Harmonious economic planning

Paul Cockshott

Algorithmic Complexity

Complexity defines how long an algorithm takes as function of problem size

Classes

1. Constant
2. Linear **O** N
3. Log linear **O**N log N
4. Polynomial **O**N^2 **O**N^3
5. Exponential **O**e^N

Economic planning must be tractable

If planning on a national scale is to work, the time taken to compute a plan must not be too great

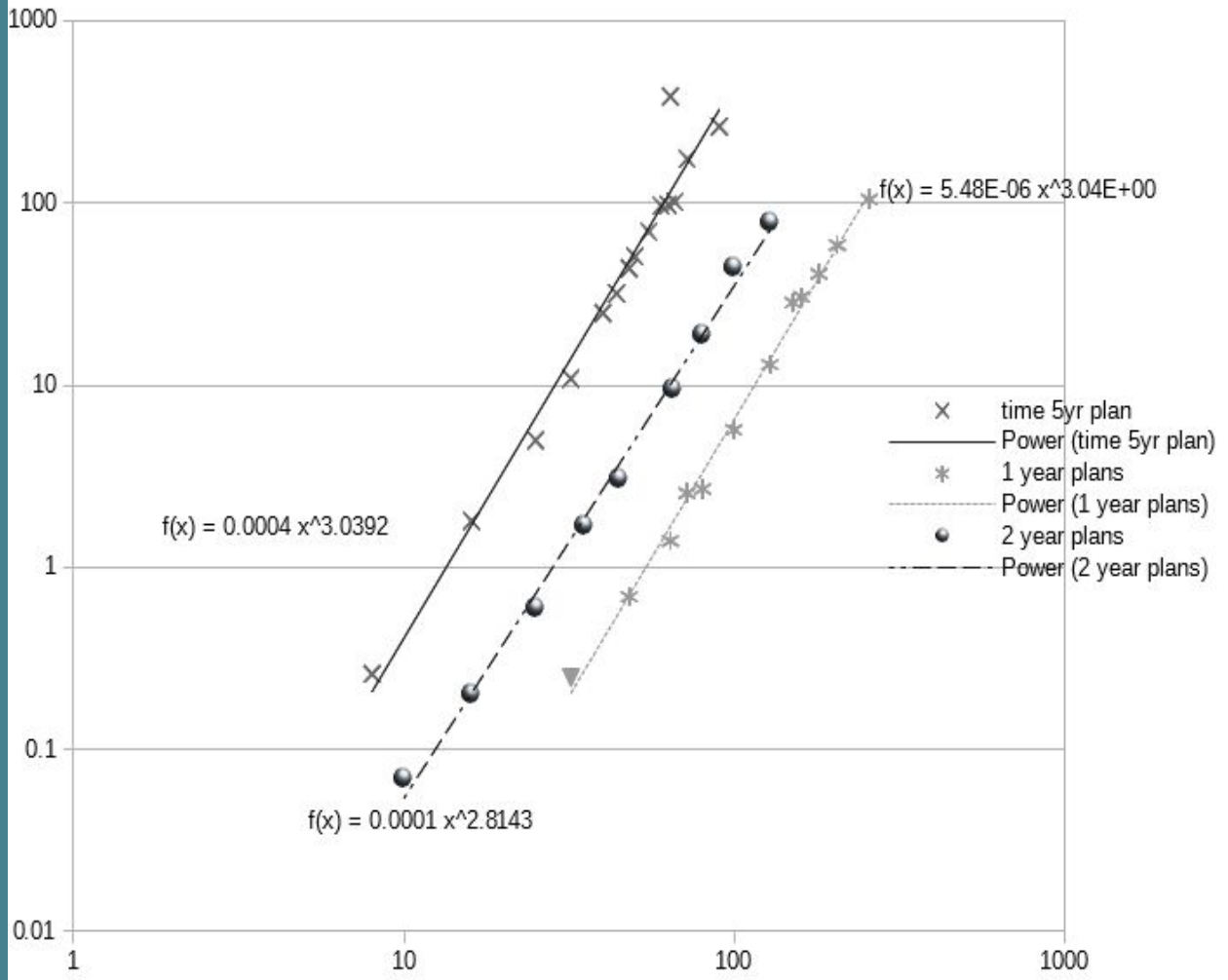
If you have 100 industries one method may work, but if you increase it to 1 million it may be prohibitively slow

What about the LP-solve package that I demonstrated earlier?

LP-solve is polynomial

Graph shows time to run economic plans with horizontal axis being number of industries, vertical axis time in seconds.

Graph is log log so straight line indicates a polynomial - roughly equivalent to N^3



This is too slow for big plans

Projected time to compute large plans with lp-solve

industries	1 year plan	5 year plan
500	0.24hr	17hr
5000	11 days	2.2 years
50000	33years	2417years

We need a near linear algorithm

In our book Towards a New Socialism we describe such a near linear algorithm.

I originally developed it in about 1989 but had long since lost the source code.

I have released a new version of it written in java on github at

Why a log linear solution must exist

Given that any explicit calculation or algorithm that humans can do, can also, in principle be done by computers. Given further, that the complexity order of an algorithm does not change depending on whether people do it by hand or a computer does it. It follows then that the existence of functioning market economies is proof that low complexity coordination algorithms exist.

It is clearly not the case that market economies depend on an algorithm of order N^3 or they would never have grown to be able to produce the hundreds of millions of products[#amazon] that actually are on sale.

