Simulation by Repeated Optimisation

R.G. Coyle

Introduction

It is a cardinal point in system dynamics that the behavior of a model can be improved, often quite remarkably, by experimental changes to the parameters representing the system's policies. Usually, even more significant enhancements in behaviour stem from changes to the model's structure. It should be stressed, therefore, that, throughout this paper, 'parameter' includes those switches which can be used to activate or suppress components of the model's structure as well as those which represent its policies.

The only drawback is, however, that there is always a nagging doubt that, had one tried only one more experiment, something even better

Journal of the Operational Research Society (1999) **50**(4)

Professor Geoff Coyle has taken early retirement from his academic post. He continues to be active in system dynamics and was the recipient of the System Dynamics Society's first Lifetime Achievement Award.

R.G. Coyle (✉)
Shrivenham, UK

M. Kunc (ed.), *System Dynamics*, OR Essentials,
https://doi.org/10.1057/978-1-349-95257-1_7

would have been found. Unfortunately, there is *always* yet one more experiment, so the doubts can never be assuaged.

The problem of an experimental approach to policy design to improve dynamic behaviour is more difficult still. It has long been a tenet in system dynamics that feedback loops are the unit of analysis in planning and interpreting simulation experiments. That is probably true for very small models but is impractical for a model of even modest size. For example, the well-known world model is only a hundred or so equations yet it contains nearly 2000 feedback loops, most of which are practically impossible to detect visually, and it is clearly not possible to assert *a priori* that some which are easy to see are more important than the rest.

It would, therefore, be highly desirable to have some automated way of performing parameter variations and reporting to the analyst the best result. However the number of possible combinations and conceivable numerical values of the parameters is usually colossal so testing all parameter combinations is impractical. One needs, therefore, some sort *of guided* search of the parameters to be considered, and the numerical value each might have, so as to seek out the result which is most rewarding in terms of enhancing the system's performance, without pursuing blind alleys. Unfortunately, there is no perfect way of achieving that, but the principle of dynamic optimisation comes very close to providing this subtle searching of the design possibilities of the system.

This concept has been used in previous system dynamics work. Winch [1], for example, linked a FORTRAN version of a system dynamics model to optimisation software. This was laborious so, in collaboration with the Helsinki School of Economics, the DYSMAP package was linked to optimisation software [2]. Coyle [3] demonstrated the method for a production problem. Dangerfield and Roberts [4] describe some aspects of the approach. Coyle [5] has applied optimisation to a complex problem of defence planning, and Wolstenholme [6] similarly studied tactical choices by an attacking force.

There is, however, no compact account of the technique and this paper will discuss the theory of dynamic optimisation, which is not widely used within the usual disciplines of OR, illustrated by examples [7]. An important aspect of dynamic optimisation is the development of suitable objective functions which is also considered.

The System Dynamics products which originally supported optimisation in the late 1970s were DYSMAP2, and COSMOS, followed within the last few years by VENSIM and Powersim. All of them automatically link an 'ordinary' system dynamics model to the optimisation software, though the details and simplicity of doing so depend on the package. DYSMAP2 and COSMOS are very similar and use an efficient hill-climbing algorithm. VENSIM appears to use a grid search approach. Powersim's optimiser uses not more than five parameters and operates as an automated sensitivity tester. COSMOS is used to illustrate the ideas in this paper. For a fuller discussion of system dynamics software see Reference [7].

The System Response and the Parameter Plane

Figure 1 shows a two-dimensional picture of a three-dimensional object. The two dimensions in the horizontal plane are labelled for two of the parameters in a model. It is essential to realise that these may be ordinary policy parameters, or they may be structural parameters, or they may be 'pressure points', at which investments of resources could be made in a system. Each parameter has a range within which it may lie, shown as, for example, $P1_{UPPER}$ and PIlower- The ordinate is a measure of the quality of dynamic behaviour which the model produces for any given combination of parameters. For the moment, we shall simply label that scale as running from 'Bad' to 'Good'. The quantification of performance is discussed later.

The initial, or base case, values are labelled Pl_1 and Pl_2 on the parameter axes. When these two values are projected into the 'parameter plane', following the dotted lines, they intersect at Point A. When the model is run with those values, the response is, we shall suppose, rather poor, so a short line is drawn in the vertical direction to indicate that. This idea of the model's response at a point in the parameter plane is valid regardless of the number of parameters being used. In fact, in earlier work for a commercial client COSMOS was used to optimise a model of some hundreds of equations with 35 parameters, the runs took a few seconds, once the search parameters had been loaded.

