178
 74.6872392  82.45790343  7.44322529 98.47120326 54.527571   85.13872263
 14.35277333 84.03773191  2.69471303 92.21419794]

The set can't be partioned
Optimal s: [ 1. -1.  1.  1.  1. -1.  1.  1. -1. -1.  1.  1.  1. -1. -1.  1. -1.  1.
 -1. -1.  1. -1.  1. -1. -1.  1.  1.  1. -1. -1. -1.  1.  1. -1.  1. -1.
 -1. -1.  1.  1. -1. -1.  1.  1. -1. -1. -1. -1.  1. -1.  1.  1. -1.  1.
  1.  1. -1. -1.  1. -1.  1. -1.  1.  1. -1.  1.  1. -1. -1.  1. -1.  1.
  1. -1.  1. -1.  1.  1. -1.  1. -1.  1. -1.  1.  1. -1.  1. -1.  1. -1.
 -1. -1.  1. -1. -1.  1.  1. -1. -1.  1.]
H(s) = 0.0010665392399864702

Additional: number of random initializations: 4999, lambda = 0.0001

Calculation time: 0.17900562286376953
```

```
Number set is: [3619.91487622 1340.04851053 3038.31556612 8450.39860725  544.53528509
 4160.52695752 3967.60884604 2525.31510053 4928.7226475  7613.99247938
 2398.92060767 9902.39577475 5971.65795933 6021.20760639 8673.78960344
 2912.44715296 6186.0890503  9696.43549356 5877.53059725 1930.1920836
 2512.25166778 4908.52629889 2712.31456507 7633.6549791  6344.82352823
 3943.95481819 6870.13185075 2597.59816105 9088.89888508 1673.70212949
 5181.59287381 5789.47869812 6058.13389807 2195.54374539 1391.2769105
 7523.08788643 9130.8081112  7640.53422447 6048.25065297 9677.71269853
 4163.41149297 3700.34285623   54.02062683 9991.47829456 9775.44491676
  999.13440351 7338.05489952 9135.78139758 8117.54235314 6844.84203936
  769.52817943  904.76576358 3381.60553767 4607.02268388 2464.36643729
 3473.78170162 5356.60463517 4795.89593383 7269.02646544 5236.27020104
  243.04273343 6958.23120802 2659.60091027 3988.20546433 6123.38295662
 7770.73125887 8204.21145259  429.57758027 3151.31153049 8360.96520611
 4194.75206196 2006.34097378 5534.88960494 8434.94912416 1495.51473497
 9890.85354347 7408.87150307 6244.31284224 9559.03824179 3947.29545184
 4502.37357817 6629.17497198 6980.25323344 2438.36731031 5641.61746875
 2375.63663342 5654.61445263 7312.90099748 2329.59501459 9507.76448924
 3672.21854585 8224.06048878 6326.91002688  326.60696491 8862.12502089
 1725.97498416 9967.01959568 5723.24583362 5700.907207   8155.59696387]

The set can't be partioned
Optimal s: [-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1.  1. -1. -1. -1. -1. -1. -1.]
H(s) = 274175588498.4974

Additional: number of random initializations: 4999, lambda = 0.0001

Calculation time: 0.17978954315185547
```

Increase # of initializations

```
The set can't be partioned
Optimal s: [-1. -1. -1. -1. -1.  1. -1.  1.  1. -1.  1.  1.  1.  1. -1. -1.  1.  1.
 -1. -1. -1. -1. -1.  1.  1.  1.  1.  1. -1.  1.  1. -1. -1.  1. -1. -1.
  1. -1. -1.  1.  1.  1. -1. -1. -1. -1.  1. -1. -1.  1. -1. -1. -1. -1.
  1.  1. -1.  1.  1. -1.  1.  1. -1. -1. -1. -1.  1.  1.  1.  1.  1. -1.
  1.  1.  1.  1. -1. -1.  1.  1. -1.  1. -1. -1. -1.  1. -1.  1.  1. -1.
  1. -1.  1. -1. -1. -1. -1. -1. -1. -1.]
H(s) = 0.07537319705116137

