Navigating the Wilds of Industrial Optimisation

David I. Wilson.

Industrial Information & Control Centre, Auckland, New Zealand

Abstract

The continual search for solutions that are better, faster and more efficient is second nature to all engineers. This activity is known as optimisation. But industrial optimisation problems are like the mythical beast, the Jabberwocky; they are big, complex, mean, ill-tempered, and prickly. What is interesting though is how we arrive at optimal solutions; how we can rapidly discard non-contenders, reduce the search-space, and accelerate the passage to the optimum. Essentially how do we optimise the optimisation process?

This paper reviews the recent developments in large-scale optimisation algorithms that are suitable for industrial problems. The important issues of correctly formulating the optimisation problem, judging when to add constraints, when to introduce binary variables, and which of the many numerical algorithms to choose are also highlighted with many actual industrial examples such as trajectory planning of the Waiheke ferry, to the optimal operation of steam utility boiler systems, to optimal design of microwave cavities, and the classification of the electrical power usage of suburbs from Dargaville to Wellsford. The take home message is this: With the right tools (many of which are free!), all the world's problems start to look like optimisation problems where even a slightly better solution is better than nothing at all.

***Keywords**: Optimisation, algorithms, large-scale, industrial applications, OPTI*

1 Introduction

If one looks at the common threads in an undergraduate engineering curriculum, then a clear candidate is the focus on the fostering of the development of a solid classical mathematical analysis**.** We can trace this back to the remarkable achievements during the age of enlightenment in the 18th century. So successful was science employing the newly developed mathematical tools such as calculus and probability in explaining natural phenomena, that those intellectuals at the forefront were regarded as rockstars. Newton ended his days in the politically appointed plumb job of Master of the Mint, Laplace was removed from his political office by none other than Napolean because "he brought the spirit of the infinitely small to the government", and poor Lavoisier saw Madam Guillotine from the wrong perspective, a beheading that shocked even the punch-drunk French science community at the time.

The scientists and engineers were successful in their pursuits to tame the wilderness, probably more so than any time before or since, due to the ingenious ways that could create mechanisms to (predominantly) improve life. Indeed the very word engineer is derived from the Latin *ingenium* meaning "an innate quality, especially mental power, hence a clever invention".

My reason for this truncated and highly biased history of technology is because I am interested in how good ideas are generated. Following the heady heights of the classical era, we hit the somewhat confused and chaotic period of the early 20th century where the universe suddenly became much stranger. Labelling the plethora of recently discovered sub-atomic particles (quarks) after a cryptic quote in an obscure work (Finnegan's Wake) by an impenetrable, penniless Irish author (James Joyce) shows just how desperate things had become. Things got worse of course, although there is nothing like a world war or two to stimulate the advancements of technology. Theoretical physics 'left the farm' about the time Sir Ernest fell out of his apple

tree, and became the big science it is today and engineers redefined their roles.

2 Optimising the optimisation process

For many engineers, their day is spent in the unending pursuit of improving something. The question is: how does one come up with these improvements, how can we shorten the development time, and how can we reduce the risk of failure? This continual search for improvement is essentially optimisation and we can embark on this in essentially two ways. The first, one currently favoured by the pedagogues, is to formulate an objective function, and then follow a well-programmed approach to reach an optimum, perhaps using some of the tools developed centuries earlier following what we now call classical optimisation strategies. An alternative search technique is to simply thrash about and hope that in our drunken wanderings we might stumble across a situation that is better than where we started. These strategies are known as random search or genetic algorithms. Both strategies have their adherents in the optimisation community and discussions can reach almost religious fervour.

But to get either approach to yield a significant finding, we need a preferably succinct and efficient model of the problem under consideration. Furthermore we need a scalar objective function that is a function of a collection of decision variables. It is these decision variables inputs to the problem that are actually the optimisation problem's solution. What is not often realised is that the simple act of crafting a well-posed optimisation problem is in itself a useful exercise. This forces one to think carefully about what the business is really trying to achieve, and what manipulated variables one can use to do this. We are forced to weight various competing objectives which in turn forces us to ascribe value. Simultaneously we explicitly state not only assumptions, but also state the constraints.