Harmony alg is NlogN

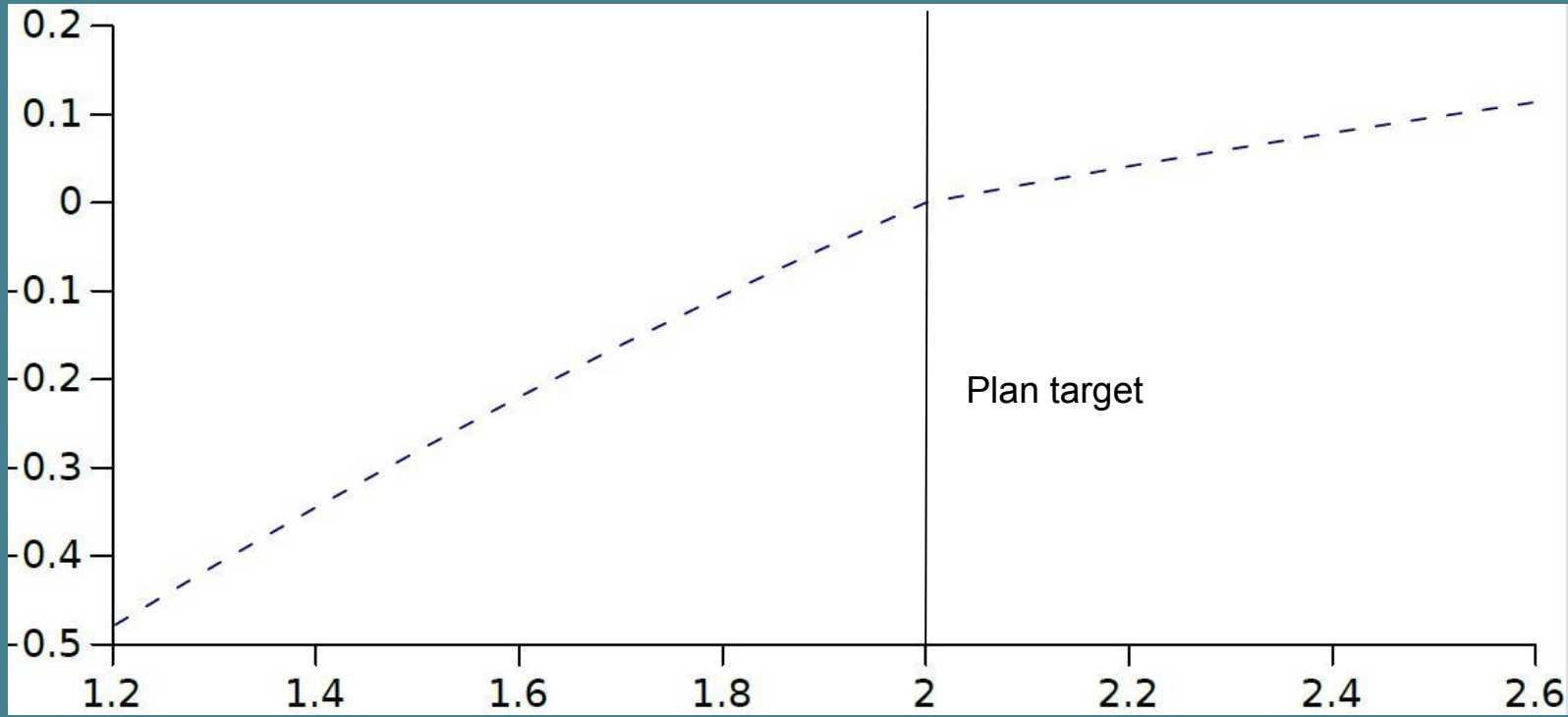
The Harmony Algorithm draws on ideas from marginalist economics and from neural nets to derive an iterative planning technique.

This has previously been shown to have N\log N complexity for single year plans.

Socialist utility function

The Harmony function is supposed to mimic the principle of positive but diminishing marginal social utility. What is required is a function whose value rises as plan fulfillment approaches but which rewards overfulfillment less than it punishes underfulfillment.

Harmony function



Actual function used

$$h(t,n) = \begin{cases} S - \frac{S^2}{2} & S < 0 \\ \ln(S+1) & S \geq 0 \end{cases}$$

Where t is the plan target, n the net output and S the Surplus of production given by

