Socialist calculation and algorithms

Paul Cockshott

Historical Background

- Immediate - the work of Prof Nove of my university and its impact in Britain
- Long term - the work of the Austrian capital theorists, particularly von Mises and Hayek
- Current relevance - application of Hayekian economics to formerly planned economies
  - Collapse of of production
  - Drastic fall in living standards and life expectancy

Plans and computers

- Starting with Von Mises, conservative economists argued that effective socialist planning was impossible because:
- No – effective cost metric in absence of market
- Complexity too great – millions of equations argument.

Paul Cockshott

No Cost Metric

- Von Mises argued that without a market one could not cost things and thus had no rational basis for deciding between production alternatives.
- One exception he allowed was the use of Labour Values – we will return to this

Millions of equations

- Computers obviously change this as they can solve millions of equations
- Need to be quite precise about how many million equations and just how hard they are to solve
- This is a branch of complexity theory

Complexity

- The complexity of an algorithm is measured by the number of instructions used to compute it as the size of a problem grows.
- We will look at a simple example before going on to economic planning

Searching

- Suppose that I have a telephone directory for Brazilia and a phone number.
- It is clearly possible in principle to look at every number in the directory until I find who the number belonged to.
- The task would probably take several days.

Indexing and sorting

- If I have a name on the other hand, I can probably look up the phone number in less than 60 seconds.
- The complexity of looking up a name from a number is of order *n*, or **O***n*, for a directory with *n* names in it.
- The complexity of looking up a number from a name is of order Log(n) in a sorted directory

Example

- Suppose I have 2 directories

1. Has 1000 entries
2. Has 1,000,000 entries

To look up a name will take 1000 times as long in the second directory, but to look up a number – given the name will only take twice as long.

I/O table

	rubber	steel	oil	zinc	cotton
rubber					
steel					
oil					
zinc					
cotton					
labour					
outputs					

Paul Cockshott

Use of I/O table

- From the I/O table one can compute how much of each intermediate product required to produce each final product.
- In particular we can compute the labour content of each output.

Computability of labour content

- Suppose we have 10,000,000 different types of goods produced in an economy (Nove quotes this)
- Labour content given by the equation
  - $\lambda = A\lambda + l$
- Where λ is a vector of labour contents, l a vector of direct labour inputs and A an input output matrix
- Clearly too big to invert, matrix is even too big to store in a computer containing : 10^{14} cells.

Gaussian solution impossible

products	multiplications	Seconds taken	
		uniprocessor	multiprocessor
1000	1,000,000,000	10	0.1
100,000	10^{15}	10^{7}	100,000
10,000,000	10^{21}	10^{13}	10^{11}sec=3000yrs

Simplification

- Matrix is sparse, most elements are zero
- Replace by linked list representation, we estimate the number of inputs directly used in a product is logarithmic in the size of the economy.
- Solve iteratively - use about 10 iterations,
- Complexity of order *n*Log*n* in number of products. We estimate that it takes a few minutes on a modern machine.

Sparse representation

- Each production process represented by a list of pairs (input code, quantity)
- On average a process can then be represented in about 100 cells instead of 10,000,000

Iterative solution

- We only need to know labour values to about 3 significant figures.
- Initially just include direct labour inputs.
- The produce second estimate taking into account indirect inputs. Repeat this step about 10 times.
- You end up with a figure accurate to about 3 digits.

Iterative solution feasible

products	multiplications	Seconds taken	
		uniprocessor	100 core multiprocessor
1000	150,000	0.0016	0.000016
100,000	100,000,000	1	0.01
10,000,000	$6x10^{10}$	600	6

At this point illustrate the lp solve planner

Key developments in productive forces since 1960s

1. Internet
2. Giant databases
3. Super computers
4. Electronic payment cards

1. Internet allows real-time cybernetic planning and can solve the problem of dispersed information - Hayeks key objection
2. Big data allows concentration of the information needed for planning.
3. Super computers can solve the millions of equations in seconds - von Mises objection
4. Electronic payment cards allow replacement of cash with non transferable labour credits.

Essentials of cyber communism

Direct democracy

Major strategic decisions taken democratically

- How much labour to devote to education
- How much to health, pensions, sick
- How much to environmental protection
- How much to national defence
- How much to new investment

All this can be done by direct voting using computers or mobile phones every year.

Equivalence Economy

Marx's principle that non-public goods are distributed on the equivalence principle - you get back in goods the same amount of labour - after tax - that you perform.

Hence goods are priced in labour hours.

Cybernetic feedback from sales to the plan to adjust output to consumer needs.

Feedback mechanism

- We assume a real time feedback mechanism which uses sales of products along with democratically determined general goals to set net output targets for all goods.
- The planning computers must derive the gross outputs required to meed these net outputs.

