ADDING INTELLIGENCE TO A SCIENTIFIC DATABASE

F J Smith

School Of Computer Science, Queen's University Belfast, Northern Ireland
E-mail: fj.smith@qub.ac.uk

ABSTRACT

An experimental system called QPS (Quantitative Problem Solver) has shown that a numerical database of quantities in the physical sciences can be enhanced by adding intelligence for problem-solving. The system needs to store not only numerical data but also the formulae that operate on the data. It needs also the logical software that enables the system to find and combine together data and formulae to solve problems. It is shown that this logical software is similar to the backward-chaining algorithm used in expert systems with factual data. It has been successfully tested on a large number of problems, including many taken from textbooks in physics and chemistry and some taken from practical problems in engineering, including problems that need the solution of simultaneous equations, and including a novel solution to the problem of choosing the optimum material for a component. It has an interface based on the well known symbols used in equations; it can work with any system of units and it can check the accuracy of the calculations. The principle can be used in any numerical database that contains data which can be manipulated using formulae, not just in the physical sciences.

Keywords: Scientific Database, Statistical Database, Intelligent Systems, Deductive Systems, Quantitative problem solving.

1 INTRODUCTION

When data items are obtained from a conventional database system in science or engineering, whether through a query language or through a data manipulation language, they are almost always processed by other software to find some other needed information or to solve a problem. In some databases the data items are factual rather than numerical; then the software needed to process the data is Boolean logic software, i.e. the software used in expert systems (Bundy, 1979; Larkin, McDermott, Simon & Simon, 1980). However, in the physical sciences and in engineering the data are usually numeric (Rumble & Smith, 1988), the software is frequently written in languages such as FORTRAN or C and it almost always carries out calculations which are based on one or more formulae. In a simple example, an engineer might want the weight, *W*, of a cylindrical rod of known diameter, *d*, and length, *l*, made from an alloy of aluminium, say *Al-2024*. The engineer would know the formula to calculate the weight, i.e.

$$W = \pi \rho r^2 l \qquad (1)$$

and would know that he can calculate the radius *r* from the diameter *d* using the simple equation $d = 2r$. So he would then need the density ρ, of *Al 2024* from a database. Having found it, he might possibly have to write it down. before calculating the weight from the above two equations using a calculator, or (in a more complex example) he might enter the value of ρ as input to another computer program He might have to change the units of the value obtained from the database before entering it into the program. It would be more convenient and less prone to error if the database had associated with it a list of the formulae relevant to the data in the database and the built-in intelligence to find the right equations that the engineer needs to solve the problem. It would be better still if it could go further and use these equations to do the numeric calculations for the scientist and complete the solution of the problem to find the weight in the units required with an estimate of the accuracy of the answer. This has been the subject of the research investigation over several years with the experimental system called QPS (Quantitative Problem Solver), described in this paper.

Data are found in source books, handbooks, or in computer databases. However, lists of formulae are usually only found in text books; indeed most text books in the physical sciences and engineering contain a large number of formulae scattered throughout the pages. The symbols within these equations refer to scientific quantities, many of which refer to data values usually stored in a database. When they refer to database values they need to be automatically linked to these values in the database. Thus if a scientist uses the symbol ρ, a link

to *density* values in the database is automatically made. A link to the relevant formulae that use ρ is also made if the symbols, formulae and database are together in the one system. This is what we have done.

Therefore, in the QPS project we have integrated into a single knowledge base symbols, data and formulae together, sometimes called basic knowledge (Tyugu, 1988) or quantitative knowledge (Smith & Krishnamurthy, 1994). The resulting system has all of the characteristics of a scientific database, but in addition it can carry out manipulations of the data to solve problems by automatically searching in the knowledge base for the sequence of formulae needed to solve the problems. These formulae are then executed in the correct order, retrieving data and units from the database when a symbol in a formula refers to that data.

Before describing this process further, we first analyse the nature of quantitative knowledge and then describe the logical software needed to solve problems with this knowledge.

