Int. J. Heat Mass Transfer. Vol. 33, No. 4, pp. 703–723, 1990
Printed in Great Britain

0017–9310/90 $3.00+0.00
Pergamon Press plc

A declarative language for the thermal design of regenerative heat exchangers

M. P. HENRY
Engineering Science Department, Oxford University, Parks Road, Oxford OX1 3PJ, U.K.

and

A. J. WILLMOTT†
Computer Science Department, York University, Heslington, York YO1 5DD, U.K.

(*Received* 9 *November* 1988 *and in final form* 17 *July* 1989)

Abstract—This paper describes some of the difficulties associated with the thermal design of industrial heat exchangers, and advocates a new approach using a declarative style for problem formulation. By way of illustration, a small declarative language is described, and an example design problem is defined in the language and solved using the corresponding problem-solving environment.

1. INTRODUCTION

HEAT EXCHANGER DESIGN (HED) is a composite discipline, combining knowledge and expertise from several fields: heat transfer, mathematical modelling, numerical analysis, and computer science. The principal HED methodology in current use is the well-known 'Case Study Method' devised by Kays and London [1], and described in detail by London [2]. The conventional means of solving a HED problem is the Fortran program. In this paper a new approach to problem formulation and solution is suggested.

Although the techniques described can be applied to all classes of heat exchanger, the regenerator presents particular difficulties and these are discussed here. A regenerator differs from a recuperator in that instead of heat transfer taking place through a wall separating the fluids, each fluid passes in turn over the same surface [3]. This is achieved either by rotating the heat storage material (or packing) between the two fluid streams (rotary regenerators) or by using valves to shut off one fluid before allowing the other to pass over the packing (fixed bed regenerators).

Regenerators present particular design difficulties because:

- Even at equilibrium, fluid and solid temperatures vary both spatially and chronologically, whereas in a recuperator the temperature varies only in space. The mathematical models and corresponding simulation methods are thus more complex for regenerators.
- A regenerator packing typically has an irregular geometry to maximize heat transfer; consequently heat transfer coefficients (htc) must in general be determined experimentally.
- Although there are few mathematical models of regenerators, they are employed in a wide range of application areas, from cryogenics to ventilation systems to blast furnaces. There is correspondingly great diversity in design variables and constraints, packings, geometries, fluids and operating conditions, as described by Shah *et al.* [4].

The chief means of communicating HED advances is through the scientific literature, and papers concerned with regenerators fall into two main groups: those that deal only with the modelling and simulation of regenerators, and those that describe complete design calculations.

The earliest simulation schemes were severely limited by the absence of digital computers: Iliffe [5] reports having spent some 10 hours using a slide-rule and five-figure tables to perform a single regenerator simulation, and it is a little-known fact that Hausen, during the 1920s and 1930s, 'programmed' his entire family to perform regenerator calculations. Correspondingly, the design calculations were simple and could be completely specified within a single technical paper as a 'design manual'. It was straightforward to obtain a state-of-the-art design method: a design manual in a technical journal could be modified to accommodate the user's particular requirements with reasonable ease. The difficulty lay in performing the calculation.

Today the reverse is the case. Workstations can perform elaborate simulations within seconds or less. However each of the fields from which HED draws its skills has expanded and become increasingly specialized: no single paper can fully describe the latest in regenerator design theory. Consequently, it requires many man-months to create software that is state-of-the-art in each aspect of HED, adequately embodies user requirements, and exploits available computational power. As engineers have deadlines, and

† Author to whom all correspondence should be addressed.

the resources for software development are rarely available, current regenerator design programs are often a poor reflection of available HED theory.