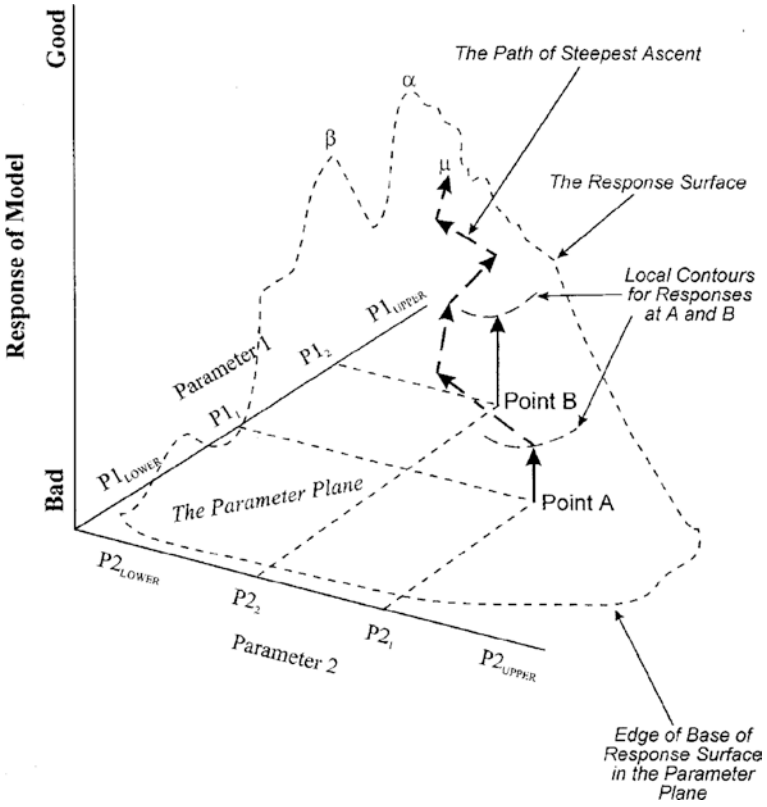

Fig. 1 The concept of hill-climbing optimisation

If the two parameters are now changed to Pl_1 and Pl_2, a new point B is defined which, we shall imagine, produces better performance and hence qualifies for a rather longer arrow in the vertical direction.

Figure 1 develops the idea of a response surface as the three-dimensional locus of the responses at all possible combinations of PI and P2. The response surface could well be a very rugged mountain, with several peaks. Two of the peaks, α and β, are of rather different heights and α is clearly 'better' than β. Figure 1 also suggests that there are all sorts of irregularities and gullies and that the mountain slope is much steeper in some places than in others; it could well resemble a photograph of the Alps. The reason for the extreme irregularities lies in

the complexity of system dynamics models and the non-linearities which they often contain.

It was argued earlier that one cannot test all possible combinations of parameters but, unless that is done, the shape of the response surface will be unknown. The problem of dynamic optimisation is how does one find one's way to the top of a mountain of unknown shape?

Hill-Climbing Optimisation

Hill-climbing optimisation can be understood by analogy with a blind man who is marooned on a mountain and wishes to find his way to the top. His strategy is to feel the shape of the ground around the point where he is sitting (the top of the arrow at Point A). Having detected the direction in which the ground slopes up most steeply in that vicinity, he takes a cautious step or two in that direction and then feels the ground again. In this way, he hopes to find the top of the mountain, as shown by the sequence of arrows moving up the surface of the hill from the point of the arrow at Point A, always pursuing the locally steepest direction. Unfortunately, the blind man's strength eventually fails him and he can go no further than μ. This is not even as high as β, so there is no guarantee that the blind man will reach his goal, α, or even another lower peak.

The analogy is implemented in some system dynamics software packages by using hill-climbing algorithms, developed in the mathematics of numerical analysis, which repeatedly iterate the model for particular sets of values of the collection of parameters being search. The first iteration uses the base case parameter values. After the model has run, the value of the objective function is calculated and retained, with the attendant parameter values, if it is the best result found so far. The sets of parameter values which give *good* results are used to predict how the parameter values should be changed to carry out the guided search for good values without wasting effort examining parameter combinations which lead nowhere. The reader interested in the mathematical technicalities should refer to the literature on numerical analysis; the pragmatist can rely on the evidence that the approach works.

The equivalent of the blind man's strength failing him before he reaches the top of the hill is not commanding sufficient iterations to be performed. If the algorithm is efficient there is no harm in demanding 500 iterations, if one wishes. Experience indicates that, for many models, about 30 iterations are sufficient to find dramatic improvements in performance. Even with a large model, a few hundred iterations take no more than a couple of minutes even on a modest PC.

An alternative to hill-climbing is a grid search in which the algorithm searches the space and gradually homes in on a good solution. This can be much more computationally demanding than hill-climbing. More recent techniques include genetic algorithms.

Overcoming the Limitations of Heuristic Algorithms

Hill-climbing by the method of steepest ascent is clearly heuristic and there is no guarantee of finding the maximum of the response surface. However, numerical hill-climbing algorithms are sophisticated and are capable of searching with something close to the intelligence that the blind man would use.

Consider Fig. 2, the vertical axis is still the model's response but the horizontal axis should be imagined to be a 'cross-section' of the parameter plane, representing varying combinations of the two parameters. The hill-climbing search has started at Point A and followed a steep ridge until it reached B. From there, a valley leads forward into the hills, and there is a ridge to the left, but the locally steepest direction is towards C. On reaching C, it becomes clear that it is a false peak, from which all directions lead downwards. Since the man is not yet exhausted, he would remember C as the best point so far discovered and, reasoning that he can always go back to C, he accepts the loss of height and searches towards D. At that point, he discovers a slowly rising path which emerges from behind peak C, as shown by the faint line, and moves off to E and hence, we hope, to F. As soon as any point is reached which is better than C, it will be remembered as the point to which he can return if no better solution is found.