2 NATURE OF QUANTITATIVE KNOWLEDGE

We use the term quantitative scientific knowledge for the combination of numerical quantitative data and formulae together. A quantity can be a geometrical quantity like *area* or *volume*, or a physical quantity like *mass* or *force*. A geometrical quantity is a variable which depends on the geometrical shapes under consideration and their dimensions (e.g. the radius r of a sphere) or the spatial relationship between any interacting objects (e.g. the distance, d, between the centres of two spheres). Physical quantities can be categorised into *units*, *constants*, *properties* and *variables*. *Units* we discuss later. *Physical constants* are the universal constants of nature, such as the velocity of light (c), $2.9979\ 10^8$ m sec^{-1}. Physical *properties* are quantities which hold different values for different materials (or elements) in different states, for example, *elastic modulus* (E), $0.91\ 10^{11}$ *Pascals* for *brass* at room temperature. The physical *constants* and physical *properties* are normally held in a database. Physical *variables* (sometime called state variables) are independent variables which describe the state of a physical system, such as, *temperature* (T) or *pressure* (P). These variables (including geometric values) are either specified by a user or computed by the system.

As mentioned earlier, quantities are normally identified by symbols, usually single Latin or Greek characters with or without subscripts or superscripts or both. The association between symbol and quantity is well known to scientists. In certain cases a single symbol may be used to represent two or more different quantities; for example the symbol E is used to represent *energy*, *electric field* or *elastic modulus*. But this ambiguity can be removed using the dimensions of the quantity either in its specification or in a formula. Thus, if a scientist specifies that $E = 0.91\ 10^{11}$ *Pa* (where *Pa* is the symbol used for *Pascals*) then E must be elastic modulus. If the formula: $E = m\ c^2$ is used, then E must be *energy* to be dimensionally correct.

A formula describes the relationship among quantities and is expressed in the form of a mathematical equation. It can be a simple definition of a physical quantity (e.g. $r = 1\ m$) or a physical law describing a phenomenon or fact. For example, the quantity *tensile stress* is defined by the formula;

$$\sigma = F/A \tag{2}$$

where F is the force applied and A is the cross-sectional area. An example of a physical law is the elastic extension δl of a rod length l_o area A under a force F:

$$\partial l = \frac{l_0}{E}\frac{F}{A} \tag{3}$$

where E is the elasticity of the material of the rod. This has a constraint: that it is valid only if the strain $\varepsilon = \delta l/l_0$. is less than a critical elastic breaking stain ε_e. This constraint can be represented as a conditional rule (called a production rule in Expert Systems):

$$\text{if } \varepsilon < \varepsilon_e \text{ then } \partial l = \frac{l_0}{E}\frac{F}{A} \tag{4}$$

The traditional backward-chaining logic of expert systems (Luger, 2005) can be applied to conditions like this to search the knowledge base for formulae that match the conditions specified with the problem to be solved. Systems such as Postgres (Stonebaker, 1991) which support objects and logical rules can be used to represent such knowledge. However, finding a set of relevant laws which obey the constraints does not usually solve the problem. There are likely to be too many of them. It is almost always much quicker and simpler to find the appropriate formulae first and then check that their constraints match the specified conditions when they are about to be used, eliminating those that do not match (Loughlin, 1998). The main problem comes, not from the conditions, but from finding the sequence of formulae needed to solve a problem. This is the principle problem in handling quantitative knowledge.

3 RELATED WORK

There have been some attempts in the past to build systems which can store and manipulate scientific formulae. For example, some of the early intelligent programs to solve Physics problems were MECHO by Bundy (1979) and ISSAC by Novak (1977). MECHO was developed to solve problems in Mechanics. Formulae were represented using predicate logic. In ISSAC formulae describing the relationship between objects in a problem were represented as procedures attached to a "Canonical Object Frame". This has been more recently developed to solve problems in astonomy (Novak, 1997), in a system called VIP (View Interactive Programming). A demonstration can be found on the internet. Other similar systems are SIGMA (Scientist's Intelligent Graphical Modelling Assistant) developed by Keller, Rimon & Das (1994) for atmospheres and ecosystems and SCARP (Development Shell for Co-operative Problem-solving Environments) by Williamowski (1995). There have also been several systems developed in the area of artificial intelligence for solving problems in Geometry (Welham, 1977).