1.1. *Design manuals and design programs*

The first published design calculations took the form of design manuals, consisting of complete step-by-step methods, normally including example calculations. Graphs and tables gave thermophysical properties and efficiency vs parameter data from simulation calculations. The schemes were open-ended: the user decided which variables to assign values and which to leave free for optimization, within the limits of available computational resources. As the design method (and often its derivation) was given in full, it was easy to modify the design to satisfy a user's requirements (e.g. add extra constraints or variables). This was necessary, because the design variables and constraints used by different engineers varied widely even for the same application area. For example, both Razelos and Paschkis [6], and Butterfield *et al.* [7] provide complete design calculations for Cowper Stoves, based upon the same Pseudo-recuperator model of regenerator performance [8], but using very different 'real-world' variables. A thorough comparison of the two manuals can be found in ref. [9]. Given the expense of physical experimentation and the lack of computational facilities (in the discussion at the end of Razelos and Paschkis [6], an engineer reported spending three working days on a single calculation), discrepancies in design theory were inevitable.

The development of finite difference schemes such as the Lambertson method [10] heralded the dependence of simulation calculations upon computers. Finite difference techniques use simple equations evaluated hundreds of times: a computational load feasible only using a computer. It was natural to extend automation to complete design calculations. The resulting programs had significant advantages over design manuals:

- they eliminated much of the computational burden on the engineer;
- they could perform more precise calculations than was possible with table, graph and slide-rule;
- they did not suffer from the parameter limitations of graphs and tables;
- speed of calculation no longer presented a barrier, so that more ambitious design schemes involving more variables, fewer assumptions, and numerical optimization, became practicable.

However, the changeover from the design manual to the program as the chief vehicle for HED has introduced problems.

- A design manual is compact, containing only essential equations, graphs and tables needed for the calculation, whereas a Fortran program to perform the same calculation requires housekeeping code, input and output statements, etc., and so is considerably longer. Complete design manuals can be published in full, together with example calculations, whereas complete design programs are too large for technical journals. Program design is merely outlined, so the engineer must often write his own code. Clearly, the more sophisticated the design scheme, the greater the effort entailed.
- Fortran programs are inflexible: a design manual allows the user to leave any of the independent variables unassigned for optimization, but a Fortran program cannot do this, as the equations are fixed in code. With ingenuity, some flexibility is possible (see, for example, Palen *et al.* [11]), but this is fairly limited.
- It is more difficult to modify a program to satisfy new requirements than it is for a manual, as the program will typically need to undergo updates to the input/output code, interface to the optimization routines, hard coded data and other areas, requiring detailed knowledge of the program structure.

Of course, if the engineer has access to a program that already satisfies his requirements, then there is no need to write or modify any code. Again we distinguish recuperators and regenerators, for whereas there are several commercially available programs for recuperator design (e.g. HTFS's Thermech [12]), the authors are unaware of any commercial program for the design of regenerators.

Despite these difficulties, the application of computers to HED has proved an enormous step forward, and has been one of the most significant catalysts in the development of HED theory.

1.2. *Advances in aspects of heat exchanger design*

The last 30 years have seen major advances in all aspects of HED. Many of the factors ignored in the earliest regenerator models can now be included, such as fouling, maldistribution of flow, longitudinal conduction, radiation, transient performance, non-linearities and reversal effects [13, 4]. It is now possible to obtain full chronological and spatial variations in fluid and solid temperature both for transient and steady-state behaviour. There has been a steady improvement in stability, parameter ranges and speed as new methods of solution are developed for the same mathematical model (see, for example, Hill and Willmott [14]). Where once point values were used for specific heat, conductivity, density and other properties, new correlations incorporating temperature dependence are available for many commonly-used materials such as the atmospheric gases. Optimization has become a specialist subject in its own right, providing entire families of solution methods. As no single optimization algorithm is guaranteed to locate the optimum point for an arbitrary HED problem, Shah *et al.* [15] recommend that a suite of algorithms be available for use.

It is clear, therefore, that each of the areas from which HED theory draws its expertise has grown so large that it is impossible for any single individual to be fully conversant with every aspect. If current practice continues, difficulties in the communication and application of expertise can only worsen as the body of knowledge grows, and the situation is developing into what the authors call the software crisis in HED.

1.3. *The current crisis*

The current crisis has several features.

Communication of design expertise. The communication problem is well illustrated by the paper of Hedderich *et al.* [16], subsequently reviewed by Rubin [17]. Hedderich *et al.* describe a program to design optimal air-cooled heat exchangers for marine applications. Rubin criticized the paper for ignoring fouling and maldistribution effects, for the ranges of parameters used in the optimization studies, and the cost function employed, concluding that the program was unusable for commercial design. Needless to say, the authors defend their approach.