Model we propose

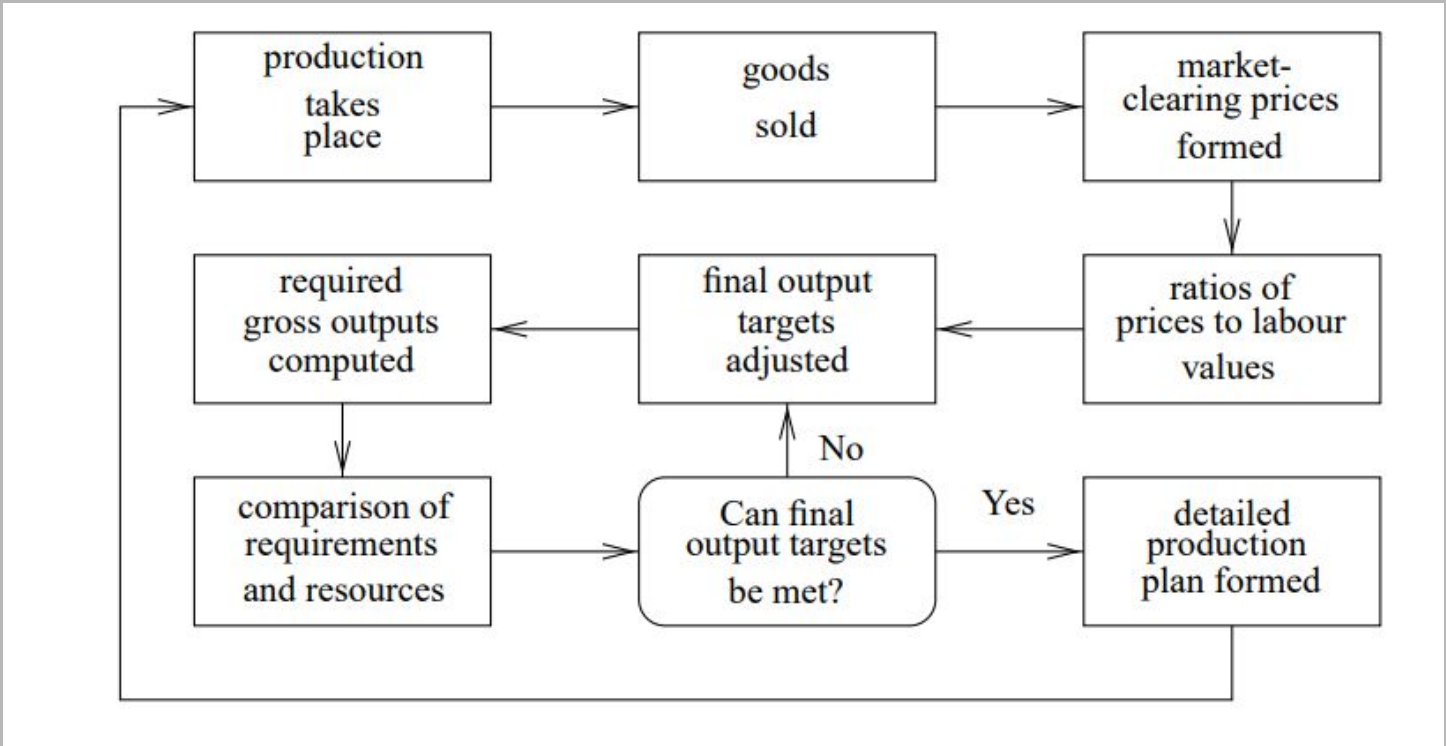
Drawn on the principles of Robert Owen, the founder of New Lanark

- Industry publicly owned and planned in physical units.
- Employees paid in labour tokens, 1 per hour.
- Goods priced in labour tokens proportional to the labour required to make them. (some discounting possible)

Market clearing prices used for finished goods

- If stocks of unsold goods grow – then reduce selling price
- If stocks fall – then increase selling price
- If price above labour value - then increase output
- If price below labour value – then reduce output

Overall cybernetic structure



Harmony planning

- Algorithm again feasible in log-linear time.
- Based on iterative adjustment of allocation of stocks between different production activities guided by the derivative of the harmony function for each industry.

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

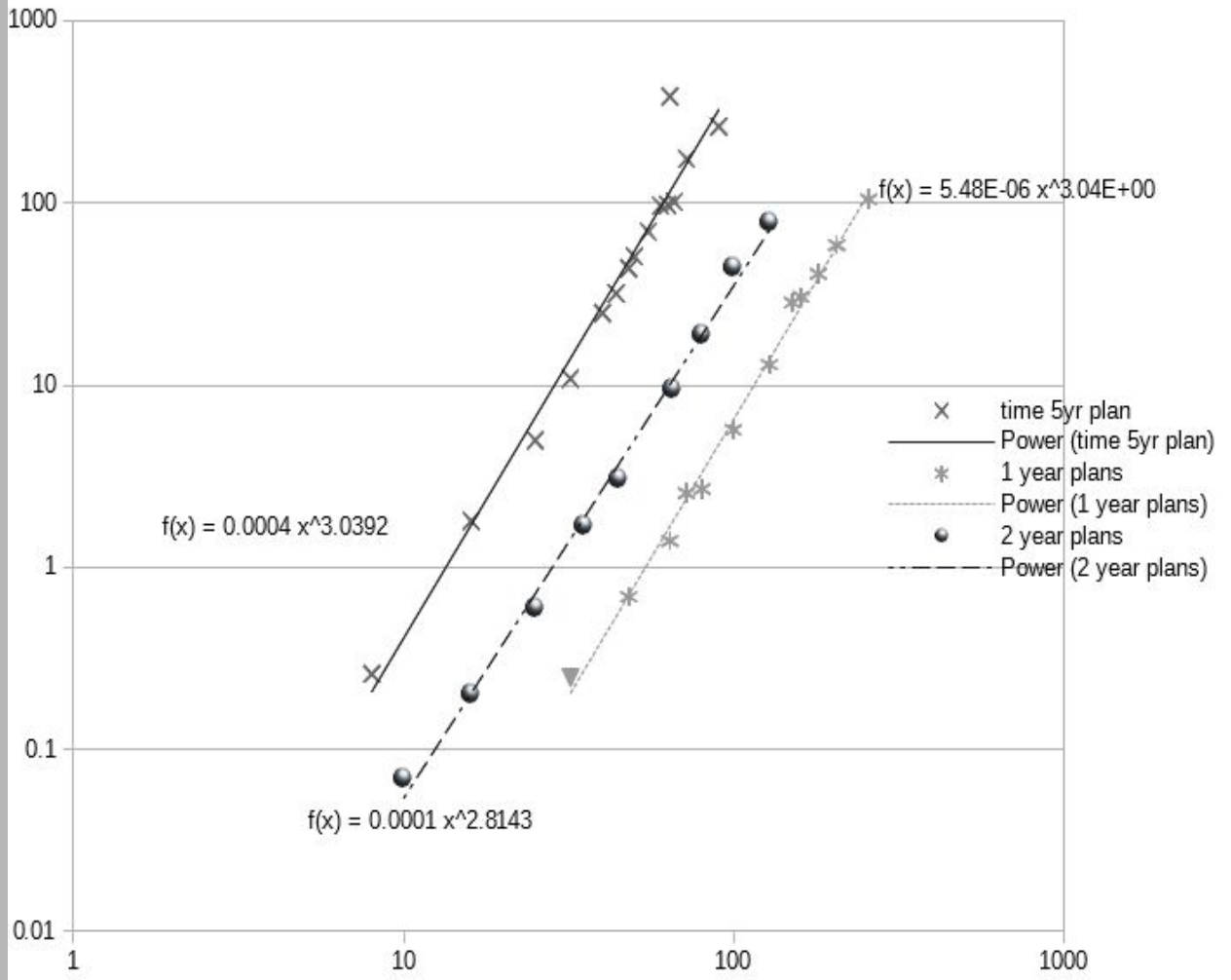
If you have 100 industries one method may work, but if you increase it to 1 million it may be prohibitively slow