$$S = \frac{n-t}{t}$$

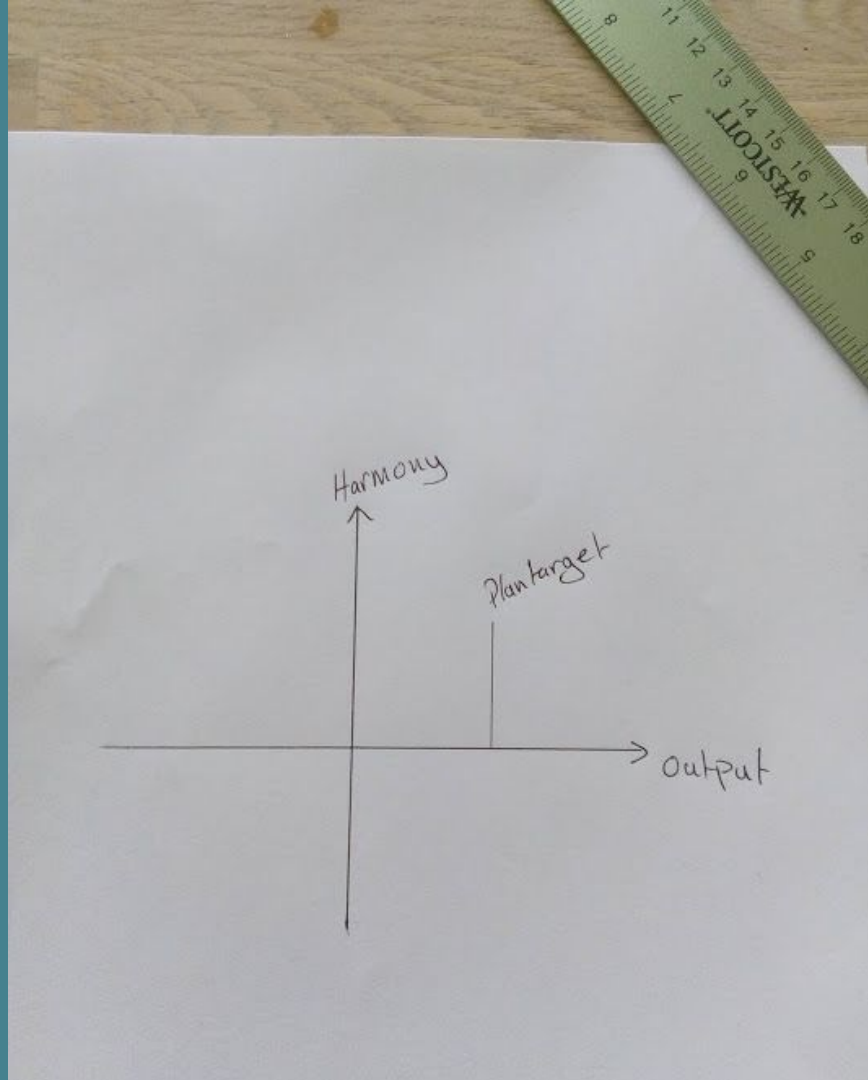
That is to say the proportional amount the plan has been exceeded by for each product.

Why use this

1. To give a more flexible plan target that rewards over fulfillment and punishes shortfalls
2. Because the function has a continuous first derivative it is allows the use of Newton's method to approximate functions. This allows us to rapidly bring all industries into approximate alignment with the plan target.

Newton's approximation method

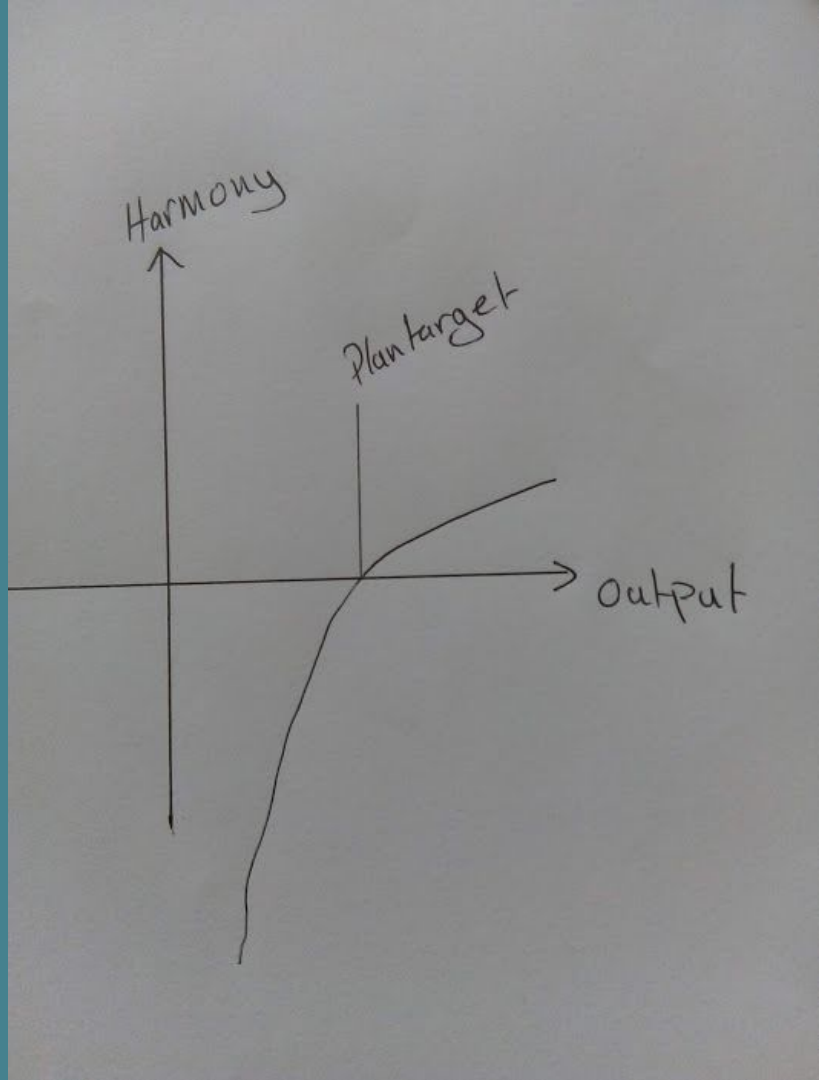
Suppose we start out with this scheme



Newton's approximation method

Suppose we start out with this scheme

We put in the harmony function, which is 0 at the plan target.