With neither the code described in the paper nor a code that would satisfy Rubin, and with insufficient information to be able to replicate Hedderich *et al.*'s program or to incorporate those factors deemed necessary by Rubin, a third party can learn very little from this exchange other than that HED is a highly contentious issue!

The communication problem has been recognized in fields related to HED. Spalding, in the editorial of the first issue of the *PHOENICS Journal of Computational Fluid Dynamics*, notes [18]

"How frequently, having read a careful description of a new solution algorithm, can the reader implement it on his own?... There is simply not enough space in conventional journals for recording what is necessary for complete transfer (of skills)... As a consequence, editors, authors and readers have learned to be content with the conveyance of mere outlines of the work performed... and what the author has lacked space to supply the interested reader has had to supplement by guesswork."

Continued development and use of unsophisticated design methods. This is well illustrated by the relatively recent (1982) paper of Kulakowski and Schmidt [19], in which they describe a method for engineers to produce their own design manual and design curves, incorporating their own design variables, pressure drop and heat transfer correlations and material properties. By way of example, design curves for three different types of regenerators are developed.

This scheme suffers from the same types of drawback as other design manual schemes: the method only deals with a limited range of regenerator parameters, only the gross efficiencies are provided rather than detailed temperature profile information, if the data changes or unusual parameters arise then the curves must be recalculated, precision is limited, optimization is not possible, and so on. However, it represents an apt and ingenious way of incorporating user-specific data into a design scheme without the need for elaborate programming, and its publication indicates the continued interest in elementary simulation and design schemes despite the availability of much more accurate and informative methods.

Code fossilization. A major problem arising when large programs are used for HED is that expertise may become fossilized in computer programs which were written by employees or consultants with whom the company no longer has any contact. Indeed current expertise may consist of how to use such a computer program to generate a reliable design for a heat exchanger, together with only a partial knowledge of how the program works. This partial knowledge will include, typically, information about experiments to evaluate heat transfer correlations, and more importantly, how previous computer aided designs have turned out in practice when new heat exchangers have been installed for customers. Attempts to update the code are fraught with danger, and many companies prefer to make do with the known inadequacies of the current program.

New developments in computer science. A most significant advance in recent years has been the development of parallel processing devices such as the Transputer, capable of 1.5 MFLOPS per processor, and which can be connected by the dozen to workstations and PCs. Phenomenal computational power will be available on the desktop in the near future, but it can only be exploited if the current serial simulation and optimization codes are rewritten to use parallel hardware.

Developments in the area of Artificial Intelligence offer the opportunity to incorporate the qualitative aspects of design into the program. This is seen by some to be most necessary:

"A practical need for storing knowledge is seen in the heat transfer industry as many experts from the great golden age of process heat transfer are retiring, perhaps without sufficiently passing on their expertise to the younger engineers [20]."

However, the introduction of qualitative factors appropriate for a particular commercial concern into a program makes it less comparable to other codes, thus adding to the problems of verification.

Both workstations and PCs now routinely provide advanced features to assist the user, including windowing, mice, pull-down menus and sophisticated graphics.

All these desirable features are achieved only by adding further dimensions of programming complexity. Existing programs will rapidly appear obsolete in the face of these advances but the task (if undertaken) of upgrading the code is formidable. Given that each sophisticated design program is essentially unique, such updating represents at best a gross duplication of effort within the HED community, and at worst an unparalleled opportunity for new errors to enter the code.

1.4. *The future of HED*

Paradoxically, the future of HED appears both exciting with the advances being made in so many aspects of the field, and yet bleak when considering how to implement and verify programs containing these advances. Jerry Taborek, an experienced designer, voiced his fears at a conference as far back as 1977. Having described some of the advances he expected, he continued:

> "Many of the above items will not necessarily constitute progress in any real sense. Reliance on computer results will become progressively more common, as there will be a lack of experienced designers. This is particularly frightening, as everyone knows that no computer program of the magnitude we are dealing with can ever be completely checked out. Errors in general and even those involving fundamental principles will be undetected. To remedy this, emphasis will be placed on increased optimisation by the computer logic, thus further complicating the programs. To break this vicious circle, we will have to eventually dust off Kern's Process Heat Transfer [21] and again start educating engineers how to design heat exchangers [22]."