Additional: number of random initializations: 99999, lambda = 0.0001

Calculation time: 3.4440109729766846
```

Hoping on randomness (

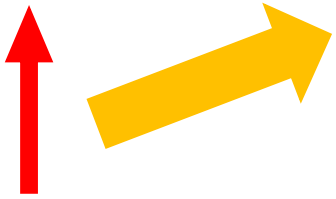
One shot learning problem

$$Increase\ \#\ of\ initializations \rightarrow randomness \rightarrow calculation\ time\ growth$$

$$Need\ to\ decrease\ calculation\ time\ as\ much\ as\ possible$$

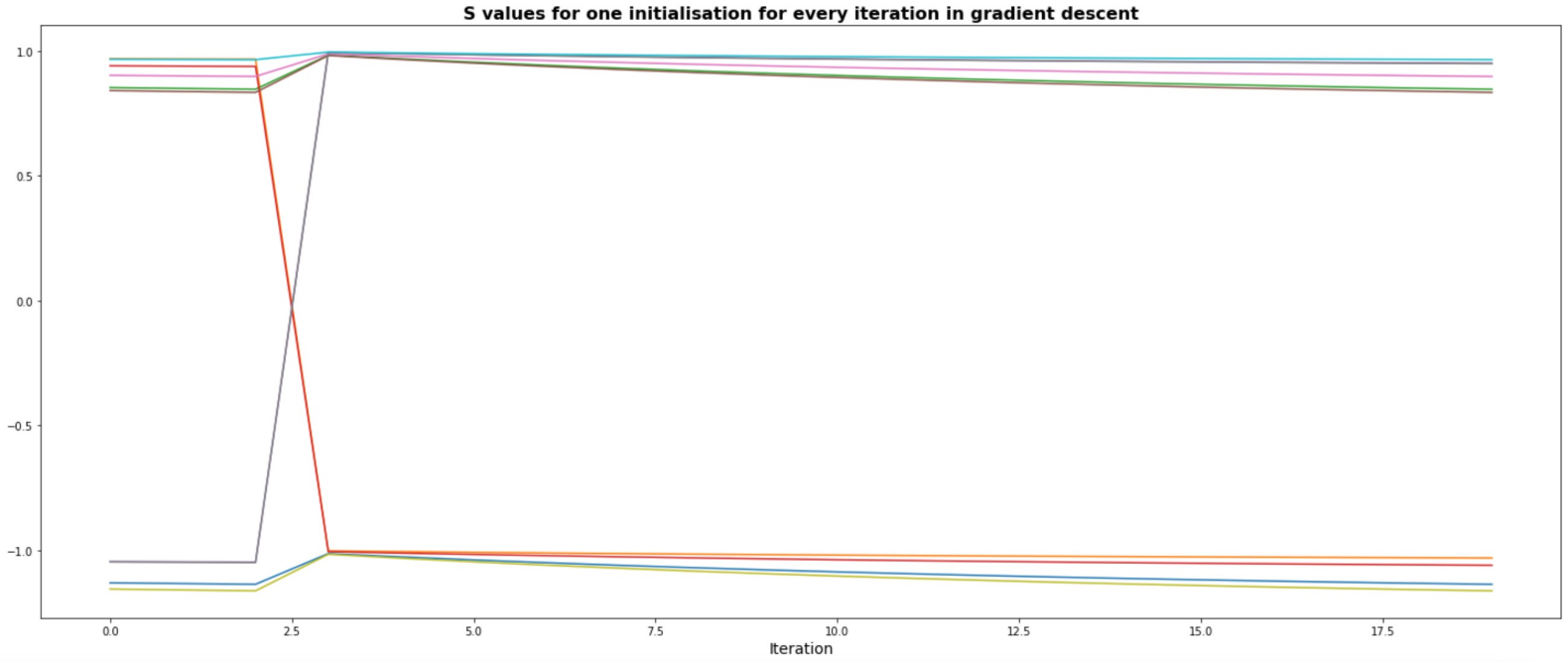
$$Oscillations$$

$$s_j = s_j - \lambda * 2 * n_j \sum_{j=1}^{N} n_j * s_j$$

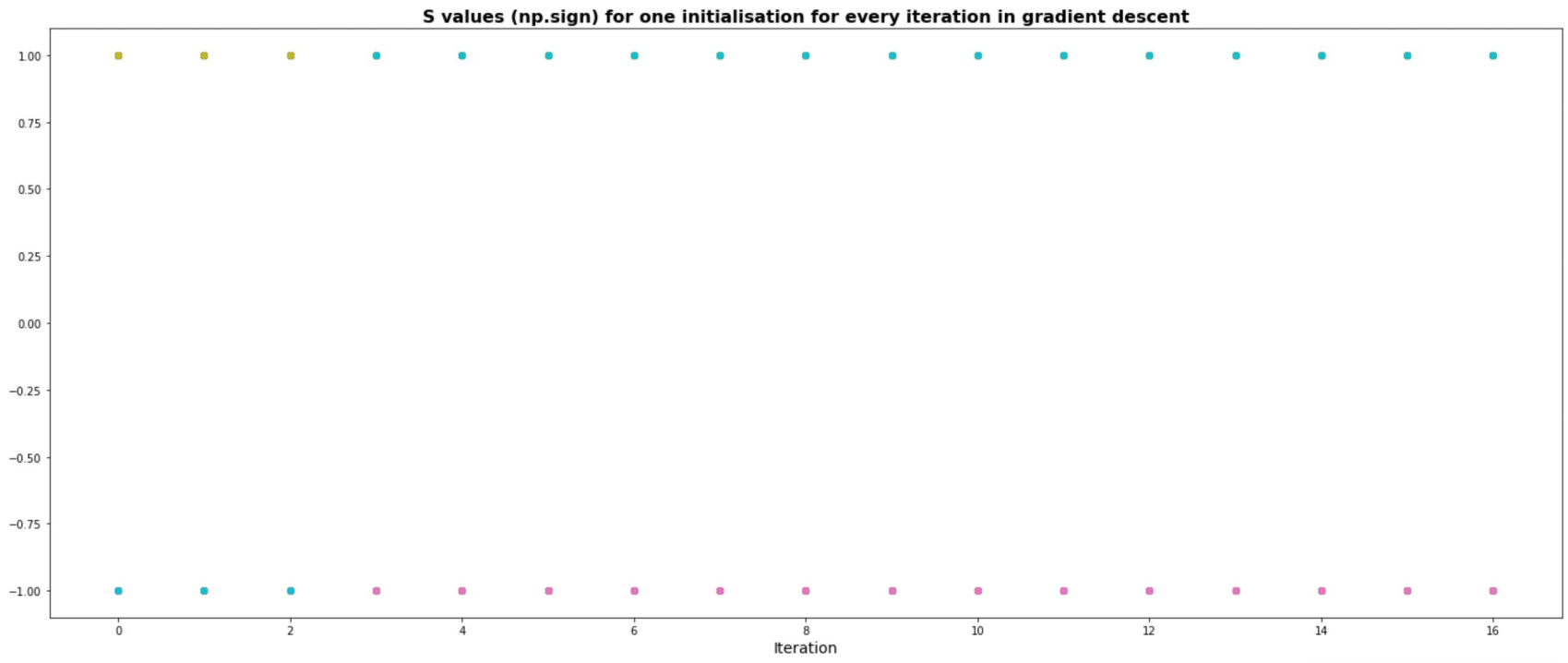


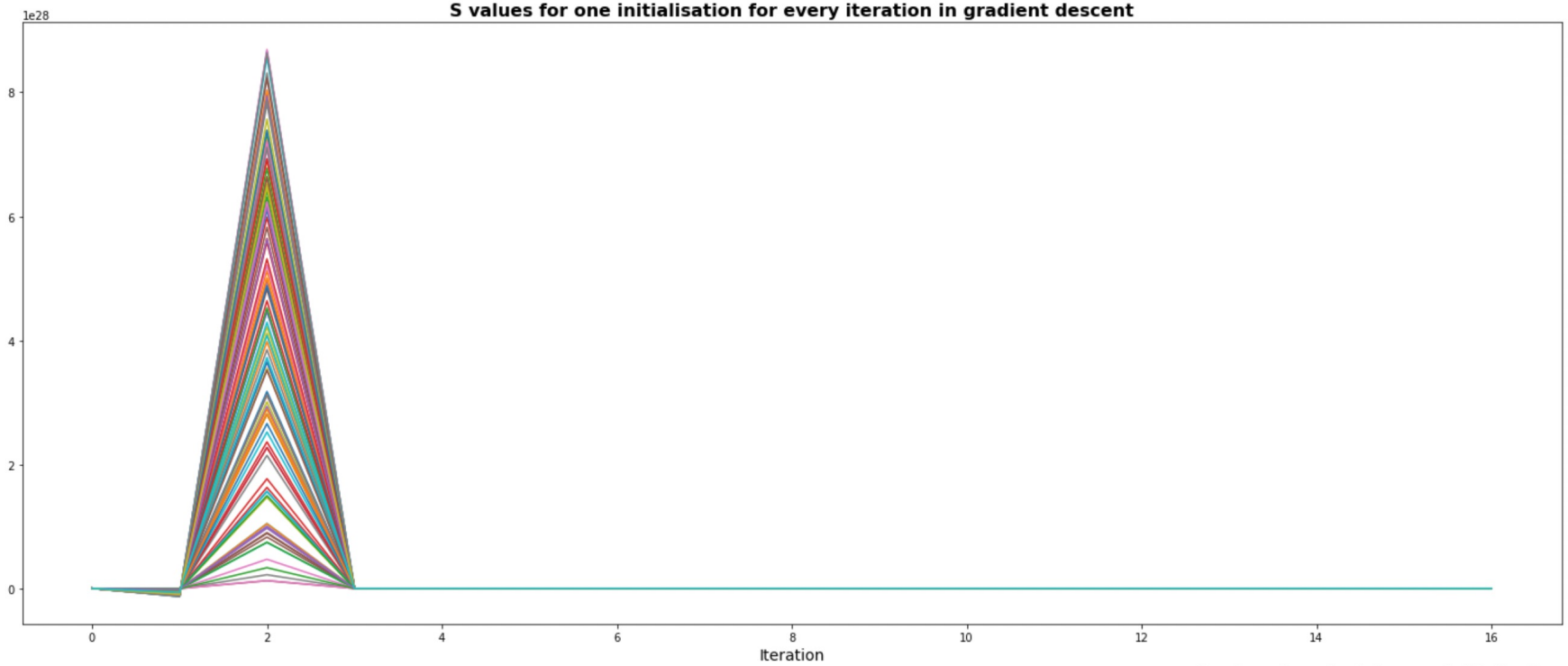
$$s_j = [-30, -122, -13, -22]$$

$$s_j = [-1, -1, -1, -1]$$

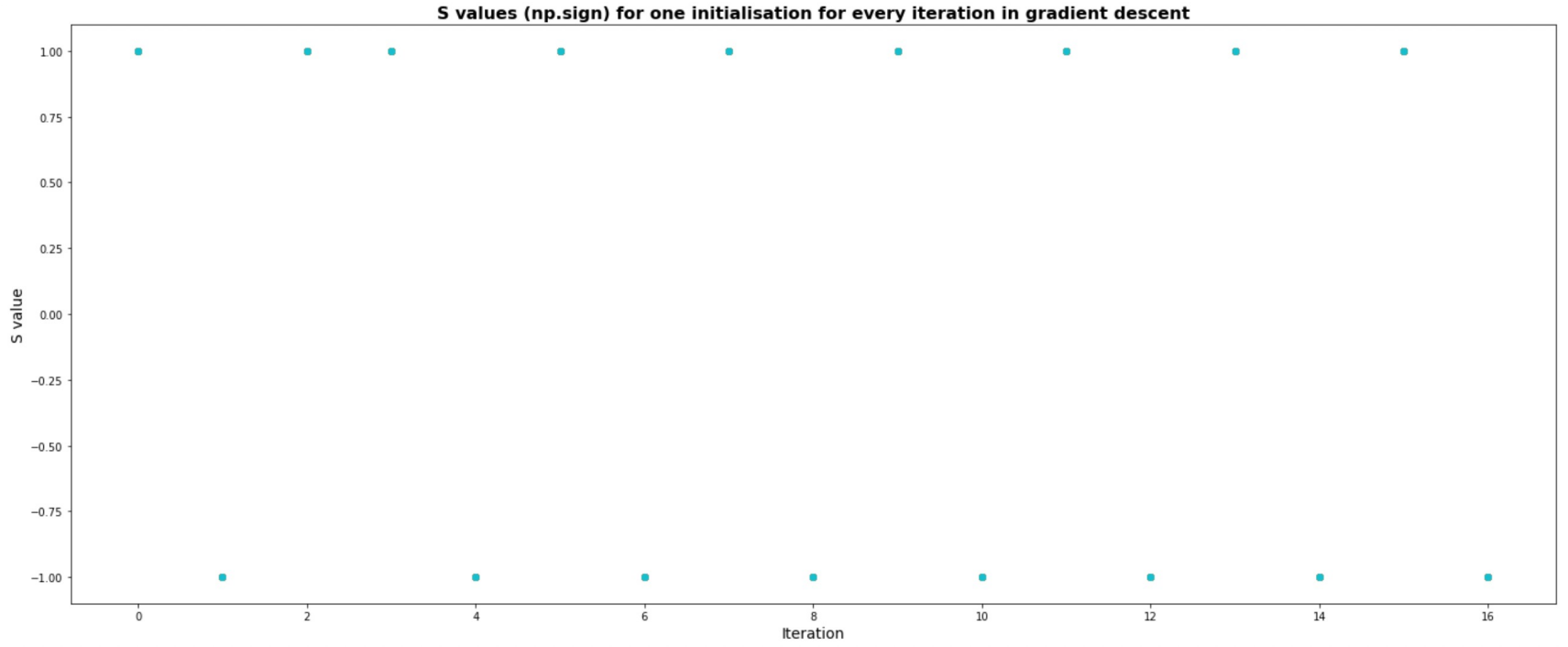