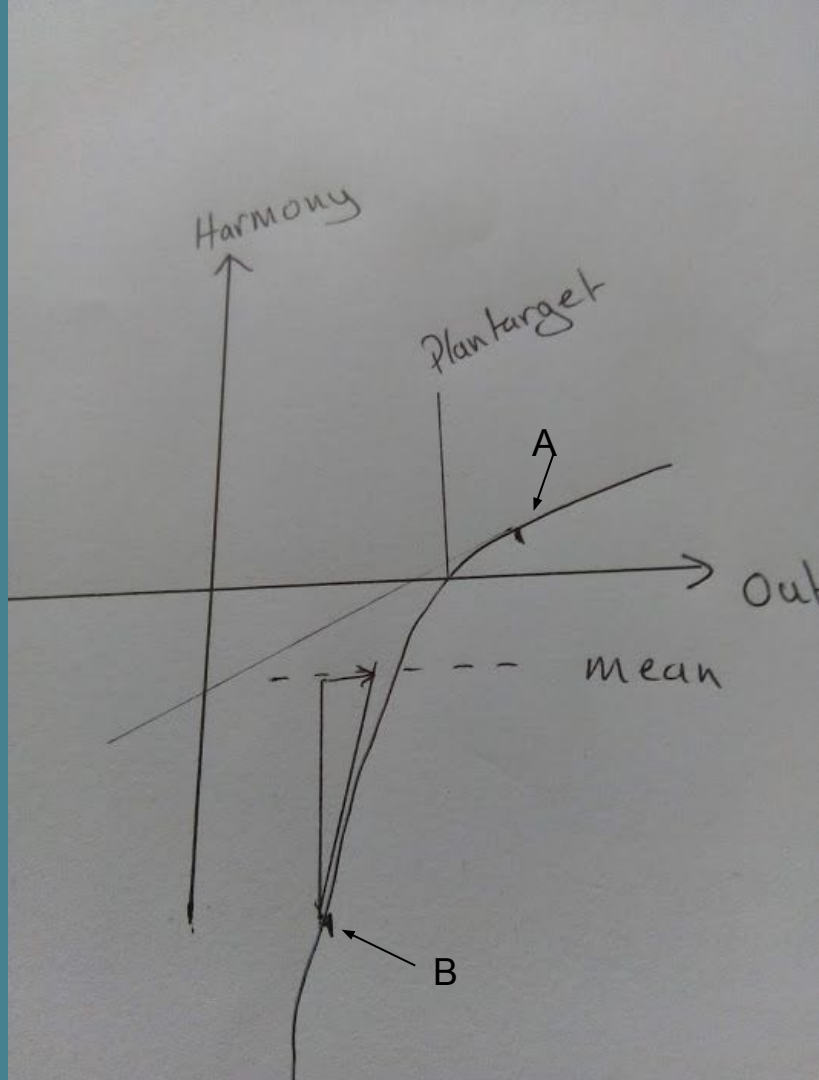
Newton's approximation method

Suppose we start out with this scheme

We put in the harmony function, which is 0 at the plan target.

We have two industries A exceeds plan and B falls short, we want to shrink A and increase B

Since we know the gradients of the function we can draw or compute lines that intersect with the mean harmony of the whole economy



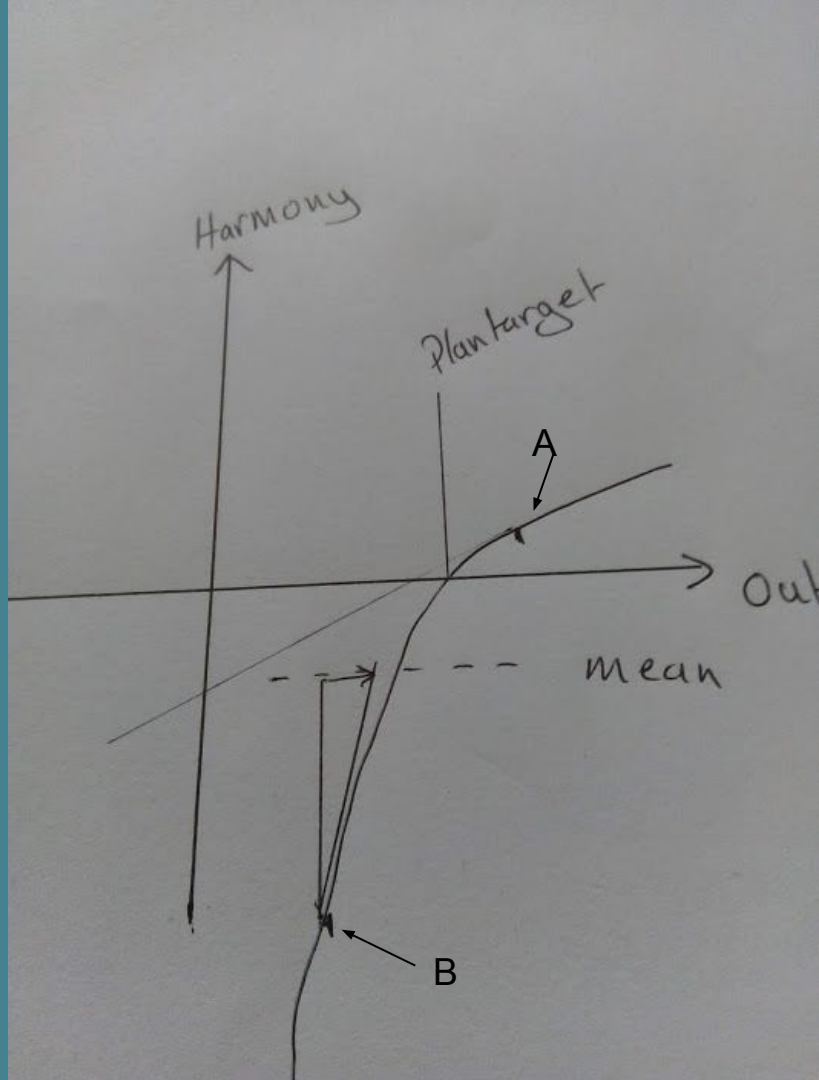
Newton's approximation method

Suppose we start out with this scheme

We put in the harmony function, which is 0 at the plan target.