Some of the points raised here will be returned to later.

As an alternative vision of the future, we pose the following question. In the late 1990s, will an engineer still be using approximate graphical methods (such as those described by Kulakowski and Schmidt [19]), when the 50 MFLOPS workstation on his desk could calculate an optimized and fully simulated design in the time it takes to read a single value from the graph, if only the appropriate, verified expertise were available to him in program form?

1.5. *Summary*

To summarize this introduction, all aspects of the theory and technology employed in HED are expanding rapidly; conversely, it is becoming increasingly difficult to communicate that expertise due to the increasing degree of specialization needed to master any one of these areas, and the limitations of the technical paper as a means of conveying program detail. The gap between the potential and actual sophistication of HED programs is large now, and is getting larger.

What is needed to halt this trend, therefore, is a medium in which design theory and data can be conveyed in compact and yet complete form; a medium through which the best simulation, optimization and thermal property codes can be readily shared; a medium through which new developments can be completely specified and thoroughly evaluated; and yet a medium which is sufficiently flexible to allow every user to freely and easily modify any of this information to suit their own particular requirements.

Such media have been created for other specialized fields such as control and computational fluid dynamics: they are Problem Solving Environments (PSEs) and associated declarative languages.

2. PROBLEM SOLVING ENVIRONMENTS

2.1. *Definition*

A PSE is a general program designed to solve a range of problems in some specialized domain. It assists the user by providing graphical, data saving and logging facilities, as well as a suite of inbuilt functions relevant to the problem domain. It also provides one of more methods of 'solution', possibly simulation, optimization or symbolic manipulation techniques as appropriate for the field concerned. Problems are stated in an appropriate declarative language in terms that are familiar to the practitioner. The language is 'declarative' in that the problem is merely stated: the PSE contains the means of solution. Typically, the user operates the PSE interactively, either typing in the problem definition or supplying a file name, and then monitoring and guiding progress towards a solution. Repeated calculations allow parameter studies to be performed with ease.

Examples of PSEs include PHOENICS [18] for computational fluid dynamics, Simnon [23] for simulating dynamical systems such as in control or biology, DELIGHT [24] for solving optimization problems and developing algorithms, Ctrl-C [25] for solving matrix-based problems, and Computer Algebra systems such as MACSYMA [26].

PSEs offer several advantages.

- It is unnecessary to write a complete program. Instead, the problem is stated in a suitable notation. This greatly reduces human effort between problem formulation and solution.
- The facilities provided by the PSE are written by a single concern but used by many. This elimination of duplicated effort makes viable the incorporation of state-of-the-art software in the PSE. Updates of the PSE, developed by specialists, may be released to exploit the latest hardware and software advances. However, as the declarative language remains unchanged the user enjoys a high degree of portability for his problem definitions.
- The problem definitions are concise pieces of domain knowledge, and as such suitable for communication in the literature. In setting out a justification for the PHOENICS journal, Spalding states [18]

> "An enormous amount of communication has been effected between the author and the readers (already)... having access to the same software... the language used for defining the particular computations... is a compact one... It therefore takes little space to convey, in every last detail, just what an author has done: and the user... can reproduce his results exactly."

PSEs are thus excellent for tutorial purposes. For example, in Åstöm and Wittenmark's forthcoming textbook on adaptive control, every example is generated using Simnon [27]. Any student with access to the program can replicate and further investigate each example until it is thoroughly understood.

2.2. *A PSE for HED*

From the above description of the features of a PSE, it is straightforward to list facilities to be provided by a PSE for HED. These include:

● functions performing simulation and optimization, and for calculating the thermophysical properties of matter, as well as facilities to interface with user-defined code.