One of the most comprehensive studies on quantitative scientific knowledge has been undertaken in Estonia by Tyugu and co-workers (Mints & Tyugu, 1988; Tyugu, 1991; Tyugu & Valt, 1997). They developed a system called PRIZ based on "Computational Models" which are special kinds of semantic networks for representing quantities in Mathematics and Physics and formulae relating these quantities. It used a special language for describing physical problems called UTOPIST. Another similar language for describing quantitative models called ASCEND (Advanced System for Computations in Engineering Design) was developed in Pittsburgh for Engineering (Piela, Epperly, Westerberger & Westerberger, 1991),

Though these systems were powerful in exploiting the domain knowledge and facilitating specification of the problems in natural language, they mostly emphasised the programming environment of the scientist. In QPS research has originated from studies on adding intelligence to scientific databases. An early system was developed for the storage of data on science and technology (Smith & Hughes, 1985) with the special feature that it included a facility for the storage of formulae as functions and subroutines that could operate on the data in the databases. A similar system was also developed to process more general queries using the relational data model and the Query-by-Example interface (Bandyopadhyay, 1987) and following on this Bandyopadhyay, Hughes, Smith & Sen (1994) developed a system called "Symbolic Information Management System (SIS)" for the storage and manipulation of structured data and algebraic formulae. This used a model called the "Symbolic Relational Model" which was a derivative of the relational model. In SIS formulae were represented as special attributes of a symbolic relation.

All these systems facilitate storage of both data and formulae. The QPS project was a new attempt to develop the above ideas further, in an object-oriented environment which provides a uniform paradigm to represent quantities, formulae, units and geometry in one knowledge base that was lacking in our earlier work. It was also based on an interface using the same mathematical symbols as are used universally in formulae by scientists.

4 OBJECT ORIENTED KNOWLEDGE REPRESENTATION

An Object Oriented design was used throughout QPS both for the data and for the software. It was not essential to do this; a quantitative knowledge base could have been built without O-O technology. But it was highly convenient to do so as we now show. A Class Hierarchy diagram for the data showing the relationship between some of the various entities in the object-oriented knowledge base is shown in Figure 1.

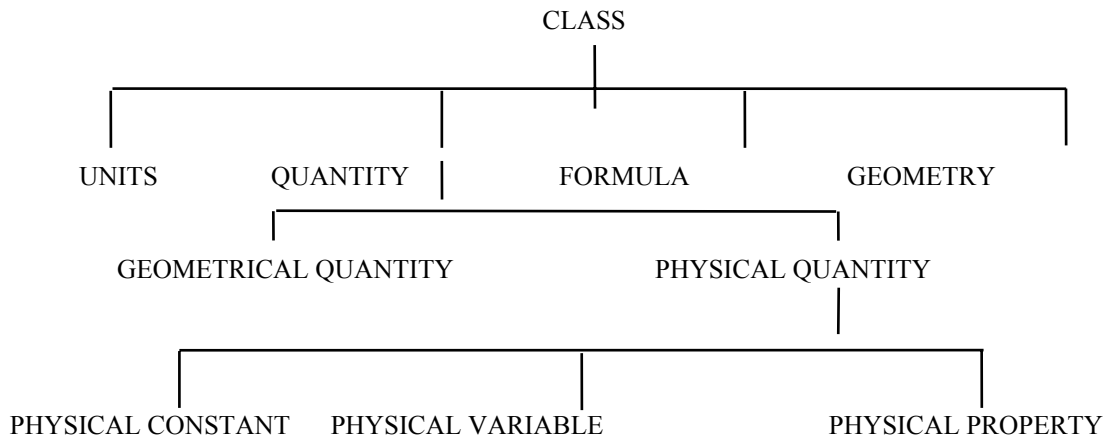

Figure 1. A Class Hierarchy Diagram showing some Data Objects stored by the QPS System

Most entities are implemented as an object or a class. For example the class *Physical Quantity* represents the attributes common to all scientific quantities. At the leaf nodes of the tree each instance of a class represents a real-world object, for example *Force* is an instance of the class *Physical Variable* and *Density* is an instance of the class *Physical Property*. Inheritance is used to allow objects and classes to inherit attributes from a higher more general class. For example the classes *Physical Property* and *Physical Constant* share common attributes like *dimensions* that they inherit from *Physical quantity*, which in turn inherits some of them from the class *Quantity* But they differ in the type of data that they represent.

In addition, the classes *Formula* and *Geometry* are described in terms of other system objects; Formulae are composed of several Quantity instances. Geometries are represented by a set of *Geometrical Quantity* objects that define the shape, and a set of *Formula* objects that relate the Geometric Quantities. The Geometry model is a hierarchical one which, through the use of inheritance, is capable of describing 3 dimensional structures in terms of their cross section plus the extra dimension of *depth*.