There is still no guarantee that this course of events will unfold, so another strategy for avoiding failing to find the real peak would be to

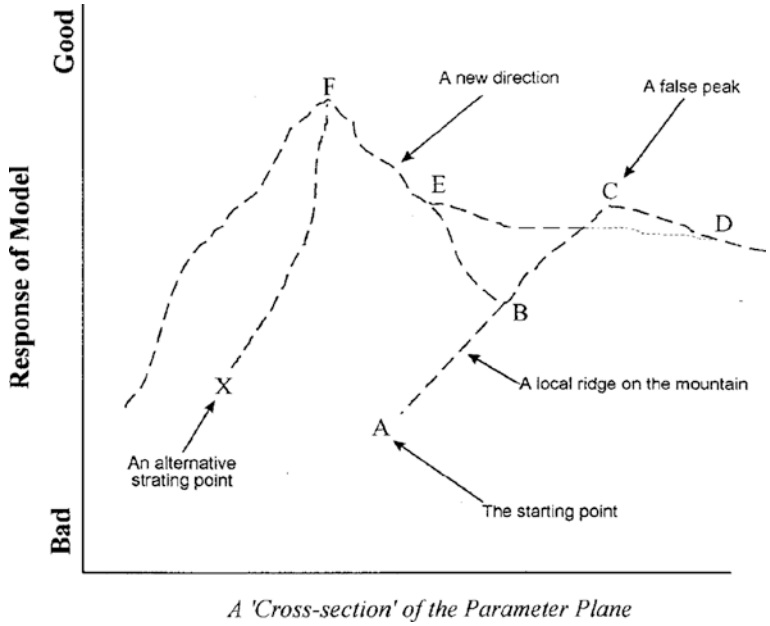

Fig. 2 Continuing to search

recommence the search at a new starting point, X, in the hope that there is a ridge going directly to F. In practice, the improvement in behaviour found in the first optimisation is usually so large that it is hardly worth the effort of repeating searches from new starting points. The main benefit of doing so would be to increase the experimenter's confidence that he has found a good solution, but there would be no end to that process.

The Performance of a Simple Model

Figure 3 is the Influence Diagram for a very simple model of a production system. The firm maintains inventory to meet unpredictable demands for parts. Inventory is replenished by ordering parts to be manufactured and delivered within a 'backlog elimination time'. The firm is hit by a sudden rise in consumption followed by an even larger fall, as shown by the solid curve in Fig. 4. The ensuing behaviour is clearly fairly disastrous. Desired

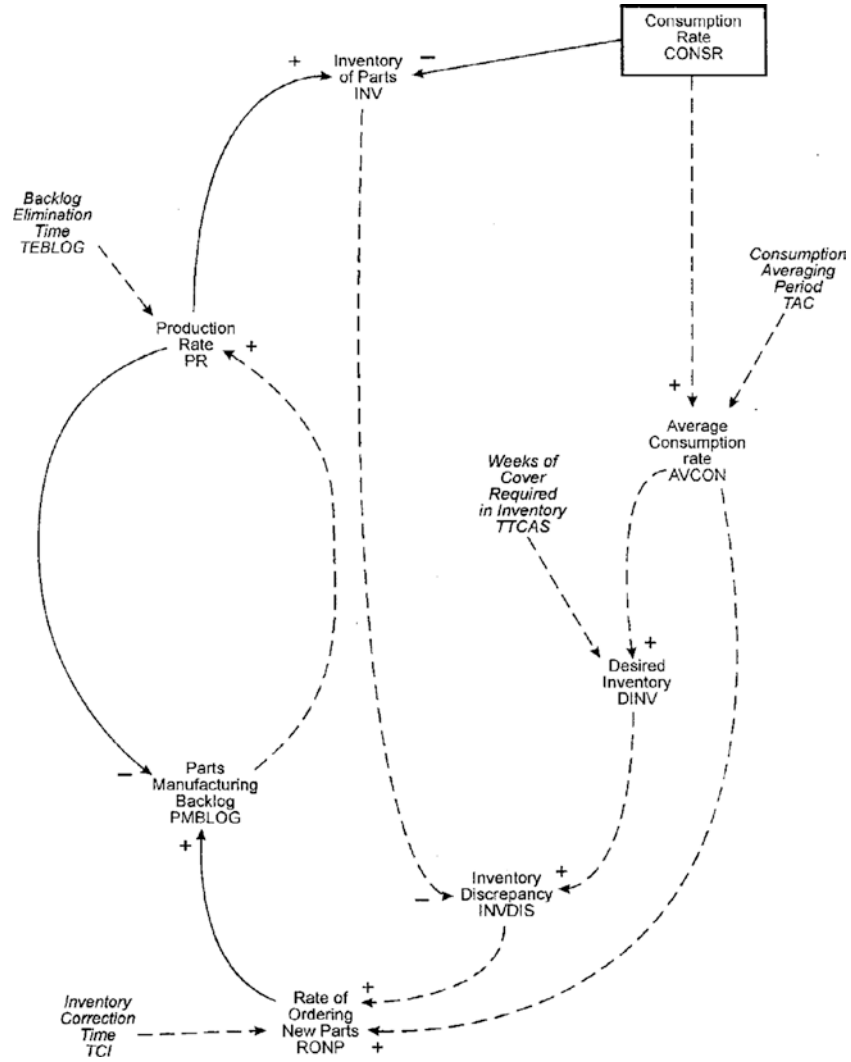

Fig. 3 Influence diagram for optimisation illustration

and Actual Inventory (DINV and INV) do not match until TLME = 140, which is two years after the shocks in consumption.