● a declarative language allowing the user to define as freely as possible design variables, constraints, data and optimality criteria;

● a sophisticated user-interface to aid in the exploration of problems, including help, problem saving/restoring and graphical support.

Such facilities would overcome many of the problems currently faced by HED engineers. The need to write code would be dramatically reduced. As problem definitions are reduced to a concise form, their communication in technical papers becomes possible (as is demonstrated later). Moreover, reported results could be reproduced by anyone using the PSE, allowing detailed analysis and evaluation of design schemes. Flexibility is inherent in that the problem definitions are stated in the declarative language and are therefore data, not code, and so can be easily modified. Thus an engineer might not only replicate a new design scheme, but also evaluate the scheme with his own data such as htc/pressure drop correlations inserted, or, conversely, incorporate elements of the published scheme into his own problem definitions. Such improvement in communication might well lead to a greater consensus concerning the role and importance of various factors (e.g. fouling, maldistribution) in HED.

We now consider how the widespread use of such an environment might allay some of the fears expressed by Taborek (quote, Section 1.5). Firstly, although indeed no program of the complexity envisaged here can be proven error free, it is a significant step forward to have one program written by specialists and with a large userbase rather than many programs written less expertly and with much smaller user-bases: the probability of error detection, reporting, and correction is much higher in the former case and, of course, much less code is written.

Secondly, HED theory, no longer obscured by the complexity of the program, is placed once more in the foreground of the designer's attention. This will hopefully eradicate many of the errors involving fundamental principles which concerned Taborek, and indeed a PSE should prove a useful educational tool. One can envisage a new generation of texts (descendents of Kern's Process Heat Transfer) in which the theory is stated in a declarative language, enabling the student or engineer to experiment with equations to gain a deeper understanding. In effect, we readopt the clarity of the design manual but retain the computational power and sophistication of the program. We have the best of both worlds.

3. DEVELOPMENT OF AN EXPERIMENTAL SYSTEM

An experimental package has been written providing a declarative language, GOLD, and a PSE, GNOSYS, to solve HED problems. Two goals were defined.

(1) The same program should be able to solve design problems for entirely different types of heat exchanger, the only change being the problem definition stated in the declarative language.

(2) The language should demonstrate sufficient flexibility to enable an industrial concern to incorporate its own design data, constraints and optimality criteria, again without the need to alter the program code.

A detailed description of the philosophy of package design is not possible here; it is provided in ref. [9]. Some key decisions are, however, summarized below.

The 'method of solution' (Section 2.1) is optimization, so that all design tasks are stated in the form of constrained optimization problems. Indeed as, roughly speaking, the set of all HED problems have very little in common with each other (in terms of universally-used variables or dimensionless parameters), the declarative language GOLD allows quite arbitrary optimization problems to be defined (this further enabled the system to be tested using standard optimization problems). However, HED-specific functions are incorporated to perform simulations and to calculate thermophysical properties.

The technique of optimization, where the user can select both the cost function and the set of independent variables, offers the greatest opportunity for exploiting HED expertise held in a declarative form, as the same design scheme can be used to solve four related problems:

Simulation. For a known heat exchanger, predict its performance for known operating conditions. This problem, the easiest of the four, does not require optimization, but merely evaluation of the performance variables.

Control. For a known heat exchanger, optimize certain operating condition variables in order to match required performance. The cost function may be either simply a least squares minimization on the required performance, or alternatively a measure of operating cost, with the performance ensured by placing tight constraints upon appropriate variables.

Design. For a known performance requirement, optimize the design of the heat exchanger, subject to a variety of constraints, minimizing some measure of cost. This problem has already been discussed at length.

Tuning. Given the actual performance of a known heat exchanger, optimize certain parameters of the model (e.g. measures of maldistribution and fouling) to match that performance. A least squares cost function is employed.

The four problems are closely connected, and in particular the ability to tune models is vital, as no manufacturer will adopt a new design scheme or a new software system, whatever predictive advantages are promised, if the performance of proprietary heat exchangers cannot be predicted as least as accurately as by the design scheme and software currently in use. Once a PSE is acquired, then new design schemes, proposed in-house or taken from the literature, can be declared and tuned to see how well they match actual data. In this way a rapid assessment of the usefulness of design schemes can be made by each manufacturer.