The QPS system software has an Object-Oriented design. At the lowest level are the definitions and methods of the primitive data structures used universally within the system, for example *list* and *set*. Also at this level are the objects that implement the system's data types, e.g. real numbers and fuzzy numbers (Kaufmann & Gupta, 1991). The next level defines the physical schema, using a filing system based on B-Trees for efficiency and the link to the database. This link is handled by one class that submits to the database the names of the data to be retrieved and the environmental variables, like temperature, and returns the data required with its unit or units, and if available, its accuracy. Also at this level are classes to retrieve the formulae corresponding to a particular quantity. These can be held in a special structure or in the database.

The objects defined in the level above the Knowledge Base exist to perform the problem solving process. Amongst them is an object which co-ordinates the systematic search for a solution and an object to evaluate equations arithmetically There is also an object to translate equations and quantities from the symbolic format an engineer or scientist employs and views at the interface, into the internal unambiguous format used by QPS.

5 COMBINING DATA AND FORMULAE

We next discuss how to solve a numeric problem using formulae and data. The problems we aim to solve are common problems for engineers and physical scientist, the computation of the value of a quantity by applying formulae. This is usually simple when there is a single formula to apply and the values of all other quantities in the formula are known, either specified in the problem or available in the knowledge base. An example is already given in Equation (1), $W = \pi \rho r^2 l$. If we are given r and l and can find ρ from a database then it is straightforward to insert these values into the formula and compute the weight, W. If, however, W and l are given and we need r then the equation needs to be reorganised in the form $r = \sqrt{W / l\pi\rho}$ before calculating

r. This requires symbol manipulation software to invert the formula. This was used successfully in QPS (Krishnamurthy, 1993). Occasionally, however, the inversion is not possible. For example, a simplified form of the formula for the current *J*, in a valve due to thermionic emission at temperature *T*, is given by: $J = AT^2 e^{-S/kT}$ where *A*, *S* and *k* are known values. It is straightforward to compute *J* if *T* is given, but this formula cannot be inverted to give *T* in terms of *J* if *J* is given (the more likely question needing an answer). This must be solved numerically by finding the zero value of the function: $J - AT^2 e^{-S/kT} = 0$. This too was used successfully in QPS and this numerical method has the advantage that it can be used for all inversion problems, even when they are simple. So symbol manipulation software is not needed and it was replaced in later versions of QPS by a numerical solution in all cases.

The process becomes more complex when there are more than one formula to apply. This happens when the value of at least one of the quantities in a formula are unknown (neither specified in the problem nor available in the knowledge base), but when it can be computed from another formula; this can be repeated in the second formula and a third formula is needed and so on.. A simple example was already given at the beginning of this paper. The weight *W* cannot be calculated directly from Equation (1) since the radius *r* is not given. But *r* can be computed from a simple second equation: *d = 2 r,* first by inverting it to *r = d/2* and then using the given value of the diameter *d*. So using the two equations in the correct order gives the solution. How do we do this in more complex problems? Humans find it difficult – as every student knows!

6 RECURSIVE DEPTH-FIRST SEARCHES

The system's search strategy is illustrated below with a sample problem using a simple knowledge base in Mechanical Engineering.

Example A cylindrical rod, made of *aluminium,* with length *1 m* and diameter *0.5 cm* is hanging from a fixed support. If a weight of 2 Kg is attached to the lower end find the resulting elastic elongation of the rod in mm.

The problem is posed in the following or similar form using the symbols in the interface.

Ontology mechanics
Geometry: cylindrical rod
Material: aluminium
l_o = *1 m*
d = *0.5 cm*
W = *2.43 kg*
δl = *? mm*

The QPS system solves this problem using the strategy shown by the example of a search tree in Figure 2.

.