Furthermore, the parts backlog rises to a peak of about 1100 at LME = 30, but then falls to zero at TLME = 55 and stays there for

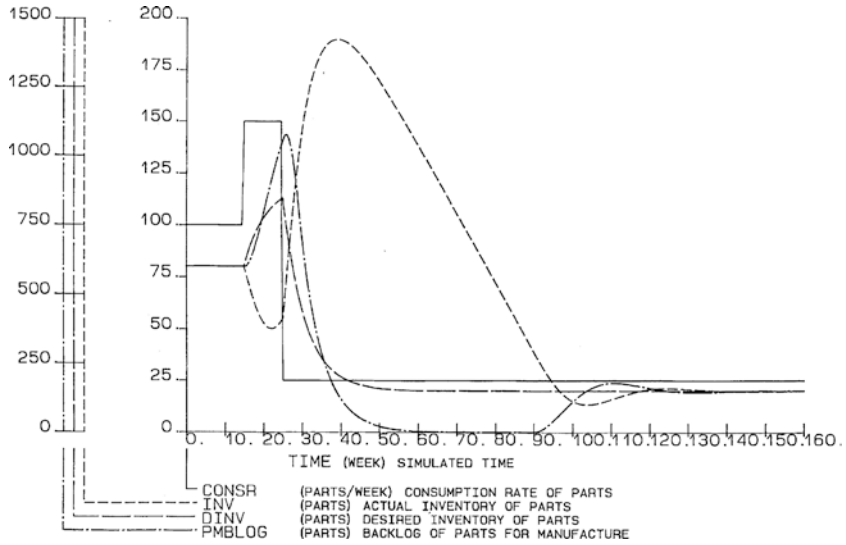

Fig. 4 Basic behaviour of the simple model (base case)

35 weeks. This corresponds to the factory having to be closed for about eight months.

Such a problem is small enough to be tackled by analysing gains and delays in the feedback loops but it is used here to demonstrate optimisation.

Formulating an Objective Function

To optimise, it is first necessary to formulate an objective function as a measure of system performance to guide the optimisation search. It should take account of what the system is trying to achieve and calculate the extent of its success. Developing objective functions for system dynamics models is something of a black art calling for much more research. Simple approaches based on common sense do, however, work well and, in this case, it seems reasonable to assume that the management objective is to make INV match closely to DINV. Much more sophisticated objective functions are used in some of the work cited earlier and that described below.

To write, using the standard DYNAMO format:

$$\text{A OBJFUN.K} = \text{DINV.K} - \text{INV.K}$$

would be misleading as it refers to a single point in time. One must take account of the behavior over the whole of the run, which requires a level variable to act as a memory of what happened during the run. To avoid the positive and negative discrepancies simply cancelling each other one might define an Inventory Penalty, INVPEN, as:

$$\text{L INVPEN.K} = \text{INVPEN.J} + \text{DT}^{*}\left(\text{DINV.J} - \text{INV.J}\right)^{**}2$$
$$\text{N INVPEN} = 0$$

The numerical value of INVPEN will be some strange numbers so, to make performance comparisons easier, we redefine OBJFUN.K as:

$$\text{O OBJFUN.K} = \text{INVPEN.K} / \text{SCALE}$$

where SCALE is chosen to make OBJFUN, the objective function we shall actually use, equal to 100 at the end of the base case run, simply to make it easier to think in percentage terms. In this case, SCALE is equal to 46.7933×10^4 which is found by running the model with SCALE = 1 and the base case parameters, observing the value of INVPEN at the end of the run, and calculating SCALE accordingly. In this case, it is evident that we wish to minimise OBJFUN, which is tantamount to the blind man finding his way down the crater of a volcano (hopefully extinct) by the method of steepest descent.

The Significance of the Objective Function

It is very important to be quite clear about the significance of objective functions. In the first place, they are extra equations added to the model for the analyst's benefit. They are not part of the real system and do not

necessarily have physical meaning. They are only there to help the analyst, and the software which serves him, to keep track of how improvements to behaviour can be found.

Secondly, the dimensions of the objective function do not have to have a real-world meaning. The dimensions of INVPEN are [WEEK*PART [2]], which does not correspond to anything in the real system, but that can be ignored. Obviously, there is nothing wrong in having an objective function which is dimensionally sensible, but the objective function, being an artefact of the analyst's thought, does not have to obey the strict requirement for dimensional consistency which applied to all the rest of the system dynamics model [8]. In this instance, SCALE has the same dimensions as INVPEN, so OBJFUN is a dimensionless ratio.

Thirdly, one has to be very careful about choosing objective functions. Minimising average inventory is an attractive though, but the true minimum inventory is zero and a firm which has zero inventory is usually not going to sell very much. As in all of OR, selecting an objective function is not a trivial matter and careful thought is needed about what the firm is really trying to achieve.

Optimisation Experiments

To optimise, one has to specify the parameters to be searched and to state the upper and lower values of each one. In this problem, there are four parameters, TAC, TCI, TTCAS and TEBLOG. TTCAS is a gain, the rest are delays. To start with, we allow all four to be in the parameter plane to be searched. All the base case values are 6, and they are all allowed to lie in the range from 2 to 10. These ranges are chosen purely for illustration but, in practice, much discussion with management would be involved and several ranges might be tried. This is often done most effectively during an intensive study period as optimisation will take only a few seconds on a Pentium PC. Thirty iterations are done and the results are shown in Fig. 5a.