```
n = np.array(10*np.random.rand(10))
```



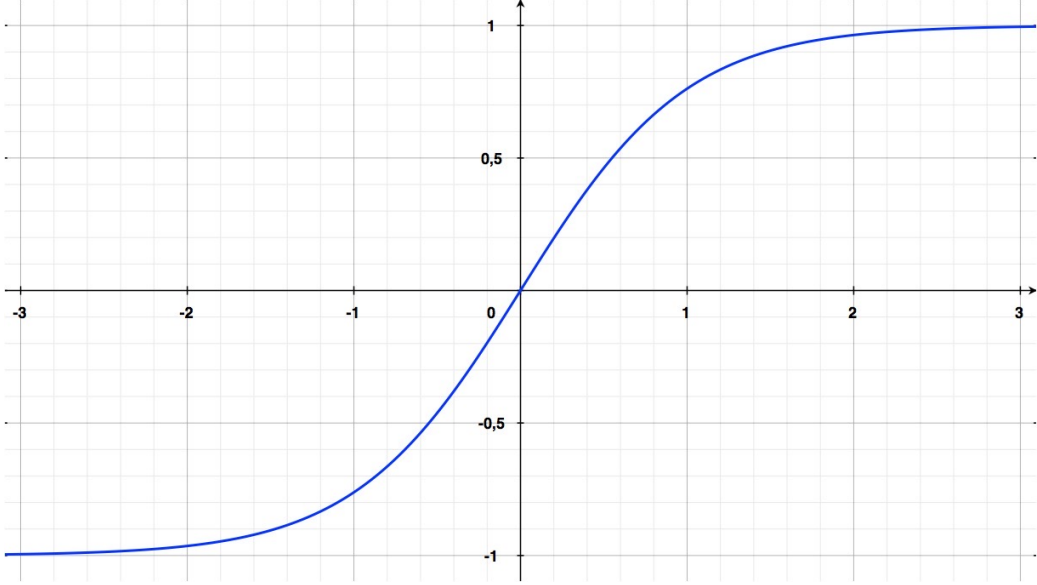


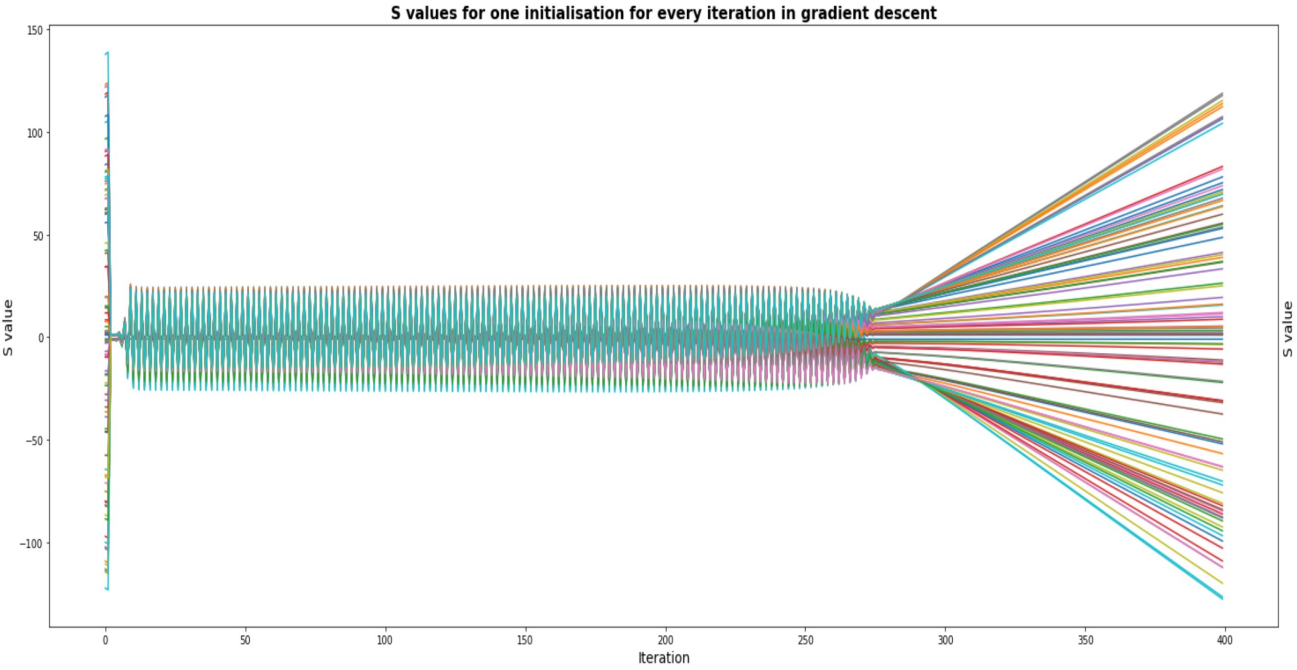