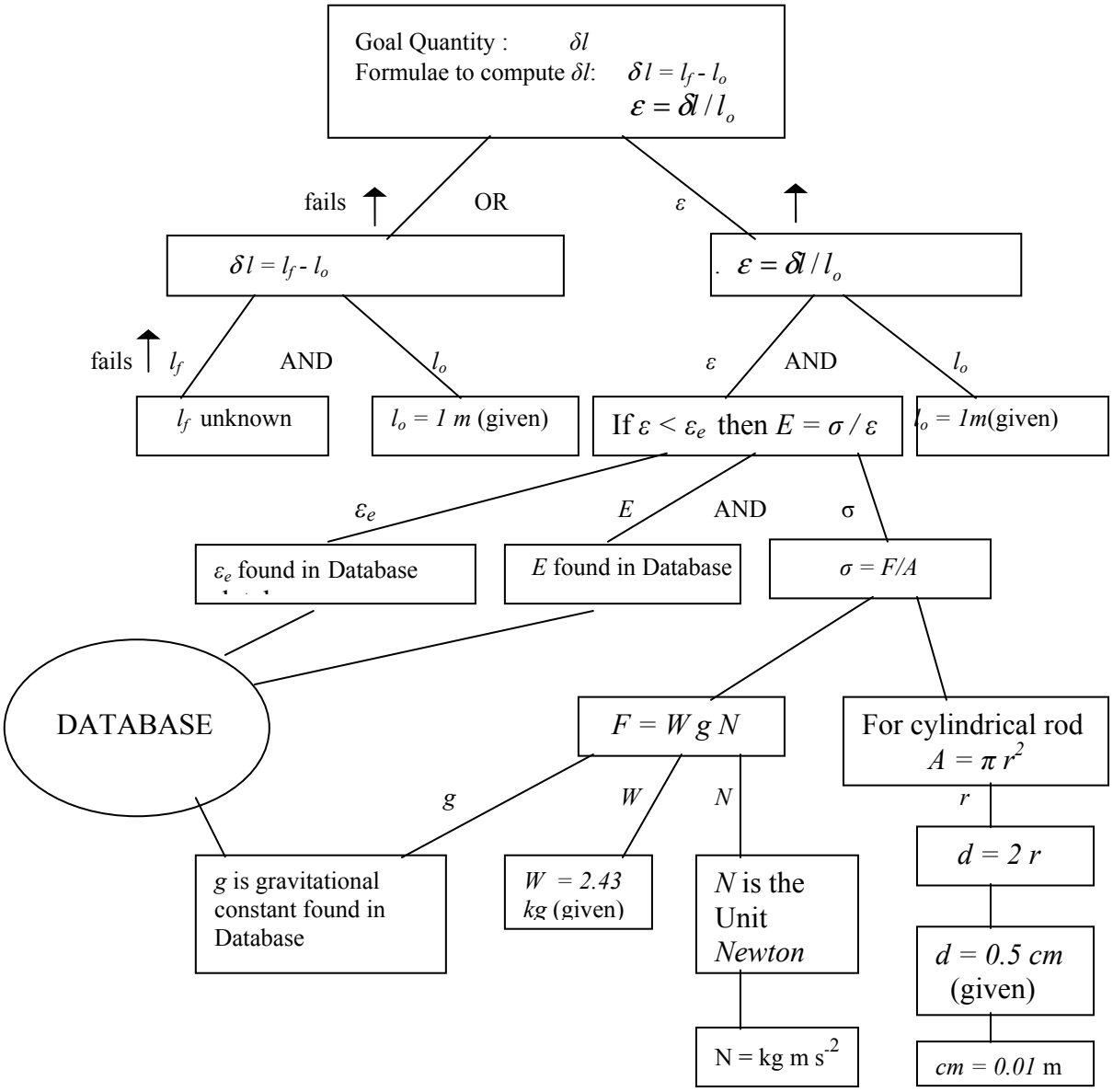

Figure 2. The Search Tree formed by QPS to solve the Example above.

It begins by looking for the goal, in this case δl, and finds all of the equations that include this goal, beginning with those in the ontology (domain) of the problem (if given). In QPS there were 6 equations in the ontology for *mechanics*, but we show only 2 of these for simplicity. There are then two OR branches to the tree corresponding to the two alternative equations. The first of these is $\delta l = l_f - l_o$. To evaluate this we need the values of the two other variables l_f and l_o in the equation. This results in two AND branches in the tree below the equation. To evaluate them, each now becomes a new goal. The second of these l_o, is given as *1 m*, so this value is returned and that branch of the tree stops there. The second variable l_f, (the final length) is unknown and there are no other equations involving l_f (apart from the equation $\delta l = l_f - l_o$ already used); so the system fails at this point and returns a failure back to the equation in the node above it. Since this equation $\delta l = l_f - l_o$ cannot now be evaluated it also returns a failure to the top node. The system next tries the second branch of the tree from this top node to the second equation $\varepsilon = \delta l / l_o$ and the process described for the first equation is repeated. This equation, besides the goal δl, includes two other variables, ε and l_o, the second of which is known and the first ε, becomes a new goal. Its value is unknown, but a search finds one other equation that includes ε which in turn becomes a new node to be evaluated. In a few recursive steps the variable ε is evaluated. The constraint is then checked, i.e. that $\varepsilon < \varepsilon_e$ after ε_e is found in the database. The value of ε is then returned. This enables the goal δl to be calculated since l_o and ε are both known.