The optimal values of the parameters are reported to TAC = 10, TCI = 2, TTCAS = 2 and TEBLOG = 2. The value of OBJFUN falls from 100 to 4.03. This reduction of practically 96% is by no means

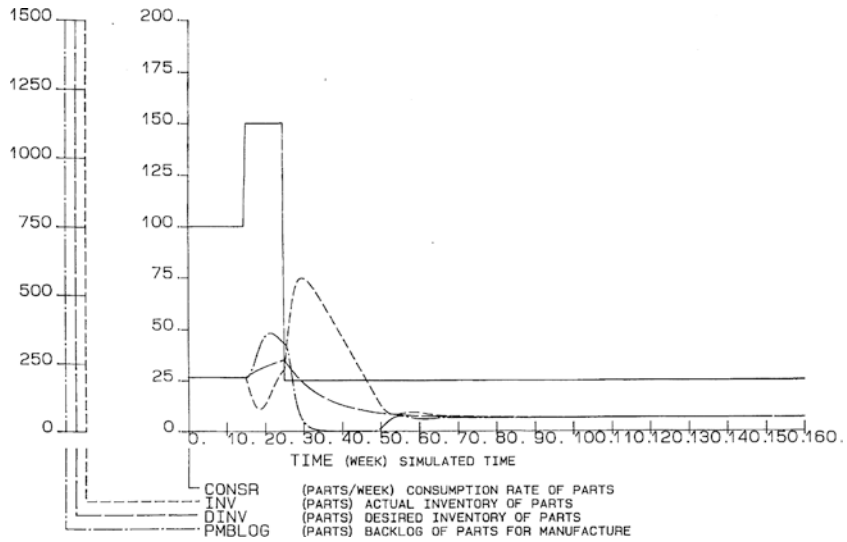

Fig. 5a Optimisation of the simple model (a) Optimisation of four parameters

unusual. All the parameters have been driven to their extreme points. Interestingly, the behaviour of other variables as well as inventory has improved, not surprisingly give the inter-connectedness of policies in a system dynamics model. A particular case is that the factory closure now lasts for only 15 weeks.

The key point is that this result was discovered after a few minutes of work to add OBJFUN to the model and a few seconds of computer time. To have discovered it by traditional loop analysis and experimentation would have taken much longer. The difference in effort would be *much* greater for a large model, even if loop analysis was tractable.

It is interesting to note that the gain, TTCAS, has been reduced, but that only one of the delays, TAC, has been increased while the other two delays have been reduced. It is a rule of thumb in control engineering that reducing gains and increasing delays is likely to increase stability but it has been clearly not worked in this case. Such rules are used in the traditional, loop-based, design approach in SD and, in this instance, optimisation has called them into question.

It is usually not a good idea simply to throw all the parameters into the optimiser and take the results on trust. To do so is to abandon thought

and rely on computation. In this case, we might feel that to increase TAC too much could make the system too insensitive to the unpredictable changes in CONSR, so another optimisation is done in which only TCI, TTCAS and TEBLOG are allowed to enter the parameter plane, with the same ranges as before. The results (Fig. 5b) are that all three are driven to their lower value of 2 and OBJFUN is reduced to 5.268. On the face of it, this is nothing like as good as the previous case, because OBJFUN is about 30% larger. In fact, the visible differences are very slight, as shown in Figs. 5a, 5b. The only difference between the two plots is that INV reaches a maximum of 562 with 4 parameters and 581 with three.

The optimisation with three parameters has driven the value of TTCAS to 2, which means that the firm is trying to operate with only two weeks of stock cover. That might be quite insufficient for any noise in the demand pattern and provides little protection against another upsurge in CONSR. One therefore optimises again with TCI and TEBLOG ranging from 2 to 10 as before, but TTCAS in the range from 4 to 10. The optimisation pushes all three parameters to their lower limits and OBJFUN is reduced to 12.84. In management terms, the parameter