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

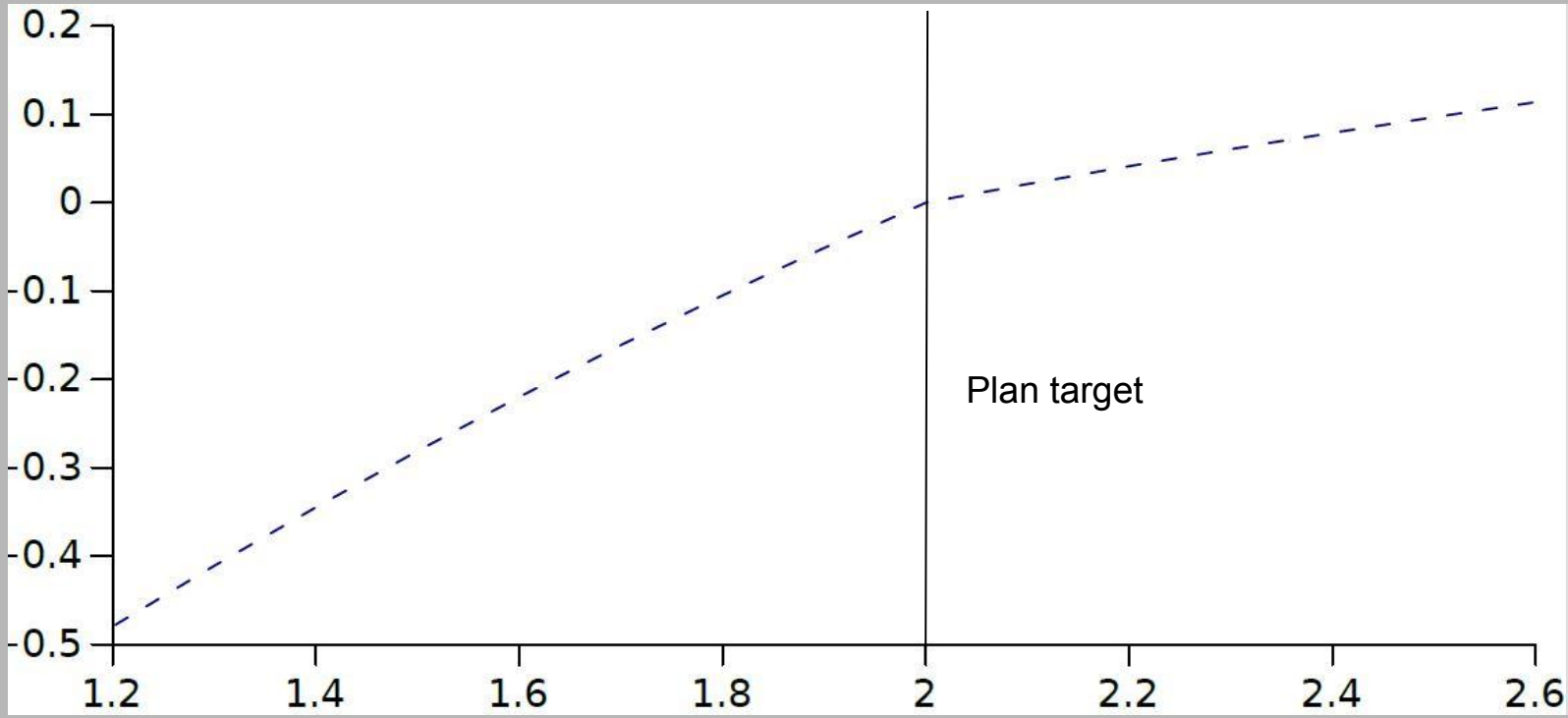
The Harmony Algorithm draws on ideas from marginalist economics and from neural nets to derive an iterative planning technique.