We have two industries A exceeds plan and B falls short, we want to shrink A and increase B

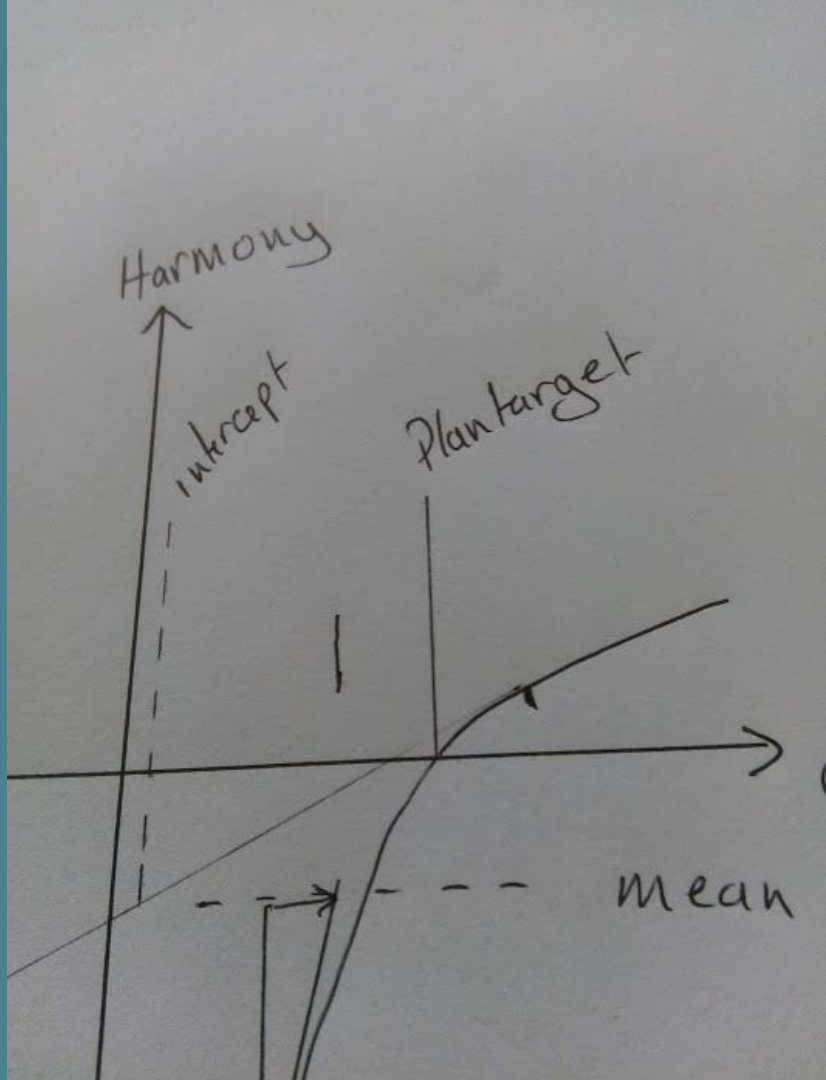
Since we know the gradients of the function we can draw or compute lines that intersect with the mean harmony of the whole economy



Newton's approximation method

Since we know the gradients of the function we can draw or compute lines that intersect with the mean harmony of the whole economy.

But these 'overshoot', they reduce A too much.