2.1 *Where is the optimisation taught?*

In the introduction I argued that mathematics was the glue within a professional engineering education. However looking at the typical curricular, we see a dominance of calculus and complex algebra, and the mastery of certain operations such as Laplace transforms, solving 2nd order ODEs and perhaps inverting modestly sized matrices. In my university, the content is not surprisingly driven by the various requirements of staff members delivering specialised senior level topics such as fluid mechanics, automatic control, electronics etc. What is missing though is a modern treatment of optimisation. Not the classical concepts, (such as solving for turning points in smooth univariate functions), but the tools and techniques for large-scale problems regularly seen in industrial optimisation. Ironically the one place we do see optimisation is the solving of linear programs (LPs) which is often included as part of our management or business courses

2.2 *The "industrial optimisation problem"*

Industrial optimisation problems are like the mythical beast, the Jabberwocky; they are big, complex, mean, ill-tempered, and prickly. They also do not like being conquered. To tackle them you need special tools, a steadfast mind, and good luck.

Figure 1: The Jabberwock from Louis Carroll's *Through the Looking Glass* as illustrated by the very disturbed John Tenniel.

Industrial-sized optimization problems exhibit the following characteristics:

(1) They are large and typically scale poorly.

(2) The objective function is never clear, and could well be considered multi-objective. (The hydra's multiple heads?)
(3) They are non-convex and it is often infeasible or even impossible to extract the partial derivatives of the objective function and constraints, hence the prickles.
(4) Real problems are always constrained which introduces the dilemma: Do we explicitly include the constraints to 'help' the optimizer by reducing the search space, or do we quietly ignore them.
(5) Algorithms that work for small-scale low-dimensional problems rarely are competitive for industrial-sized problems.
(6) Near optimality is generally good enough, and guarantees of globalist or quality of solution, while desirable, are of only secondary concern.

All these characteristics are well recognised and considerable time and expertise has been spent in developing hardware to address these problems. What is sometimes forgotten though is that the underlying algorithms themselves have also been in rapid development, and as Rice [1] shows in Fig 1, these software developments have at least kept pace with the oft-quoted Moore's law exponential hardware improvements.

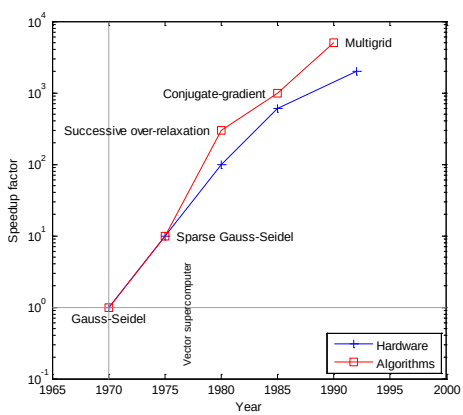


Figure 2: Comparing the improvements in speed due to both hardware and mathematical algorithms over the last 40 years.

The algorithms given in Fig 1 are only concerned with a single, but extremely important, computational task, that of solving linear equations. An important question is: Is it fair to extrapolate these improvements across all scientific computing, or even numerical optimization? To answer that, we must establish what are the key tasks, or computational molecules, used in scientific computing. One such list is known as the "7 Dwarfs of Supercomputing" originally proposed by Phil Collela, [2],

(1) Dense linear algebra
(2) Sparse linear Algebra
(3) Spectral methods, FFT
(4) N-body methods
(5) Structured grids
(6) Un-structured grids
(7) Monte-Carlo

and subsequently expanded by others to an unwieldy 13 dwarfs, [3]. In this context, a "dwarf" is a numerical method that Colella believed that was important for scientific computing. The field of numerical optimisation draws on almost all of these 7 computing paradigms while in the expanded version, dwarfs number 10 (dynamic programming) and 11 (backtracking and branch-and-bound) are of special interest in the fields of optimization. It is important therefore to leverage off these developments in both hardware and algorithms when choosing between competing optimization strategies. The good news is that many of these developments are incorporated in a transparent manner to the end-user.

2.3 *The "industrial user"*

One of the aims of our research group (www.i2c2.aut.ac.nz) is to evangelise the idea of optimisation; that is using optimisation to aid the design process and improve the outcome. The key point here is that we must always remember when talking to our customers or clients is this: they are the customer; their business is to manufacture rice cakes or milk power, to ensure that electricity is delivered in an efficient and timely manner or to improve the efficiency of a microwave to cook human biowaste ("poo" in the vernacular). They are not typically specialists in numerical linear algebra, they are

uninterested in the constant upgrading of processors to be in the top 500, and their optimisation problem tomorrow will almost certainly be quite different from the problem of today, especially if we are successful. Furthermore they tend to be reluctant to port their problem over to a super-computing cluster, possibly because of intellectual property concerns, but also because they may (mistakenly) believe that current generic installations are ill-equipped to deal with their specific problems.

This group we call the "industrial user", [4], and they comprise a surprisingly large and predominantly untapped market. These experts in their particular niche industry, but beginners in the field of operations research or optimisation, are the focus of our research group and the focus of this paper.