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

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

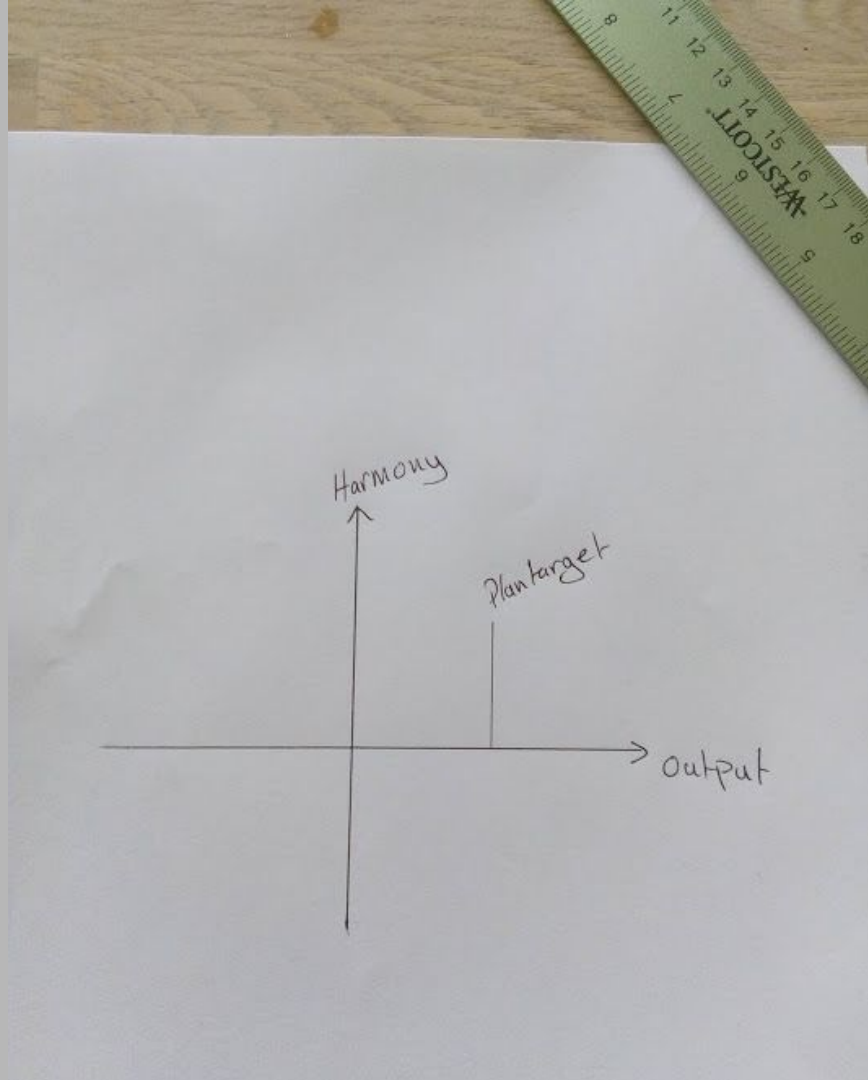
That is to say the proportional amount the plan has been exceeded by for each product.