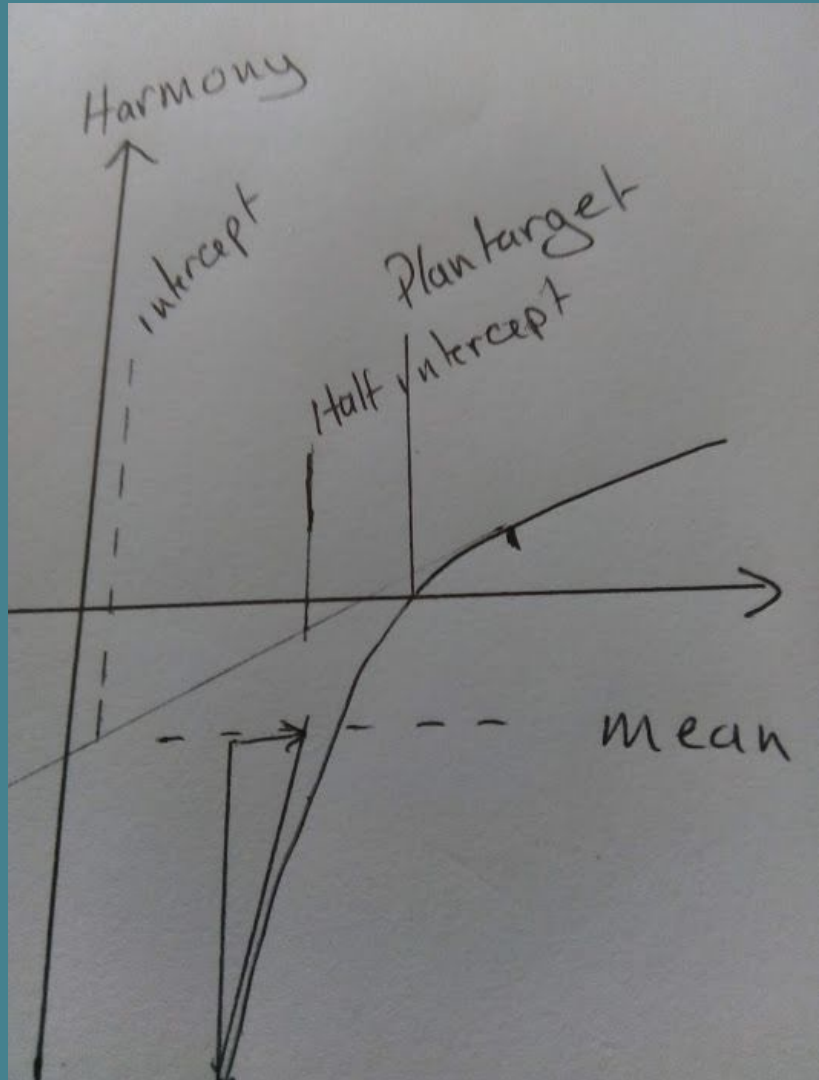


Newton's approximation method

Since we know the gradients of the function we can draw or compute lines that intersect with the mean harmony of the whole economy.

But these 'overshoot', they reduce A too much.

But what if we look at half the shrinkage of A

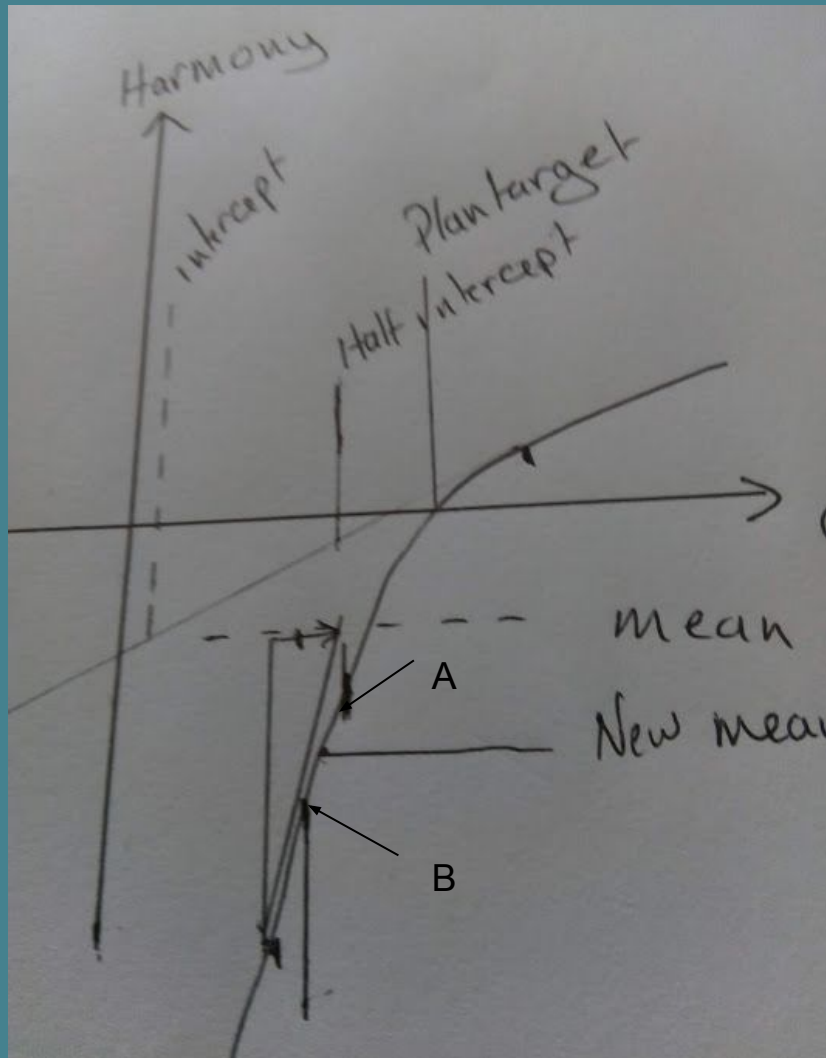


Newton's approximation method

But these 'overshoot', they reduce A too much.