This process described above of a depth-first recursive search in an AND-OR tree is known as backward chaining in Artificial Intelligence (Nilsson, 1980; Luger, 2005). The leaf nodes represent quantities. If their values are known it is returned to the previous level; but if no further progress can be made a failure is returned. When a dead-end is reached for a selected formula, the system backtracks to a previous level and selects another formula if available. If the complete search space is exhausted, the system reports that the problem is unsolvable and prompts for more information.

A study of over 100 problems in mechanics, electricity, magnetism and gas kinetics, taken from elementary University text books shows that all but a few per cent can be solved successfully by backward chaining. However, a common difficulty arises if the problem needs the solution of 2 (or more) simultaneous equations in 2 (or more) variables. For example, this difficulty arises in the example above if the final length l_f is specified rather than the initial length l_o. In a depth-first search one equation containing both variables will call a second equation to obtain a solution, but if this second equation also contains both variables it will call the first again, causing infinite recursion. QPS is capable of detecting potential infinite recursion, and if found will abort the depth-first search by trying formulae containing the goal quantity at each step in parallel (Collis, 1996). The process detects if two (or more) formulae have two (or more) unknown quantities in common, and if so, they may be solved simultaneously. A component then computes an answer using a numerical method known as the Broydn algorithm which can be used on both linear and non-linear simultaneous equations (Press, Teukolsky, Vettering & Flannery, 1992).

Another problem related to databases that cannot be solved using a depth-first search and is important in engineering design is the selection of an optimum material to meet a specification of a component in a design. It may be concerned with the strength, size and weight of the component; e.g. a bolt has to be a certain shape and size, and withstand certain stresses. This requirement has to be turned into a specification of a material; this needs the calculation using the correct sequence of formulae of the maximum breaking strain for the material. This specification of the material can then be used for matching materials in a database. The problem is similar to the one described above, but it was found that a forward chaining algorithm was more appropriate than the backward chaining algorithm described in Figure 2 (Winstanley, Loughlin and Smith, 1998).

7 UNITS AND ACCURACY

In the example above, we have not discussed how the units of the quantities in the solution are handled. Most physical quantities are expressed in terms of units; e.g. in the example above the specified quantity *d = 0.5 cm* is made up of a numerical value and a unit. The sentence *d =0.5 cm* can be treated by the logic of QPS just as any other equation and the expression *0.5 cm* is interpreted as *0.5* multiplied by the unit quantity *cm*, just as *0.5 x* would be interpreted as *0.5* multiplied by *x*. So the same logic that is used to process quantities and equations can be used to process units and physical quantities together.

All calculations are carried out in SI units. Therefore, at input, specified data are converted from the user's units into SI units. Data from the database may also have to be converted if not already in SI units. The decimal multiple prefixes, such as *m* for *milli* or *k* for *kilo* are converted to their numerical values before processing begins. For example, 0.5 cm is changed to 0.005 m. The user is always required to specify the units of the result and conversion is made at the last step before the result is displayed or printed.

In most scientific and engineering problems data and quantities are rarely shown exactly (Rumble & Smith, 1988). Some possible error is always present with real data, either measured or calculated or obtained from a database. This is often not specified. However, in the example above the weight is given as *2.43 kg*. This suggests that there might be an error of ±0.005 kg. It is well known that such errors can accumulate during a calculation, for example when two similar values are subtracted from one another. Sometimes calculated values are meaningless. So it is important to carry forward the possible errors at each step in the calculation. The system has therefore been designed to deal with such imprecise or fuzzy data using fuzzy arithmetic (Kaufmann & Gupta, 1991), including the specification of quantities in terms of both values and errors, or as ranges. The aim is to find solutions to problems as numerical values, but always with error estimates provided. This facility, although not fully tested, presented no difficulty in principle.