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

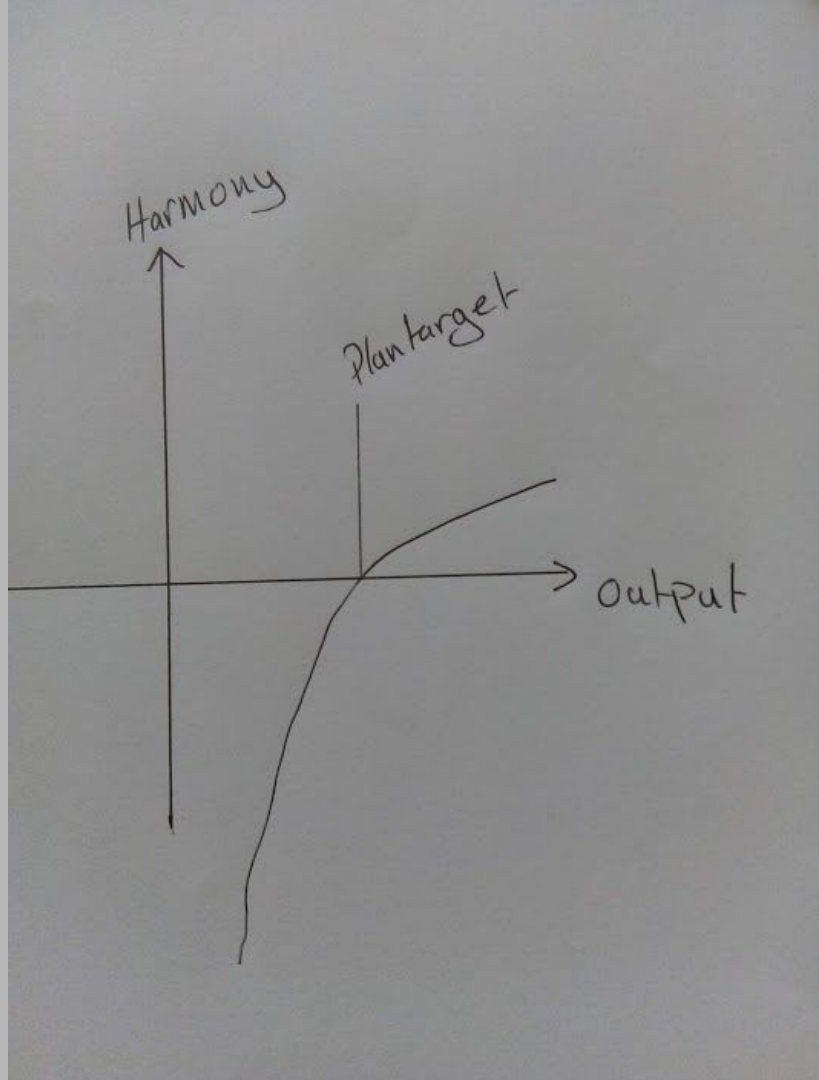
Suppose we start out with this scheme



Newton's approximation method

Suppose we start out with this scheme

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



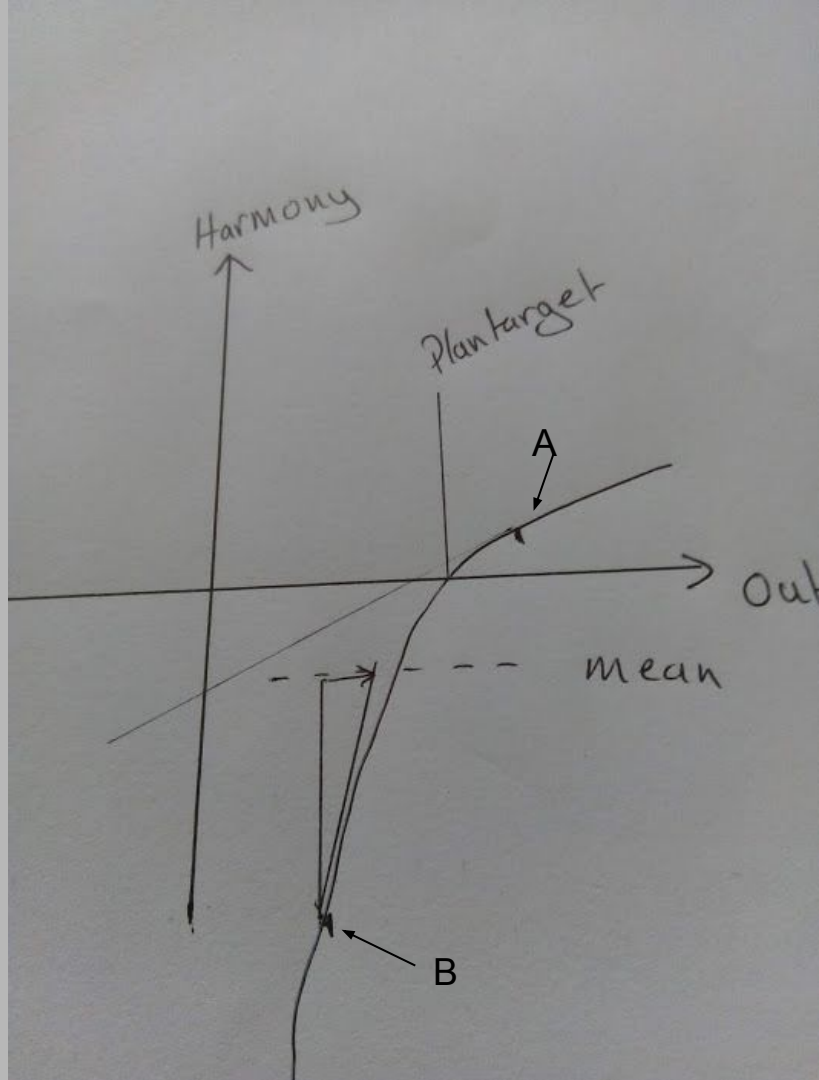
Newton's approximation method