But what if we look at half the shrinkage of A, half the increase of B

We move them there and get a new mean



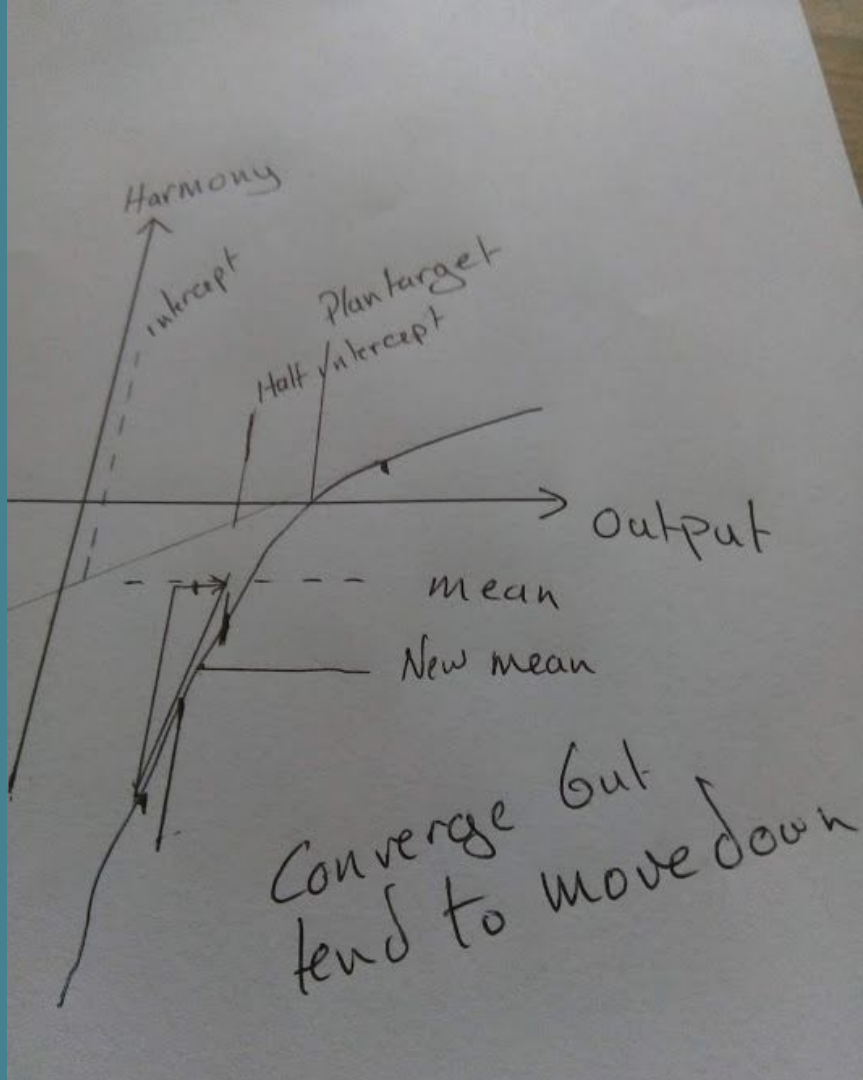
Newton's approximation method

But these 'overshoot', they reduce A too much.

But what if we look at half the shrinkage of A, half the increase of B

We move them there and get a new mean,

Converge on the mean but it shifts down. We need another stage of the algorithm to slide the mean up along the harmony function



Use harmony gain rate as dummy profit rate

In a capitalist economy resources are supposed to shift towards sectors with a higher rate of profit.

In the planning analog, we have to shift free resources to where the marginal harmony gain rate is highest.

Essentially we use the derivative of the harmony function of each product as a shadow price and compute a rate of return for each technique.

Use harmony gain rate as dummy profit rate

We expand each industry in proportion to its harmony gain rate.

G_i is the gain rate of industry i

This can lead to instabilities if industries have negative harmony gain - could lead to industries being set to negative rates of production which are impossible.