```
n = np.array(100*np.random.rand(100))
```

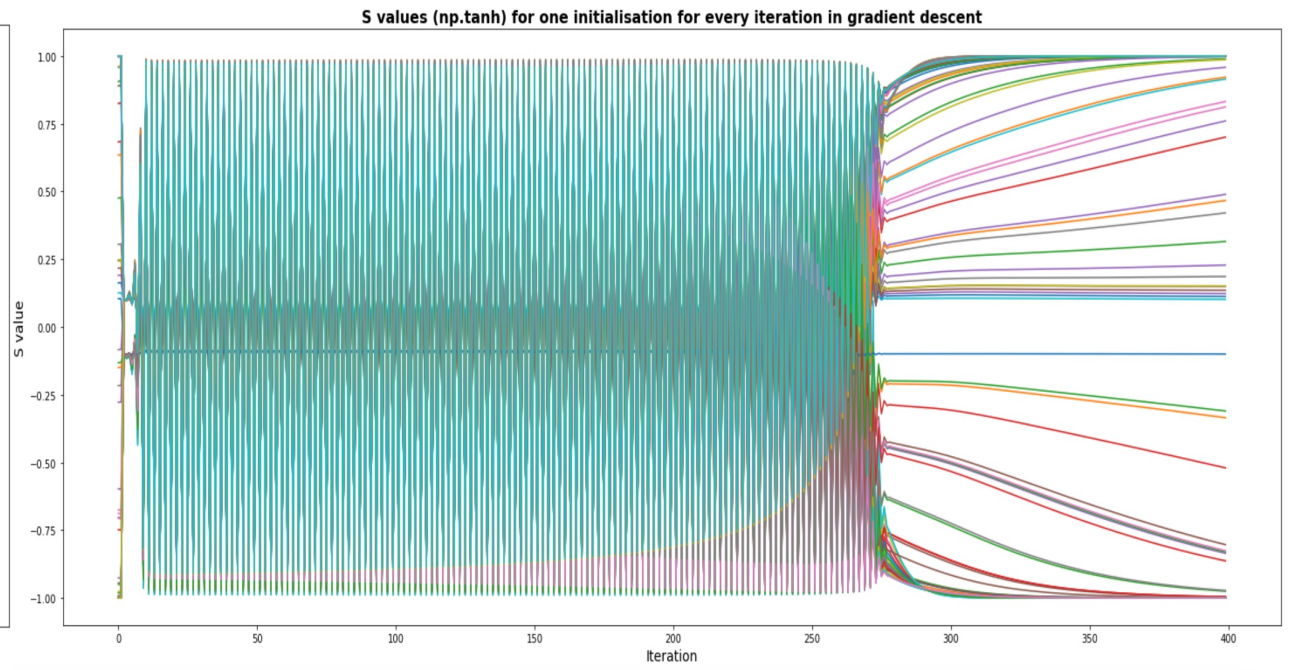