3 Some examples of industrial optimisation

The following four industrial examples each illustrate an interesting characteristic of industrial optimisation. Each case is an actual problem that our research group has worked on for an industrial client.

3.1 *Robust solving of algebraic equations: friction factors in pipe flow*

A common example in the optimum design of piping is the determination of the friction factor which requires solving the nonlinear algebraic expression (known as the Colebrook equation)

$$\frac{1}{\sqrt{f}} = -2.0\log\left(\frac{e/d}{3.7} + \frac{2.57}{\mathrm{Re}\sqrt{f}}\right)$$

for f. Solving Eqn (1) for f can be cast as a least-squares minimisation problem using standard algorithms. This is a classic piping design problem, although most textbooks (such as [5]) will sidestep the computational problem by choosing appropriate starting estimates and ensuring that the problems to be solved are well in the turbulent regime using some sort of direct iteration method. Such an approach is fine for one-off calculations, but in our case, we wished to design a set of mini-tool plugins for Excel that needed to be both fast and most importantly robust. In this case that meant we needed to devise robust starting algorithms for all reasonable piping problems, something that is difficult to achieve. Furthermore, when solving piping *network* problems, we need to be able to solve multi-variable algebraic versions of Eqn 1. This again is an optimisation problem where we have chosen to cast the problem to minimise the norm of the residuals. Such multivariable algebraic equation problems are notoriously hard to solve, especially at high dimensions, and the success rate depends strongly on the quality of the starting estimate. In this case, we know a considerable amount about the underlying physics of the problem, and we can place specific bounds in the parameter space in which to reduce the area in which to search for optimal solutions. Using this domain specific information is key to developing robust mini-tools, but at the expense of generality.

3.1 *Large-scale least-squares regression*

The state-owned enterprise Transpower is the electrical grid owner in New Zealand responsible for delivering electricity to the local retailers at 80 or so points around the country known as grid-exit points. To minimise the cost of correcting equipment, Transpower needs to regress a dynamic model of each exit point. Essentially this means that from Transpower's perspective, each suburb can be approximated by a single large static load and a single large dynamic motor.

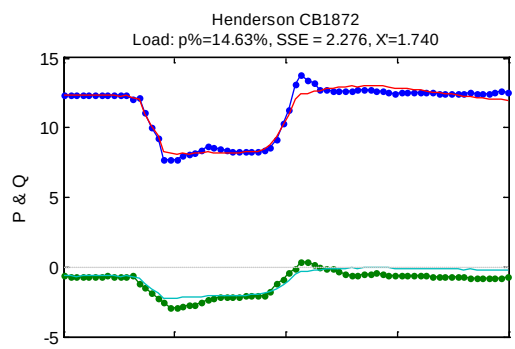


Figure 3: Fitting the reactive and active power models to experimental data at the Henderson grid exit point during a fault.

The characteristics of this optimisation problem are that the objective function was non-smooth, partly due to the embedded ODE inside the optimiser, but also due to the saturation components in the motor model. Consequently gradient-based numerical search techniques failed, but those using heuristic particle swarm techniques, whilst slow, did find adequate solutions. An interesting characteristic of this problem was that Transpower needed to regress these dynamic models for each of the 80 or so grid exit points; for different combinations of outages during weekends and weekdays, summer and winter comprising of almost 1000 combinations. Clearly such a procedure must be automated, particularly since we need to continually refine these models. Some of the exit points feed predominantly domestic suburbs; others feed areas of light industry, and some such as Glenbrook Steel mill, exhibit very unusual behaviour and occasionally require manual intervention. Such problems however are embarrassingly parallel and well suited to a naive parallel computing implementation.

3.2 *Computationally expensive problems used to optimise mechanical design*

To reduce the amount of municipal waste going to landfill, our client had the idea that he would pyrolise the wet media using microwaves. The resultant inert carbon could then be easily disposed or even used as fertiliser. The problem was an efficient design of the microwave cavity, [6]. Figure 4 shows the results of an electromagnetic simulation using the 3D finite element package COMSOL. In this case we needed to tweak the mechanical design in order to achieve even heating throughout the waste material tube which was extruded in a cylinder without exceeding a known temperature limit where undesirable dioxins would be produced.

This type of optimisation problem is characterised by an extremely computationally expensive function evaluation since we are to solve a full multi-physics PDE system, modelling both electromagnetic fields and the associated heat transfer. Parallel computing can help to a degree for these sorts of problems.