Given that almost arbitrary optimization problems can be defined, there arises the question of which optimization algorithm(s) to use. In view of robustness requirements, Box's Complex Method [28] was selected as the first algorithm to be implemented. This is a direct search method which makes no assumptions about the problem domain (for example that the search space is convex, or that the constraints have derivatives). It has in the past been used to solve many industrial problems, including HED [11, 29, 30], but has largely been replaced by more efficient and accurate algorithms. However, its very insensitivity renders it extremely robust and hence suitable for a general and experimental system. A thorough investigation of the properties of the Complex Method and its variants is described in ref. [9]. More efficient algorithms can be added in later versions, although the Complex Method might still have a role as a backup in circumstances where other algorithms failed [31].

A full description of the GOLD/GNOSYS package is not possible here: the interested reader is referred to Appendix 1 of ref. [9], which contains a user manual for the system. The following sections outline the salient features of each to aid understanding of the example calculation.

3.1. *GOLD*

GOLD is a declarative language designed to allow straightforward definitions of design optimization problems. A file containing the problem definition is created using a standard editor, and then the GOLD parser is invoked to prepare a condensed version of the problem for use with the interactive system.

A GOLD file consists of the name of the problem, an optional list of constants (the USE statement), a list of variables with their upper and lower bounds (the VARIABLES statement), a list of equality constraints (the CONSTRAINTS statement) and, optionally, the expression to be maximized or minimized.

It is also possible to create subproblem modules in GOLD. These are similar in content to normal problems, except that they can only be loaded into GNOSYS on top of an already loaded problem definition. This feature is of particular use in a design context, where a series of design options (e.g. different heat exchanger packings) are represented by a set of subproblem modules, each reflecting the necessary alterations (in the form of additional constraints and/or variables) to the original design problem.

A subproblem must name in its header the problem module to which it must be attached: it is then able to refer to identifiers in the parent module without redeclaring them. Subproblems may also be attached to other subproblems to form hierarchies of arbitrary complexity.

3.2. *GNOSYS*

The GNOSYS interactive system provides a robust and flexible environment for the investigation of design optimization problems. A loaded problem can be modified and optimized repeatedly in a single GNOSYS session, and the system incorporates ten *Solution Registers* which store starting points and optimal points. By default register 0 is used for all commands, and will be used in all examples. A summary of the basic GNOSYS commands is given below.

load(*sub*)*problemname*	read the named GOLD (sub)problem file from the current directory.
help	give help information.
detail {*1*...*5*}	adjust level of information given in all messages : 1 = concise, 5 = verbose.
list	output contents of Solution Register.
fix	accept one or more new constraints of the form *independent-variable = expression*.
maximize/minimize *expression*	set cost function to *expression*.
optimize	optimize the current problem definition using the given starting point, supplying intermediary reporting as indicated by the current **detail** level, and return the best point found to the Solution Register.
quit	end GNOSYS session.

The process of optimization involves several levels of searching. Firstly the current problem definition is examined, and any spurious variables (such as those with fixed numerical values and those independent variables which effect neither the cost function nor any dependent variables) are excluded from the optimization process. The start point for the Complex Algorithm is taken from Register 0 (by default). However, this point may not be feasible, and indeed may even be *illegal*, that is, may generate an arithmetic exception or a parameter range exception for a function call. Thus the algorithm must first check (and if necessary search for) legality, and then feasibility, and finally optimality.

4. A PACKED BED PROBLEM

This section demonstrates how the GOLD/GNOSYS package can be applied to HED, by developing a series of GOLD modules used to solve a packed bed problem. Several points need to be made at the outset, however. The problem definition which follows is intended to be a demonstration of both the package and the methodology advocated in this paper. It is not an attempt to define a state-of-the-art packed bed design scheme. Having said this, the example is not trivial: it deals with a real industrial problem using typical data, and as such involves a large number of variables and constraints. The package, though experimental, is capable of handling realistic problems. Space limitations prevent a detailed explanation of the two-dimensional linear regenerator model applied here, or indeed aspects of the packed bed design. Readers unfamiliar with the former may consult any text on regenerators (e.g. Schmidt and Willmott [29]), and the latter is developed by Cutland [33]. In any event, it is hoped that where readers are unfamiliar with the equations involved, they may appreciate the techniques being used and consider how they may be applicable to their own domains of interest.