```
s=np.tanh(0.1*s_loop)

diff = (- np.dot(n,s) * n + n*s*n)
s_loop = s_loop + lamb*diff

s_loop = np.sign(s_loop))
```

$$F = \sum_{\substack{j=1 \\ i<j}}^{N} 2n_i n_j * s_i s_j + \sum_{i=1}^{N} (n_i s_i)^2$$







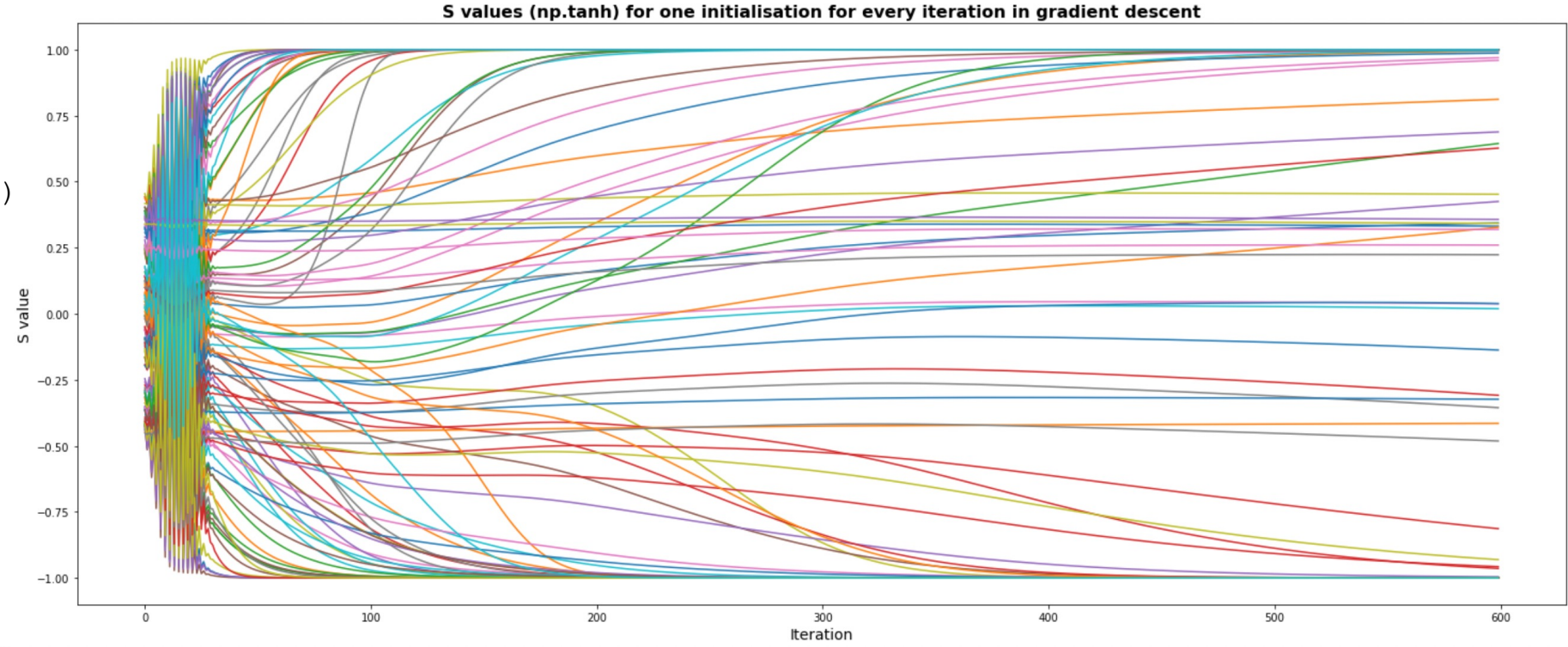
Hophield

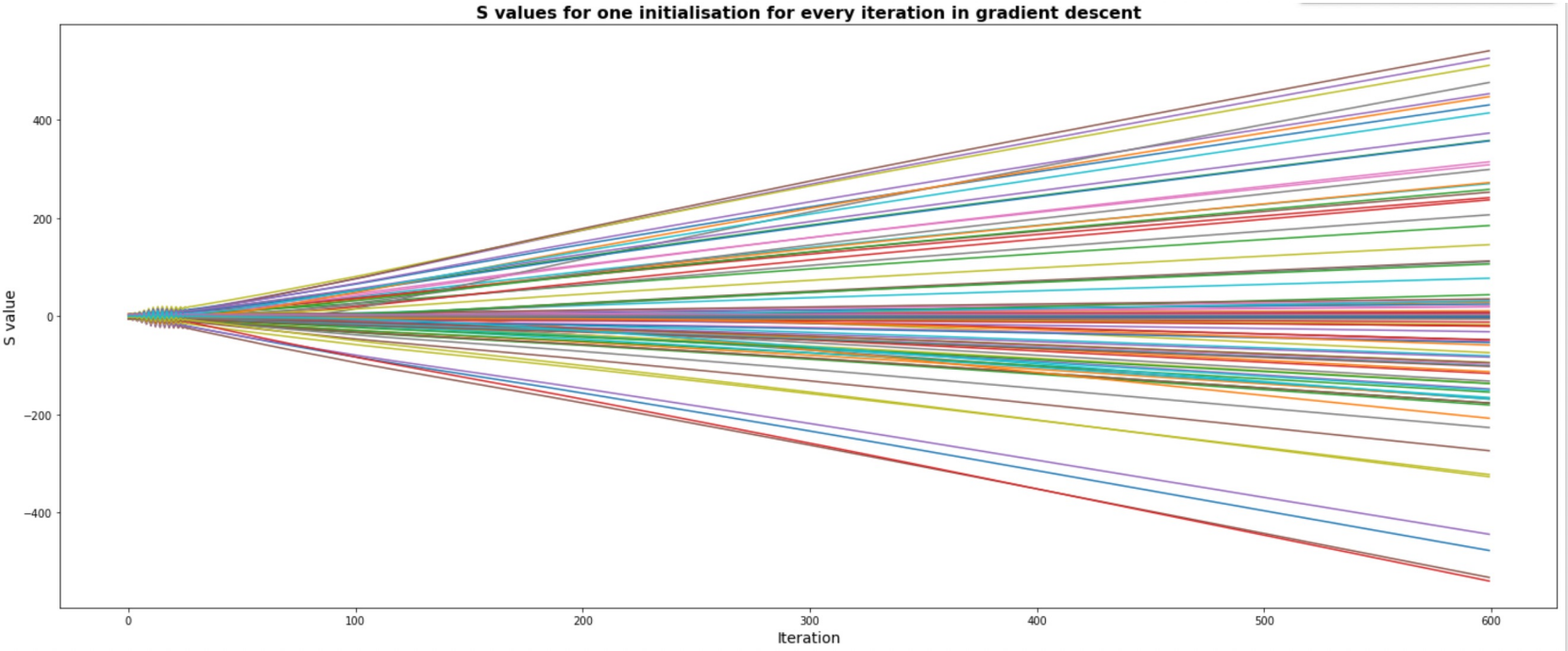
```
s=np.tanh(beta*s_loop)

diff = (-s_loop/tau - np.dot(n,s) * n + n*s*n)

s_loop = s_loop + lamb*diff
```

$$dx_j/dt = -x_j/\tau + \Sigma_k T_{jk} g(x_k) + I_j \,.$$

- We observe physical system with such behavior (exciton polariton).
- Previously we were considering one shot learning approach which describes the photon interaction with internal field, which is an ideal system.
- But in fact our system is usually affected by external conditions, which can be described using Hopfield approach.