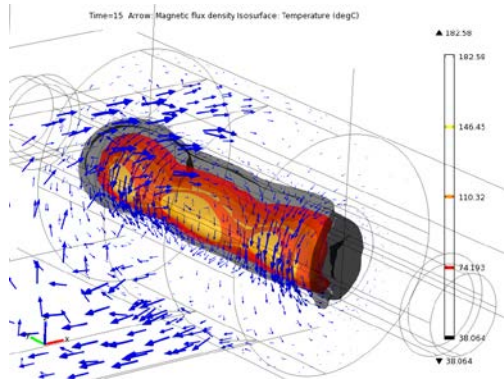
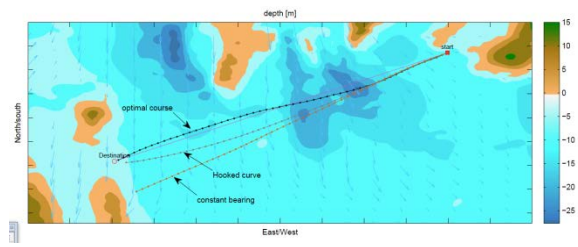


Figure 4: The temperature and surrounding magnetic flux density due to microwave heating of a tube of human waste biomatter flowing through a microwave cavity.

3.3 *Dynamic optimisation: Trajectory planning of a car ferry*

The various routes shown in Figure 5 give some idea of the choices the captain of the 500 tonne car ferry must make when travelling from the island of Waiheke to Auckland city in the Hauraki Gulf, [7]. The distance is relatively short, but shallow, and the area is subjected to strong tidal streams. Simply continually aiming at the destination results in a "hooked curve" which is clearly not optimal. Blindly following an initially correct bearing (perhaps while having a cup of tea) will put the vessel on the rocks!



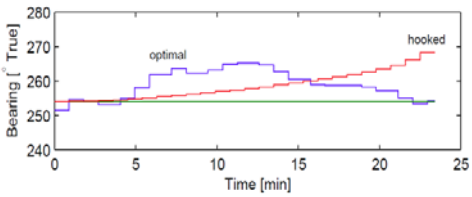


Figure 5: Find the optimum trajectory for a 500 ton passenger ferry battling tides and negotiating varying depths.

To find the optimum route in this case we must decide if we are to minimise fuel, passenger discomfort, passage time, or some combination. Furthermore the solution is a curve which means the solution is has an infinite dimension and one we could find using techniques such as variational calculus if the problem was smooth, which it isn't. These sorts of problems are known as an optimal control problem (OCP), and even if we were to use more modern techniques such as Pontryagin's method, we still need to embed an optimiser inside a boundary value differential equation solver. Again, these problems are notoriously delicate to initial conditions and prove difficult to solve, especially if the underlying dynamic system is non-smooth. However recent numerical improvements in the quality of the BVPs using collocation have improved the situation slightly.

As evident in Figure 5, we chose not to compute the true continuous solution, but rather discretise the problem into a number of discrete control moves.

3.4 *Optimisation problems with integer constraints*

Probably the most challenging optimisation problems are those where the objective function is nonlinear, and where some of the constraints are restricted to only integer or even binary variables. Such optimisation problems are non-convex and the current solution algorithms scale very poorly (non-polynomial).

Figure 6 shows a simplified version of a steam utility system our group developed for an international oil and gas company in South East Asia which wanted to establish optimum operating conditions given varying downstream demands for steam, and varying electricity supply costs, [8]. The problem requires integer constraints due to the possibility that boilers are shut down, or that turbines are bypassed. We, when formulating the optimisation problem, also have the option to approximate continuous nonlinear constraints as piecewise linear constraints combined with binary variables. It should be noted that simply truncating the relaxed optimum solution to integer or binary values does not typically result in the optimum, or even feasible solution.

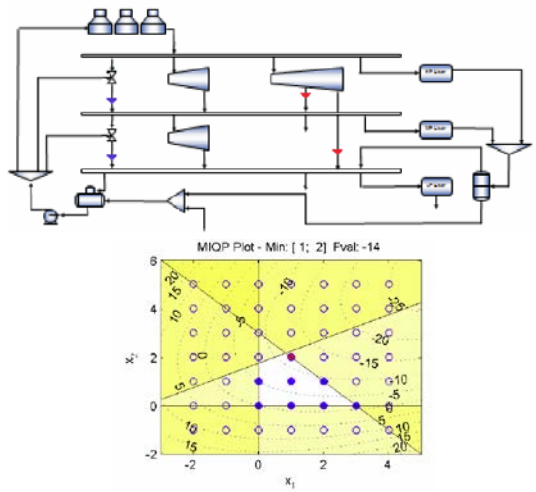


Figure 6: Establishing the optimum operating condition for a steam utility plant designed to service a large petrochemical plant is an integer optimisation problem.