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



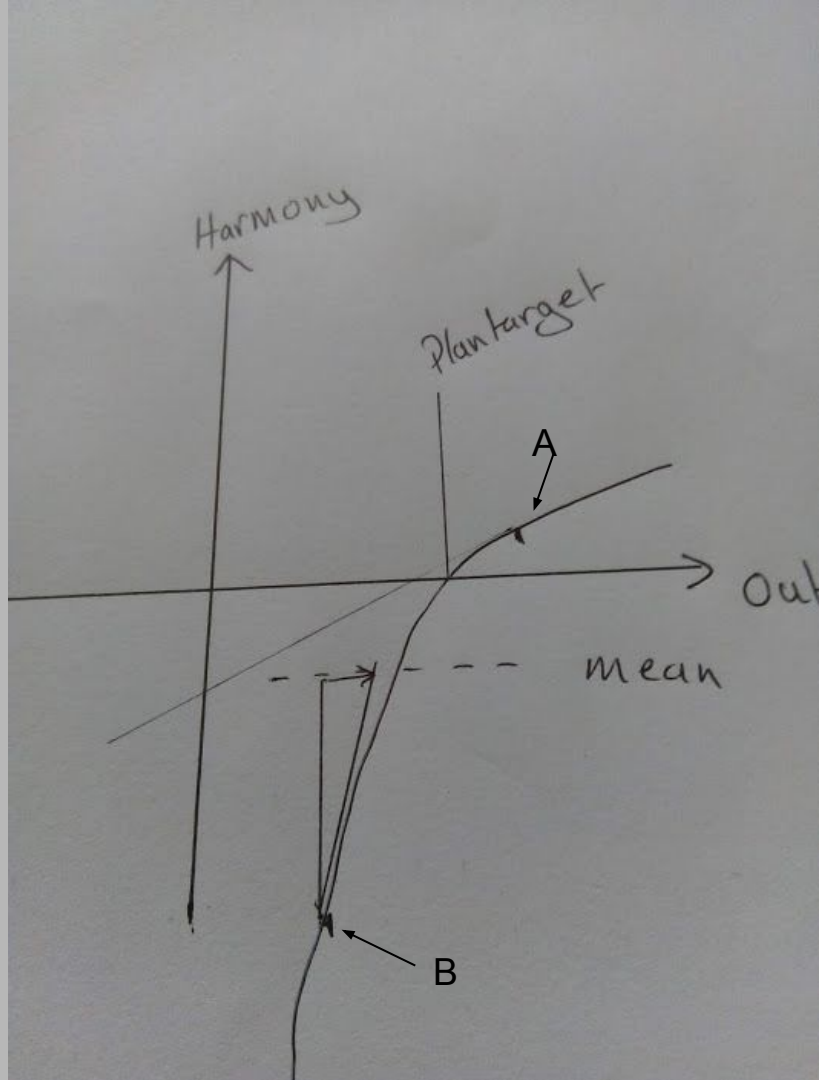
Newton's approximation method

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

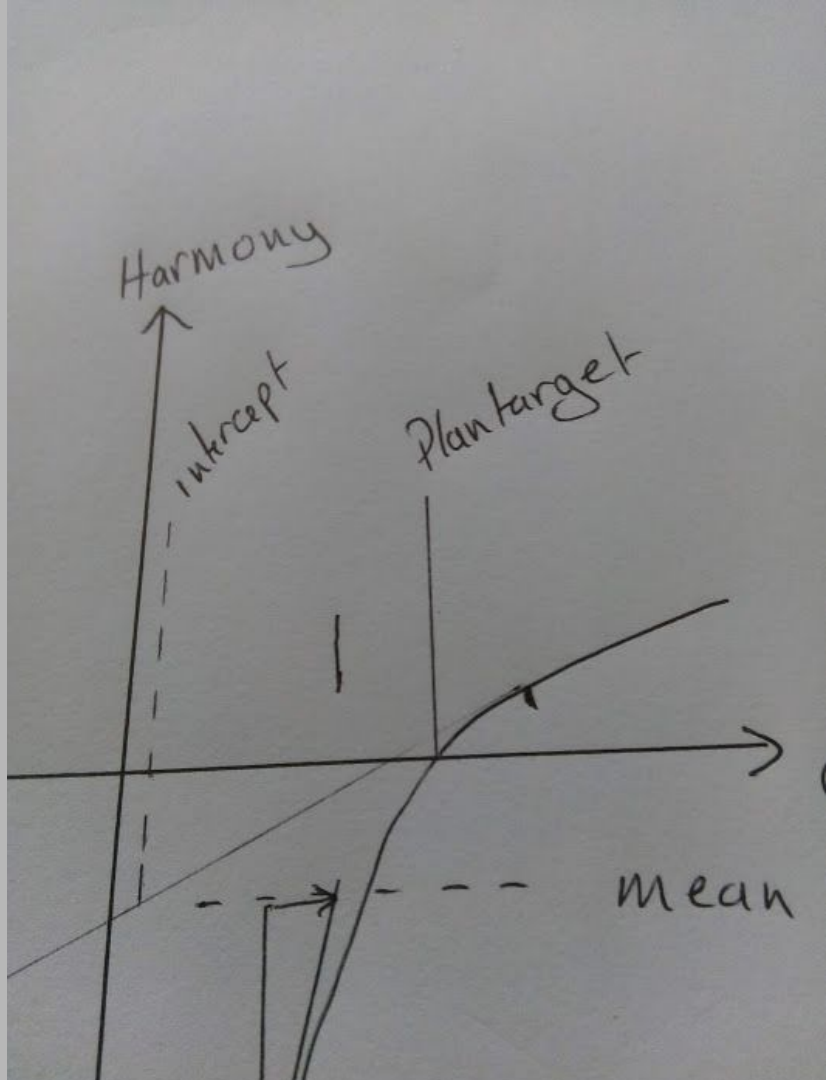
Since we know the gradients of the function we can draw or compute lines that intersect with the mean harmony of the whole economy