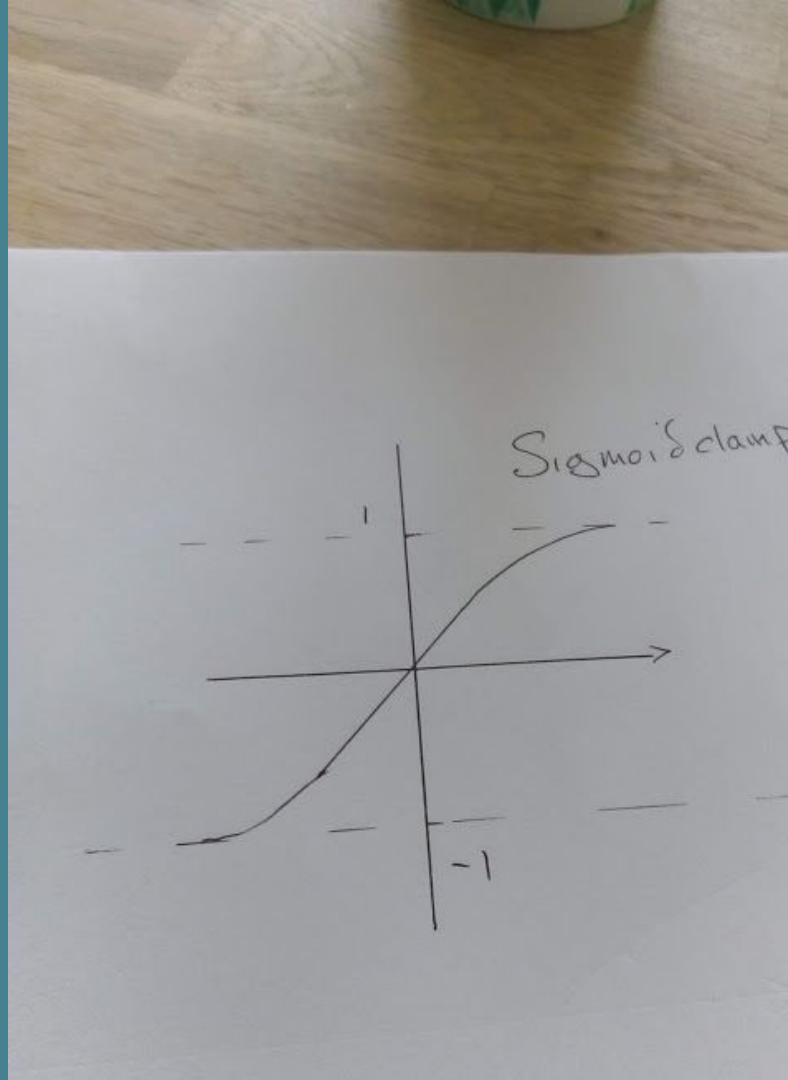
Use neural net technique

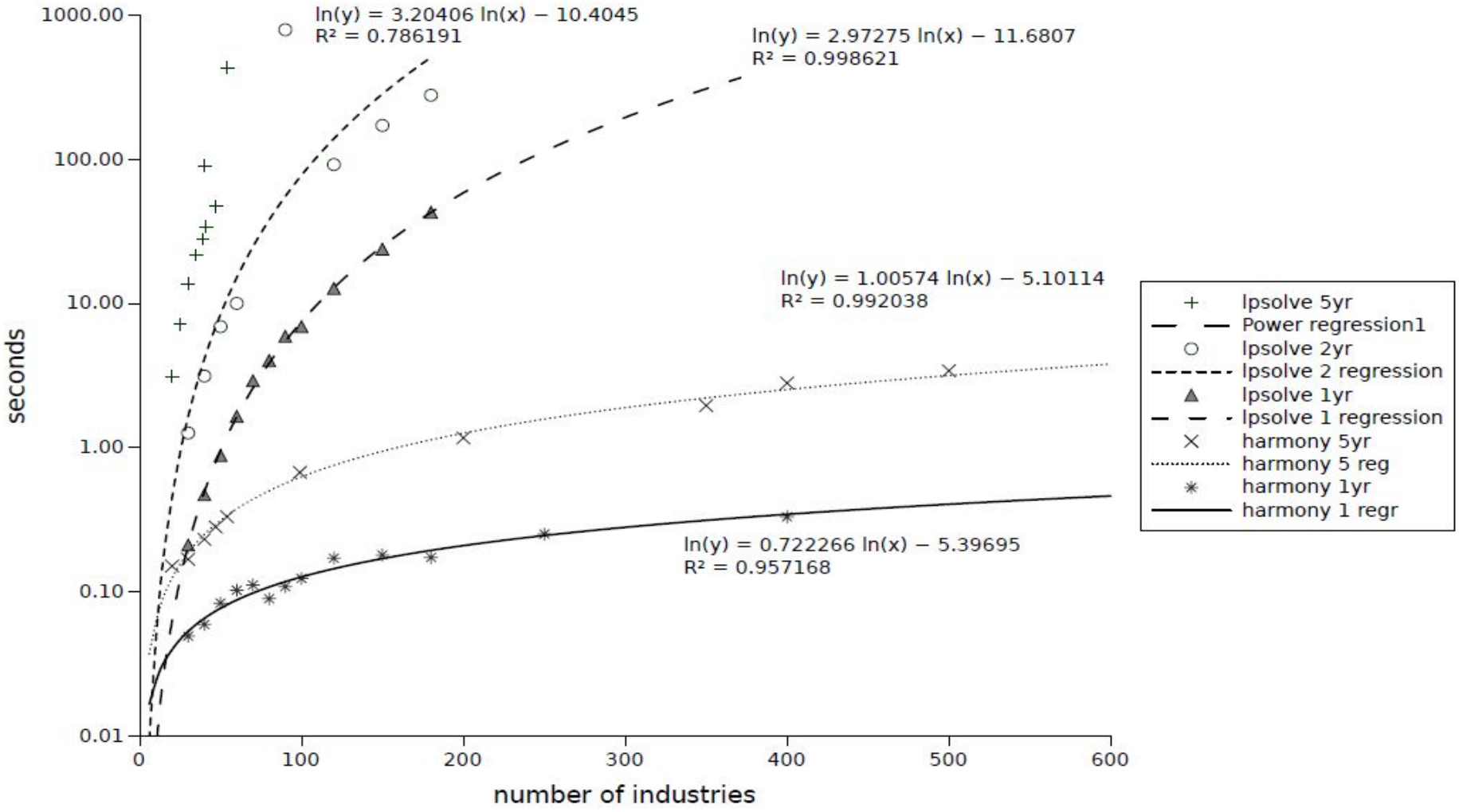
We use a sigmoid function as commonly used in neural nets, to clamp the rate of harmony gain to be in the range -1 to +1.

We then adjust the intensity I_j of production of each industry j thus:

$$I_j = I_j \times (1 + \sigma(G_j)\phi\psi)$$

Where phi and psi are tuning constants <1




ln(y) = 3.20406 ln(x) − 10.4045
R² = 0.786191
ln(y) = 2.97275 ln(x) − 11.6807
R² = 0.998621
ln(y) = 1.00574 ln(x) − 5.10114
R² = 0.992038
ln(y) = 0.722266 ln(x) − 5.39695
R² = 0.957168
1000.00
100.00
10.00
1.00
0.10
0.01
seconds
0
100
200
300
400
500
600
number of industries
lpsolve 5yr
Power regression1
lpsolve 2yr
lpsolve 2 regression
lpsolve 1yr
lpsolve 1 regression
harmony 5yr
harmony 5 reg
harmony 1yr
harmony 1 regr

Implication

This indicates that were only two CPU cores of the same power as used earlier, a 50,000 industry 5 year plan could be optimised in under ten minutes, which compares favourably with the 2400 years it would take using a linear programming system.

Restriction

The package currently reads in data to specify the problem in IO table format which is very inefficient.

A 50,000 sector IO table would occupy a prohibitive amount of disk space. If you give it an IO table that is too big, you get a java heap overflow.

As a viewer pointed out, for large plans you would have to supply the data in relational database form.

Extensions

You would also have to parallelise the algorithm if you want to handle millions of products.

But the structure of the algorithm lends itself to massive data parallelism.

Producing a production quality package for detailed national plans is not something that it is worth my while doing now as an amateur. But it would be easy for software teams at super-computer centers to do.