A main hurdle in solving plant-specific optimisation problems is that the plant is continually changing topology and capability due to upgrades, or pieces of machinery being taken into, and out of, service. Unless the optimisation model is updated in parallel, any results should be treated with suspicion. For these sorts of applications, our research team developed a modelling environment that would automatically generate models that we termed "optimisation friendly". That is, the models executed quickly, and functions returning the partial derivatives of the objective function with respect to the decision variables are automatically generated, some even using symbolic calculations.

In this case, we could also optimise the steam utility system using a global optimiser, BARON, [9]. The interesting feature here is that while global strategies are orders of magnitude slower than competing algorithms, and the fact that they are applicable only to a reduced class of nonlinear problems, they do find better solutions. Of course whether that is significant or not depends on the application.

4 A general optimisation environment

The Industrial Information and Control Centre has developed a comprehensive optimisation suite which collects together a number of public domain optimisation codes to solve linear, quadratic and nonlinear programs (LPs, QPs and NLPs), mixed integer linear and nonlinear programs (MILPs, MINLPs), and nonlinear least-squares problems. The package is available from www.i2c2.aut.ac.nz in the software section and is designed to run via Matlab although most of the underlying algorithms are written in C/C++ or even Fortran. Some of the underlying algorithms such as IPOPT, CPLEX are generally regarded as world class in their specific domains, and the latter is available at no cost for academics.

5 Conclusions

Our research group encourages the many engineers who are charged with the challenging task of improving their product or process to pause for a moment to think about advantages of the discipline of optimisation. While optimisation in its own right is not often particularly stressed, or even taught in the undergraduate engineering curriculum, the readily available tools, (many of which are free), coupled with the rapid proto-typing environments such as Matlab (or the free Scilab) mean that even the non-specialists can take a structured, disciplined, and hopefully fruitful journey into the world of optimisation. We also argue that even if one decides not to continue formally to search for an optimal solution, the very act of constructing an objective function forces one to state what is really important for the organisation; the act of listing the decision variables forces one to acknowledge what is under their control and what is not, and the listing of constraints should be regularly revised since if some turn out to be relaxed, then the optimal solution may well turn out to be very different.

Acknowledgments

The author acknowledges the assistance of his past and present students in colleagues, particularly Jonathan Currie.

References

[1] John R. Rice. (1983) *Numerical Methods, Software, and Analysis.* McGraw–Hill.

[2] P. Colella, (2004) *Defining Software Requirements for Scientific Computing, presentation*, 2004. Public presentation available from Berkley, Media: DARPAHPCS.ppt

[3] Asanovic, K, Bodik, R, Catanzaro, B C, Gebis, J, Husbands, P, Keutzer, Patterson, D, Plishker, W, Shalf, J, Williams, S. and Yelick, K. (2006) *The Landscape of Parallel Computing Research: A View from Berkeley*, EECS Department, University of California, Berkeley, available from www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.

[4] Jonathan Currie and David I. Wilson, 2012. Opti: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In Nick Sahinidis and Jose Pinto, editors, *Foundations of Computer-Aided Process Operations*, Savannah, Georgia, USA, 8–11 January 2012

[5] Frank M. White. (2011) *Fluid Mechanics.* McGraw–Hill

[6] Jonathan Currie and David I. Wilson. (2011) Microwave Pyrolysis Study: Cavity design and numerical simulations. Technical report, Industrial Information and Control Centre, AUT., Prepared for SpectioNZ Technologies, PO Box 13-578, Johnsonville, Wellington 6440, New Zealand.

[7] David I. Wilson. (2012). Optimising Ferry Routes. In Tariq Samad and Dawn Tilbury, editors, *American Control Conference*, pages 3992–3997, Montreal,

Quebec, Canada, 27–29 June 2012. ISBN 978-1-4673-2102-0.

[8] J. Currie, D. I. Wilson, N. Depree, B.R. Young, S. Azmanai, L. Karim (2011) Steam utility systems are not business as usual for chemical plant simulators. *In American Institute of Chemical Engineers (AIChE) Spring Meeting*, Chicago, USA, 13--17 March, 2011.

[9] Jonathan Currie and David I. Wilson. (2012) The Efficient Modelling of Steam Utility Systems. In *Australian and New Zealand Annual Chemical Engineering Conference, Chemeca*, Wellington, New Zealand, 23–26 September 2012. Engineers Australia. ISBN 978-1-922107-59-6.