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.

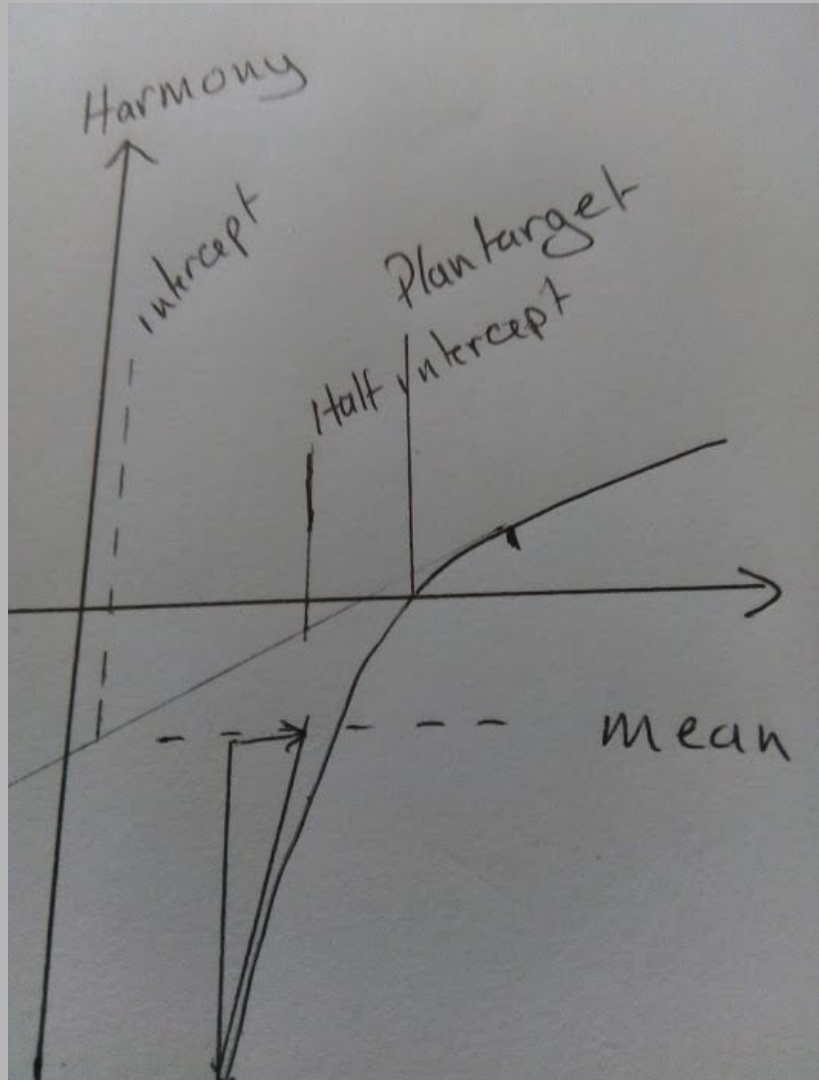


Newton's approximation method

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

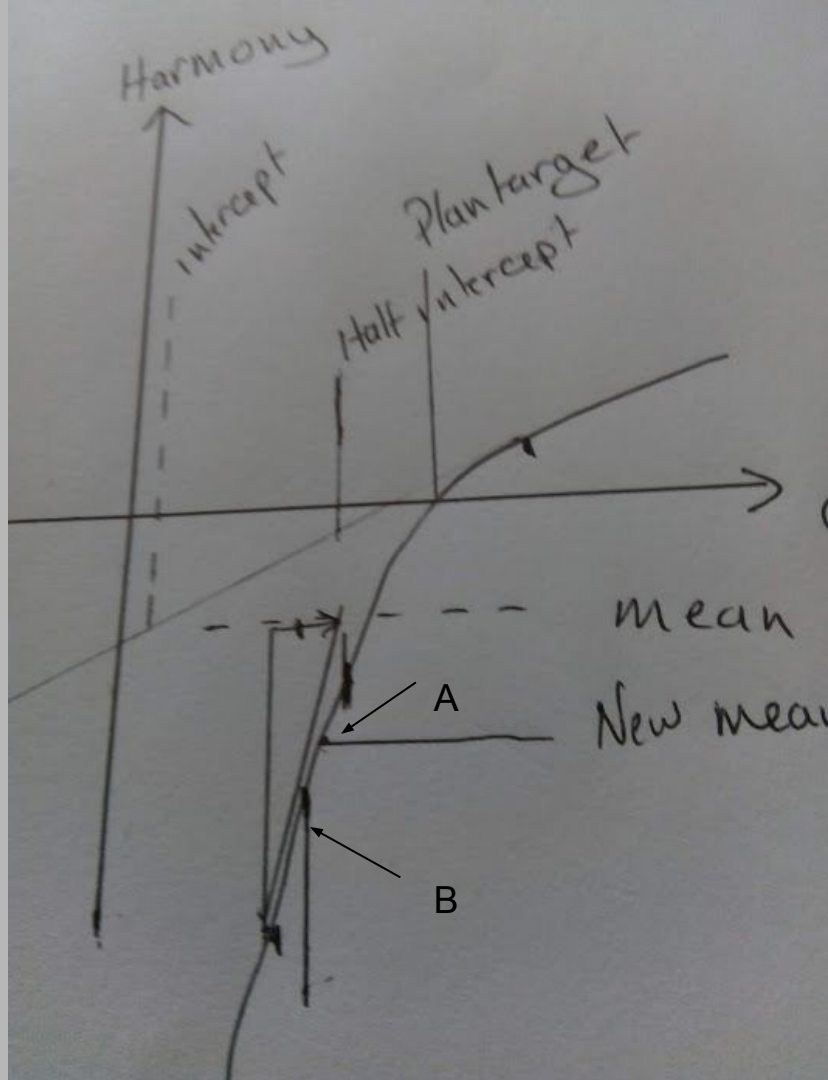


Newton's approximation method

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



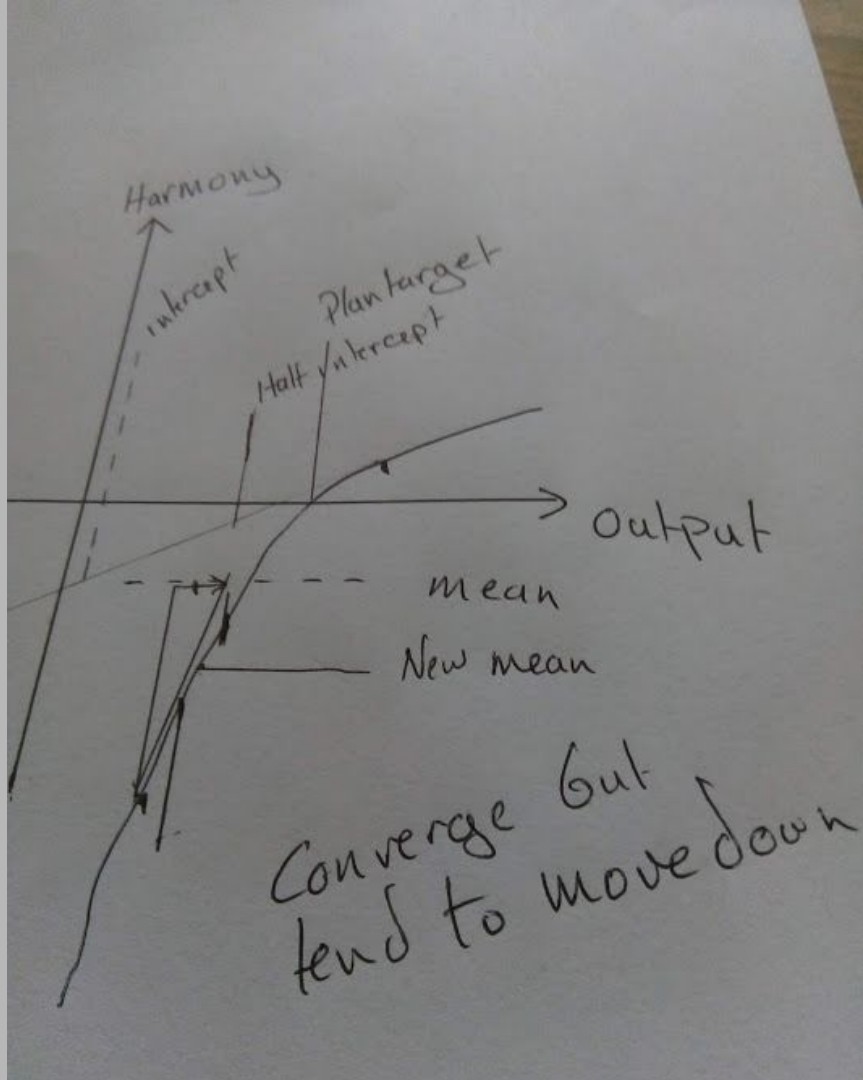
Newton's approximation method

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.