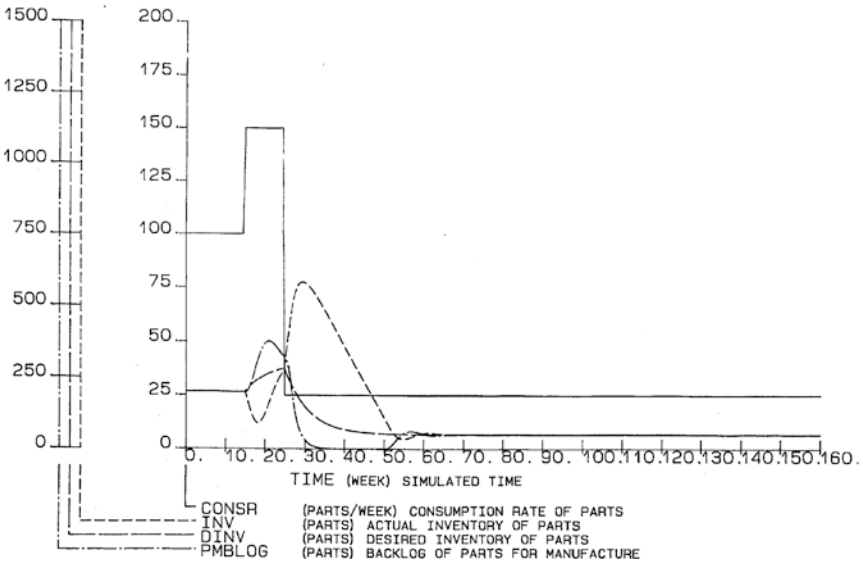

Fig. 5b Optimisation of the simple model (b) Optimisation of three parameters

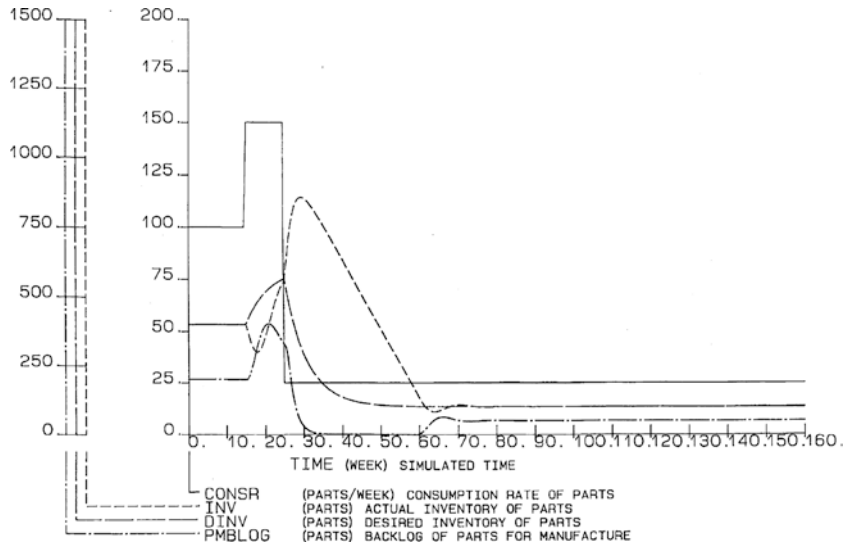

Fig. 5c Optimisation of the simple model (c) Higher stocks

values correspond to holding a prudent level of stock, but reacting quickly to changes. The behaviour is shown in Fig. 5c, and is clearly not as good as the cases in Figs. 5a, 5b. The maximum value of INV is now 857. The difference between 857 and 562 is a quantified indicator of the costs of a qualitative decision to be prudent, though the factory closure is now 25 weeks rather than 15, which might affect the cost of prudence.

This example has been deliberately chosen to be very simple so as not to lose the subtlety of optimisation in the explanation of a complex model. The real power of optimisation comes through with more complex models in which loop analysis can offer no guidance.

Simulation by Repeated Optimisation

The concept that the top of the hill is sought by repeatedly running the model makes it clear that the technique is optimisation by repeated simulation. However, the two other cases, with reduced numbers of parameters and with different ranges, should make it clear that the real underlying

theme is *simulation by repeated optimisation*. The model is optimised and something is learned. That leads to further optimisation and more learning, and so on. The value of the optimisation calculation is to provide a much more powerful guided search of the parameter space than could possibly be achieved by ordinary experimentation, but the principal aim is still to experiment and to understand so that a better experiment can be designed.

A notable example of this is the observation of differing periods of factory closure. The model could be optimized again to minimise that or to balance inventory against closure. In a more complex model, there might be many possible objectives, which would often be discovered by studying the effects of previous optimisations. Some of these objectives might, of course, be expressible in financial terms.

Objective Functions for Constrained Optimisation

The optimisation of the simple model was free in the sense that, in theory, no costs are incurred when adopting a new policy. Optimisation of resource investments is not free and the response surface can be visualised roughly as indicated in Fig. 6. The slope is as rough as the mountain but a barrier exists at the point at which all resources have been expended. Since the unit costs of different resources are different, the barrier has to be imagined as being moveable as large amounts of cheap resources may have a different effect from small amounts of expensive ones. Since one is, however, usually optimising several resources of differing unit costs, the barrier can be thought of as being on hinges, as well as on rollers.

Objective functions for constrained optimisation have to allow for resource costs and can be written in terms of OBJFUN as a measure of what is to be achieved and a penalty to be applied if the resource budget is exceeded. Therefore to maximise a performance index, INDEX, subject to a given budget, BUDGET, one could use:

$$\begin{aligned} \text{A OBJFUN.K} = {} & \text{INDEX.K} / \text{SCALE} - 100\text{E}06 \\ & \times \text{MAX}\left(0,\ \text{COST} - \text{BUDGET}\right) \end{aligned}$$

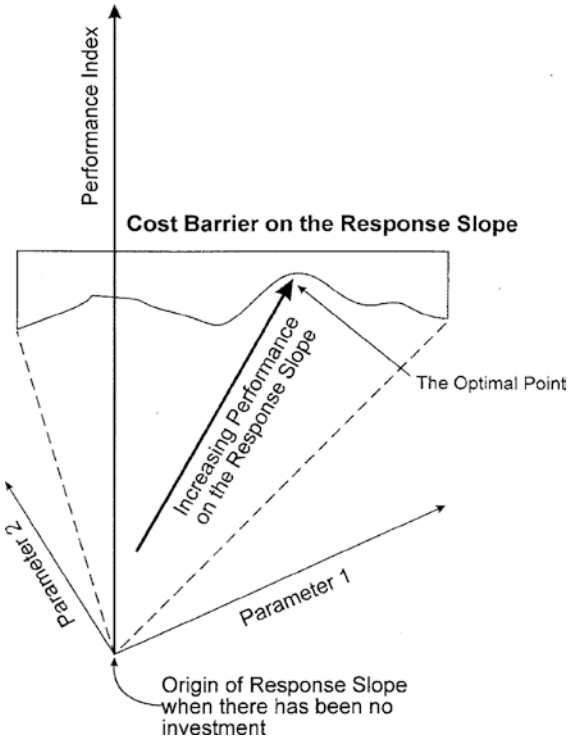

Fig. 6 Response slope for constrained optimisation

COST would be calculated to reflect the various resources expended and, as soon as it exceeds BUDGET, a massive penalty is subtracted from INDEX, making maximisation at that level of COST impossible. Note that, in DYNAMO terms, COST is a computed constant and hence has no K time postscript, the full details are covered in Reference [7]. It is trivial to extend this so that there are two COSTs and two BUDGETS when there are two resources such as manpower and money, and so on for any number of budgets and resources, such that exceeding any one would invoke the penalty.