```
n =
np.array(100*np.random.rand(100))
```

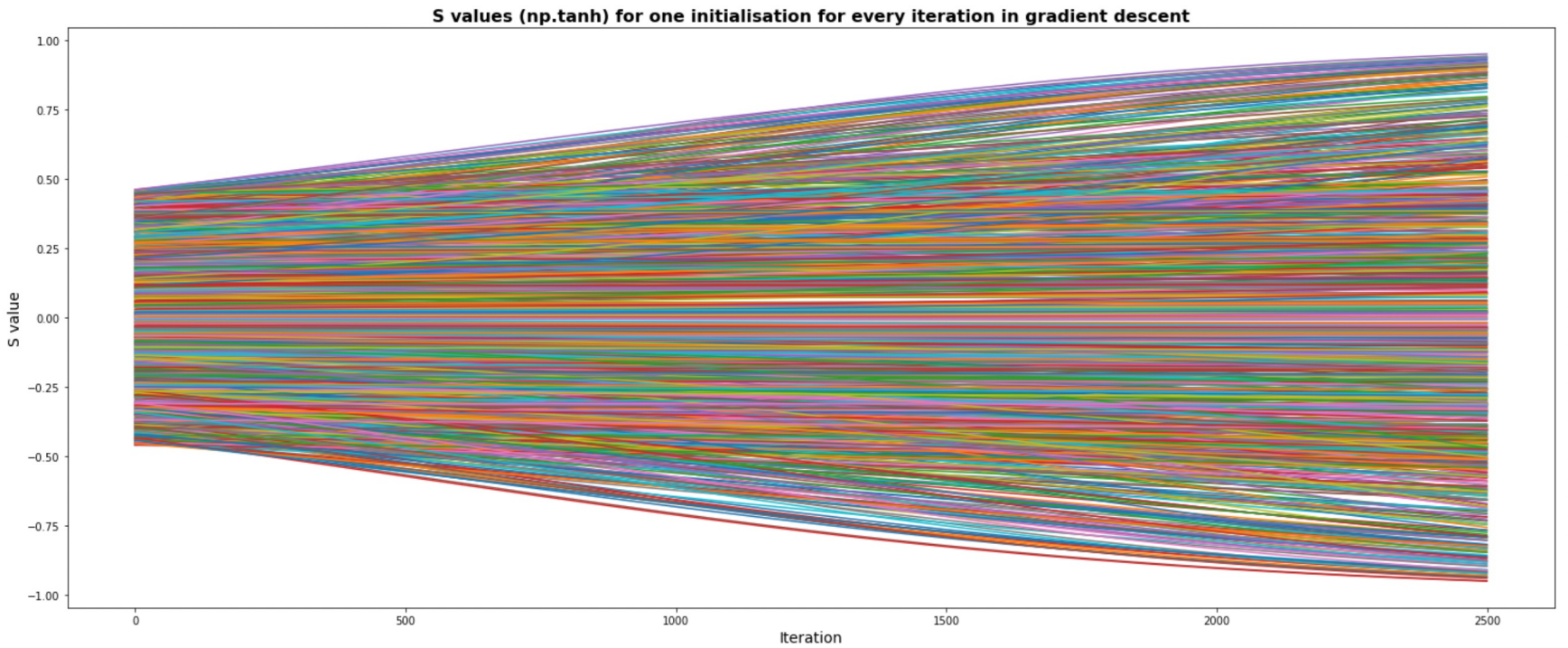


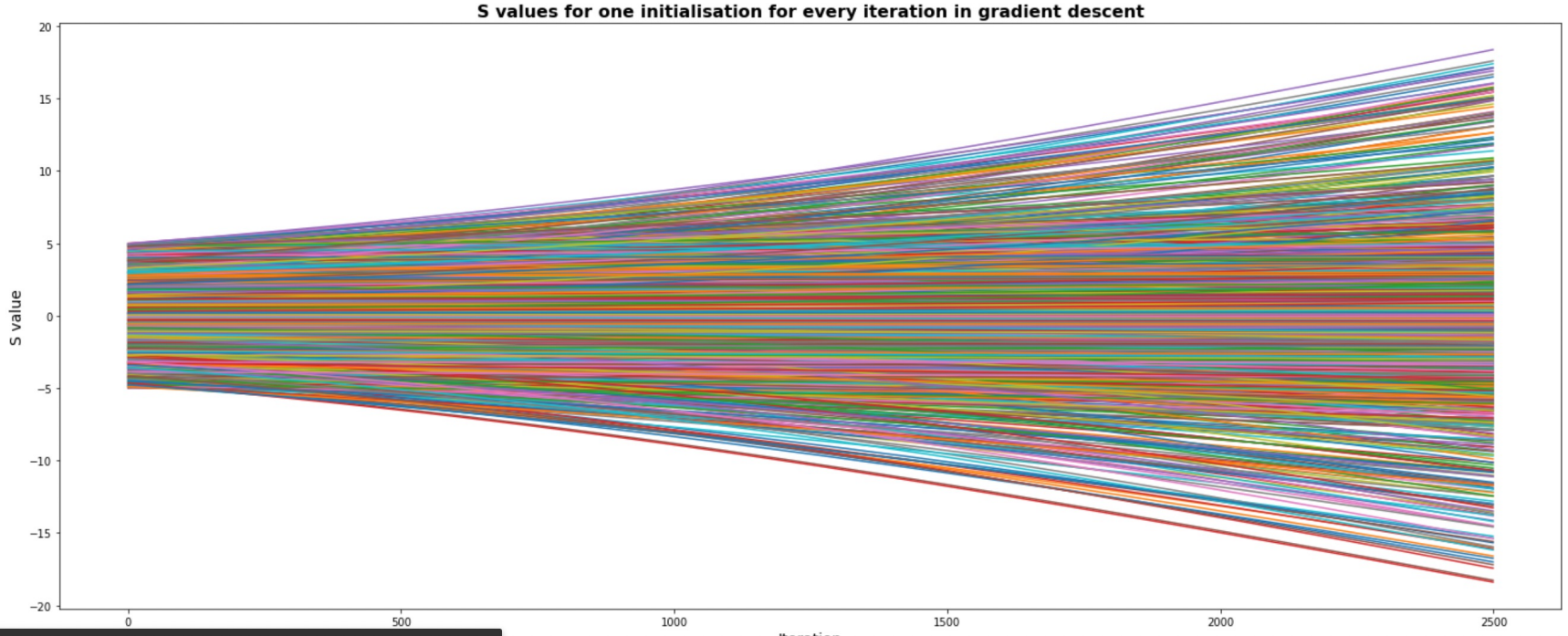