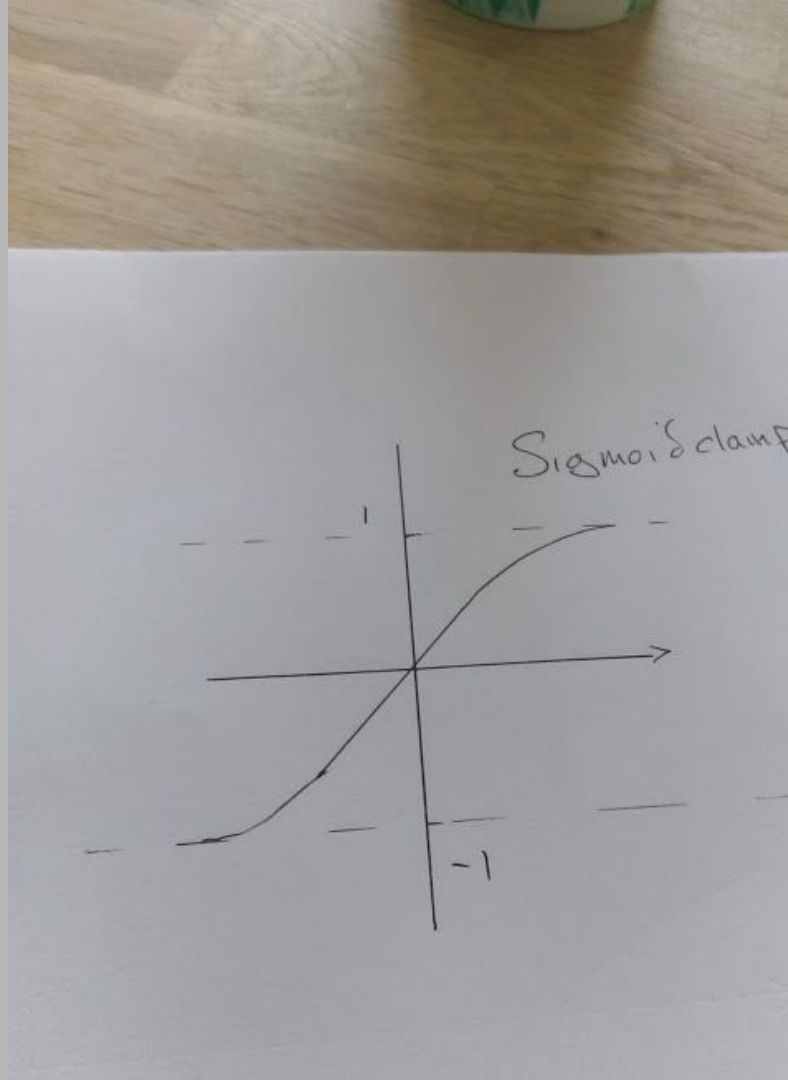
Use neural net technique

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.

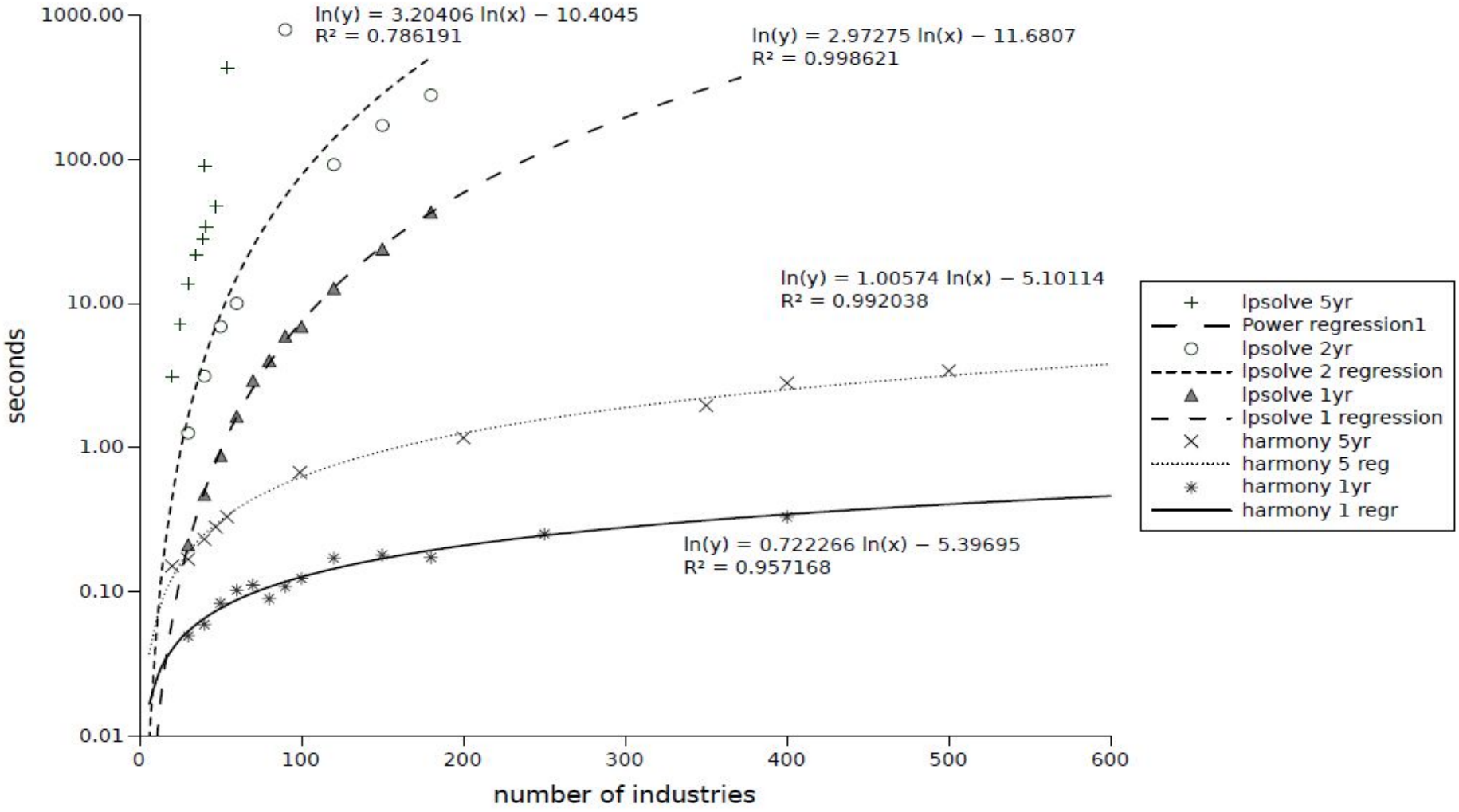
We then adjust the intensity I_j of production of each industry j thus:

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

Where phi and psi are tuning constants <1



Illustrate the use of the harmony algorithm


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.

Why computers better than markets

- The market can be viewed as computing engine - this is explicit in Hayek.
- Cycle time is slow, measured in months or years.
- Arrives at answer by physically adjusting production up or down.
- Constantly tends to overshoot in an unstable way.
- Human costs to these adjustments- poverty and unemployment

Computers are faster

- Computers can predict where an ideal market economy would get to if it ever had the chance.
- Production can then be adjusted directly to this target.
- Cycle time for computation is in the order of hours not years or months.

Computers and democratic control

- We propose system of online electronic voting on key issues like the proportion of national income to be allocated to health, eduction, research etc.
- This done in terms of the fraction of the working week in labour units that is to go on it.
- Taxes automatically adjusted to the democratic vote on social labour allocation.

References

- Towards A New Socialism, Cockshott and Cottrell, Spokesman, available from Amazon, pdf version online.
- A number of related papers from my google scholar pages.