It is equally trivial to adapt this process to minimising the expenditure required to achieve a given level of performance as opposed to maximising the performance for a given level of expenditure.

Constrained Optimisation—A Case Example

A defence example of constrained optimisation may illustrate the approach. In a model of land operations by a divisional formation, there are 10 parameters which can be freely changed and 20 which involve expenditure. The former represents 'decision rules' or concepts of operations, such as when to commit reserves, when to use forces of a particular type and so forth. The latter group includes such factors as the numbers of attack helicopters, the amount of ammunition, the numbers of combat engineer vehicles and so forth. There is a cost for each of these, such as each attack helicopter costs £X, and there is an overall budget, £Y.

Given a suitable objective function, such as the time for which the division could continue to fight an enemy force of a certain size, three optimisations can be done. In the first, only the 10 free parameters are used. In the second, the 20 costly assets are used with a budget constraint added to the objective function. In the third, all 30 parameters and the constraint are used. The results are in Table 1, in which the base case performance without any changes to any of the parameters is scaled to be 100, and the other performance figures are purely illustrative. F and C, with subscripts 1–3, denote vectors of the optimal values of the 10 free and 20 costly parameters, respectively.

The results are informative. The change from a performance of 100–120 with the free optimisation really means that changing F from F_1 to F_2 is the concept of operations which will get the best from the available forces. Similarly, the change from 100 to 140 as one optimises the costly parameters means that C_2 is the best force composition with the given budget for the existing concept of operations. However, the salient result arises when performance increases to 180 and both F and C change

Table 1 Optimisation of a divisional force

Base case	Free parameters only
Performance 100	Performance 120
Vectors: F_1, C_1	Vectors: F_2, C_1
Costly parameters only	Both costly and free parameters
Performance 140	Performance 180
Vectors: F_1, C_2	Vectors: F_3, C_3

to the new values of F_3 and C_3. This can be interpreted to mean that F_3 is the concept which gets the best out of forces C_3 or that C_3 will allow a better concept, F_3 to be adopted.

Finally, there is the point that the performance differences between 120 with free optimisation and 180 with costly and free optimisation is the marginal return to investment of the budget of £Y. The difference between 120 with costly optimisation and 180 is the marginal benefit of changing concepts to get the best results from investment.

Constrained Optimisation and Marginal Investments

Again, this is a defence example which summarises the work in Reference [5]. The small nation of Heroica faces potential aggression from its powerful neighbour, Nastia. Heroica is allied in its region with two other small nations, North Phalia and East Phalia. All three are threatened by Nastia and are allied with the distance superpower of Columbia. War is not certain but, should Heroica be attacked by Nastia, its forces would fight to maintain land, sea and air control of its territory. If Columbia is able to come to Heroica's aid, air reinforcements can fly directly into Heroica's air bases, if there is room for them, but land force reinforcements will have to travel through dangerous oceans to the region and will then reinforce Heroica, or one of the Phalias, depending on need and on whether they can cross Heroica, or one of the Phalias, depending on need and on whether they can cross Heroican waters in the face of the Nastian navy.

A very simplified influence diagram for the problem appears in Fig. 7 in which some of the feedback loops have been emphasised. The heavy black loop is the snowball loop by which, once Heroica starts to lose control, the loss will accelerate. The two dotted loops are command and control loops which re-allocate forces to prevent the snowball running out of control, or even to turn into Heroica's favour. Again, an objective function can be formulated to maximise Heroica's retention of control subjects to the available budgets for defence improvements over the next few years.