```
H(s) = 0.21407654184481772
Additional:

number of random
initializations: 19,

lambda = 0.0001

Calculation time:
0.968699216842651 4
```

The optimized parameters are:
lamb = 7.7e-07;
beta = 0.1;
tau = 100.0

```
n =
np.array(100*np.random.rand(1000))
```

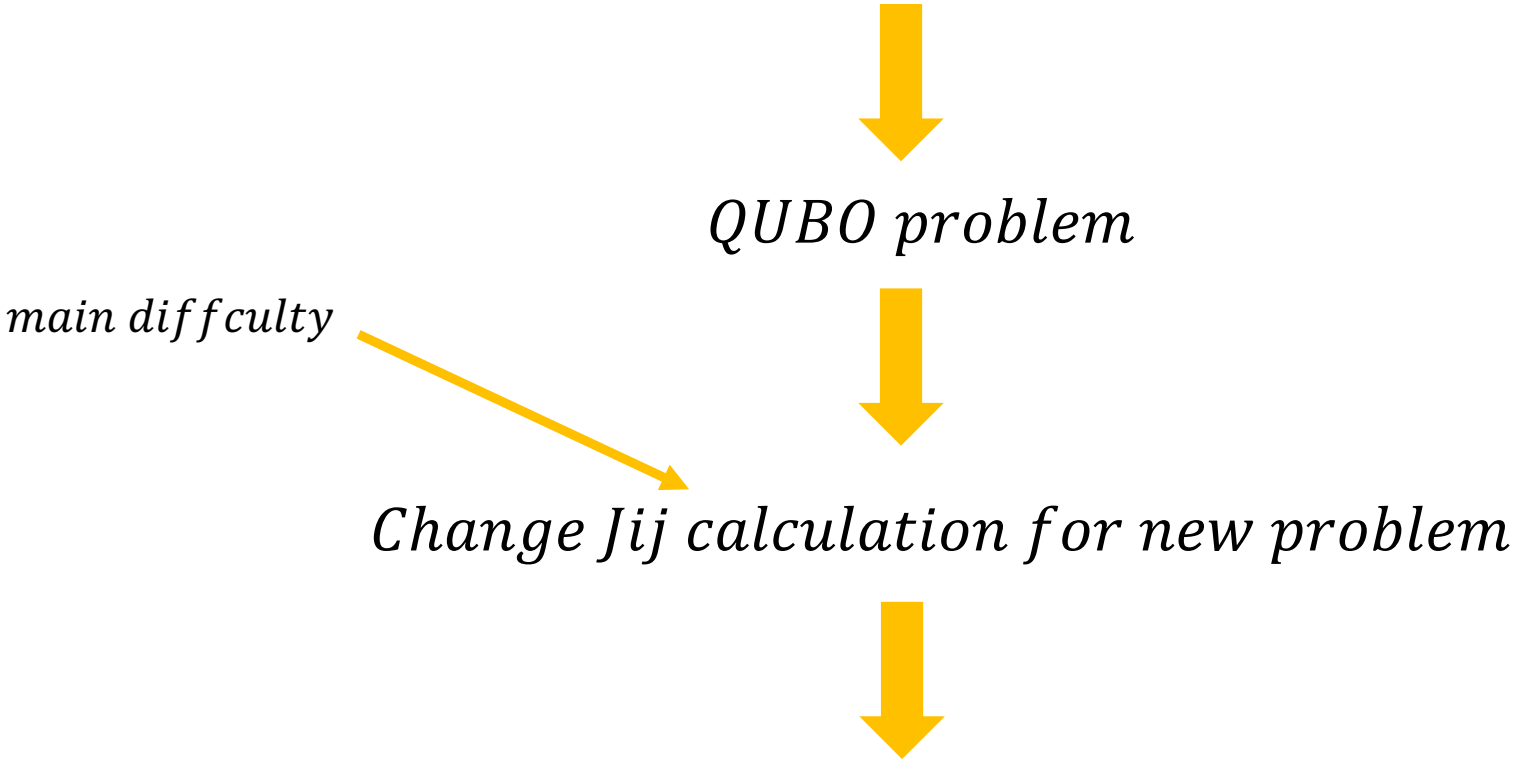




H(s) = 0.15902696559204882

number of random initializations: 79

```
def Jij_init(n):
return np.multiply(np.multiply(n,n.reshape(n.size,1)),np.ones((n.size,n.size))-np.eye(n.size))
```



Community detection

- Girvan-Newman algorithm
- Fluid Communities algorithm
- Label Propagation algorithm
- Clique Percolation algorithm
- Kernighan-Lin algorithm
- **Belief propagation**
- **Message passing**