8 CONCLUSIONS

We have shown that the knowledge based system QPS can have the properties of a scientific database system, but at the same time it can use the data in the database to solve problems as in an expert system, with facts replaced by numerical data and logical rules replaced by formulae. It has applications to physics, chemistry and engineering design in circumstances where occasional calculations using data and formulae are needed which would otherwise require a special computer program. It has proved to be successful in solving over 100 problems in mechanics, electricity, magnetism and gas kinetics taken from textbooks. The only ones which cannot yet be solved are those more complex problems that involve differential equations, matrices or composite body problems, although these present no difficulty in principle. In many cases these more complex problems require whole programs to solve them. In these cases the execution of a formula in QPS could be replaced by a call to a whole program. The result would be the same as the evaluation of a formula: values of variables would be read into the system and computed values of the required variables would be returned by the system, whether that system is a program, or a procedure, or an equation.

The QPS system can be used not only to solve numerical problems in the Physical Sciences and Engineering, it can also be used in any numerical database where the numerical data are manipulated using formulae. This might include economics, medicine or biology. Indeed QPS has been used to help solve problems in genome research (Collis, 1996).

9 REFERENCES

Bandyopadhyay, S., Hughes, J.G., Smith, F.J. & Sen, K. (1994) A Generalised Scientific Information System. *Comp Phys Comm 33*, 49-53.

Bundy, A**.** (1979) Solving Mechanics Problems Using Meta-Level Interface. In Michie, D (Ed.) *Expert Systems in Micro Electronic Age.* Edinburgh, Scotland: Edinburgh University Press.

Collis, J. C. (1996) *An application of artificial intelligence to quantitative problem solving in engineering.* Ph.D. Thesis. The Queen's University Belfast, N. Ireland.

Kaufmann, A. &Gupta, M.M. (1991) *Introduction to Fuzzy Arithmetic : Theory and Applications*. New York: Van Nostrand Rentold.

Larkin, J., McDermott, J., Simon, D.P. & Simon, H.A. (1980) *Expert and Novice Performance in Solving Physics Problems, Science 208* (20), 1335-1342

Mints, G. & Tyugu, E. (1988) The Programming System PRIZ. *Journal of Symbolic Computation 5*, 286-291.

Nilsson J.N. (1980) Chapter 3, *Principles of Artificial Intelligence*. Berlin, Germany: Springer International. Bandyopadhyay, S. & Devitt, J.S. (1987) A Symbolic Information Management System. *Journal of Symbolic Computation, 4*, 397-408.

Novak, G.S. (1977) Representation of Knowledge in a Program for Solving Physics Problems. *Proc. of 5th International Joint Conference on Artificial Intelligence (*pp 286-291).

Press, W.H., Teukolsky, S.A., Vettering, W.T. & Flannery, B.P. (1992) *Numerical Recipes in C: The Art of Scientific Computing,*(2nd Ed.) Cambridge, UK: Cambridge University Press.

Rich, E. & Knight, K.S.(1991) Chapter 3, *Artificial Intelligence (*2nd Ed.) New York, USA: McGraw Hill.

Rumble, J.R., & Smith, F.J. (1988) *Database Systems in Science and Engineering.* Bristol, United Kingdom: Adam Hilger

Tygu, E. (1988) *Knowledge Based Programming.* Workinghom, United Kingdom: Addison Wesley.

Tyugu, E.N. (1991)**,** Knowledge Based Programming Environments. *Knowledge Based Systems 4*(1), 4 -15.

Smith, F.J. & Hughes, J.G. (1985) The Belfast Scientific Database System. In Glaeser, P.S. (Ed.) *The Role of Data in Scientific Progress (*pp 435-437). North Holland, Amsterdam: Springer.

Smith, F.J. & Krishnamurthy, M.V. (1994) Integration of Scientific Data and Formulae in an Object-Oriented Knowledge Based System. *Knowledge Based Systems*, *7*, 135-141.

Stonebaker, M. (1991) The Next Generation Postgress Database System. *Communications of the ACM 34*(10), 78-92.

Welham, B. (1977) *Geometry Problem Solving* Technical Report, Edinburgh, Scotland: Department of Artificial Intelligence, University of Edinburgh.