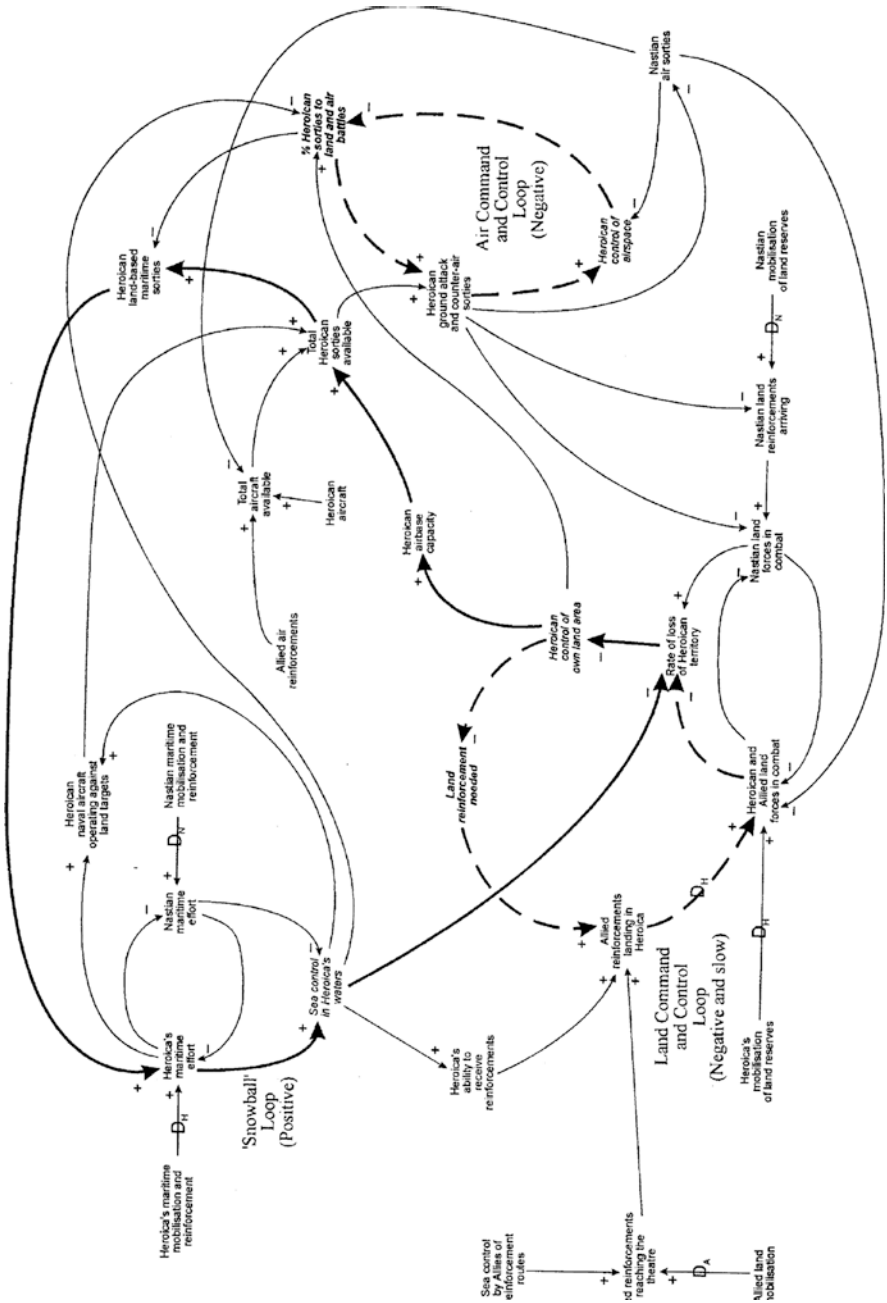

Fig. 7 Simplified influence diagram for Heroica's defence strategic planning model

Table 2 The optimisation results

Spending level	0	1	2	3	4
Overall outcome	Defeat	Time gained	More time gained	Enemy halted	Enemy defeated
Regular land force	2200	2400	3100	3100	3200
Land reserves	3300	3300	3300	3500	3600
Mobilisation delay	4	2	2	2	2
Air force	525	525	525	550	650
Air *base* capacity	4400	4500	4500	4500	4500

The problem is that the budget is unknown, so the optimisation is repeated with four separate levels of expenditure and using a few illustrative parameters, the results appearing in Table 2. It is, however, no use saying to the high command that 'if your defence budget is X your force composition should be such-and-such', as no-one knows what the defence budget is going to be over the next few years. However, scrutiny of Table 2 shows that the mobilisation delay is always driven to its minimum value, the regular army increases somewhat as the budget increases, but then levels off. The reserves and the air force only increase at high levels of expenditure and air base capacity does not change very much. That implies that the first priority should be to reduce mobilisation delay. After that, if there is any money left, the regular army should be increased to the point where it can defend the frontier for a few more days and, finally, if there is still some money left, it should go to the reserves. The air force, as its assets are costly (in this imaginary case), only benefits at very high levels of budget. In this case, optimisation has identified a *programme*, rather than a *single result*, which may, in practice, be the more useful product.

Concluding Remarks

There are numerous potential applications of optimisation, such as fitting models to historical data as part of the process of validation, and in business and government modelling, but space precludes a more extended discussion.

This paper has summarised the underlying theory of optimisation as applied to system dynamics models and shown something of the power of the approach by some examples. The main point to grasp is that optimisation, for all its superficial attractiveness, is not a panacea and it does not guarantee good analysis. There are, indeed, some limitations to the approach.

The first is that the hill-climbing algorithm does not guarantee an optimal solution. In practice, that matters less than might be thought because most managed systems perform so badly that any improvement is welcome and the differences between optima are usually much less than the objective function values might imply, as we saw in Figs. 5a, 5b.

Secondly, the optimisation technique does not, of itself, give any guidance on the development of a good, subtle, objective function. In many cases, the objective function reflects qualitative criteria, such as how well targets are achieved and these can have significant effects as they are linked to management thinking. A simplistic objective function, such as minimising inventory, might be truly disastrous.

The final weakness is that the thought of optimizing something is so seductive that the naive analyst might stop thinking.

In practice, it is only the second and third limitations which are serious and, provided one thinks, optimisation may well be a very powerful development in system dynamics.

Finally, if classical system dynamics involves improvement by successive simulation guided by the analyst's understanding of the feedback loops, optimisation is improvement by successive multiple simulation guided by an objective function which evolves as the analyst's understanding of the ability to meet objectives grows.

References

1. Winch GW (1977). Optimisation experiments with forecast bias. *Dynamica* **2**: 000–000.
2. Coyle RG (1997). System dynamics at Bradford University. *Sys Dynam Rev* **13**:311–321.

3. Coyle RG (1985). The use of optimisation for policy design in a system dynamics model. *Sys Dynam Rev* **1**: 81–91.
4. Dangerfield B and Roberts C (1996). An overview of strategy and tactics in system dynamics optimisation. *J Opl Res Soc* **47**: 405–423.
5. Coyle RG (1992). The optimisation of defence expenditure. *Eur J Opl Res* **56**: 304–318.
6. Wolstenholme EF (1990). *System Enquiry.* Chichester: John Wiley and Sons.
7. Coyle RG (1996). *System Dynamics Modelling: A Practical Approach.* London: Chapman and Hall.
8. See the work cited in Reference 7 for a full discussion of dimensional analysis.