In the HED context, a problem module is used to define the mathematical model to be used in the design, by declaring a call to appropriate simulation routine and by defining appropriate variables. Problem module Reg2D (Fig. A1) defines an entirely general two-dimensional linear model of a thermal regenerator, calling the inbuilt function *Hill–Willmott* which uses the improved Hill–Willmott scheme [32] to calculate the cold side thermal ratio, *cold-eta-reg*. The declaration is best explained by demonstrating what happens when this module is loaded into GNOSYS. Figure A2 shows the effect of loading and listing this module. The independent variables are the inlet temperatures and the dimensionless parameters. From these the thermal ratios and outlet temperatures are calculated. No cost function is defined.

Although this problem module may be used in isolation, its very generality limits its usefulness. However, subproblem modules may be defined to transform this general declaration into something more specific. Herein lies the strength of a modular approach, for different subproblems may be defined to modify the original definition in different ways. Given that the Hill–Willmott simulation routine contains several hundred lines of code, the final product of several man-months of effort, it is clearly desirable to facilitate its use in as many appropriate design problems as possible.

Subproblem module Packed-Bed (Fig. A3) transforms Reg2D into a packed bed problem by introducing new variables and constraints. It is based on a model of the York University Rig, developed by Cutland [33], from which a detailed explanation can be obtained. The module introduces variables for the bed geometry (a cylindrical bed of spheres), the gas properties on each side and the packing properties. The constraints define some of these variables in terms of others, and, most importantly, define the dimensionless parameters (independent in Reg2D) in terms of the packed bed variables, as is shown in Fig. A4. Listing the problem definition after loading shows that the dimensionless parameters have become dependent.

Many modules could be defined for different heat transfer correlations: module Denton (Fig. A5) introduces the Reynolds and Prandtl numbers and defines the htcs according to Denton's analysis [34]. Similarly, module Ergun (Fig. A6) defines the hot and cold side pressure drops using correlations developed by Ergun [3].

The thermophysical properties of gases can be calculated within GNOSYS by calling a set of inbuilt functions. Each routine, largely based on code developed by Brooks [35], takes as parameters the relative proportions of 'air', nitrogen, oxygen, carbon dioxide and water vapour, together with the temperature at which the particular property is to be calculated. While this falls well short of the ideal, namely, a suitable interface to a state-of-the-art thermophysical property library such as that of HTFS [36], the functions are sufficient for demonstration purposes.

Again, several possible gas property subproblems may be defined. Module hot-gas-comp, for example, (Fig. A7), introduces variables for the relative proportions of nitrogen, oxygen, carbon dioxide and water vapour and calculates the hot side gas properties accordingly. This module might be used when the hot gas consists of combustion products of known composition. Module cold-air, on the other hand (Fig. A8), assigns the cold side gas properties to be those of air. These vary with temperature. However, the two-dimensional linear model assumes that the fluid properties are constant throughout each period. Both modules therefore calculate gas properties at the 'average' temperature in the regenerator, which is taken to be the mean of the hot and cold inlet temperatures. Thus the properties are invariant on each side, and depend only upon the inlet temperatures. This simplification is a consequence of the use of the linear model.

So far, the modules that have been described are general rather than specific: no cost function has been defined, no packing properties specified, and many variables remain independent. The next section demonstrates how the design requirements of a specific industrial concern can be incorporated into the problem definition.

4.1. *Subproblem Module Client-1*

A packed bed manufacturer was approached and asked for design data and an example problem to solve. The data included in the response has been incorporated into the subproblem module Client-1, which is given in Fig. A9.

New variables are introduced for the minimum cold outlet temperature, ***required-cold-out***, and the difference between this minimum value and the predicted value, ***cold-out-diff***. The effect of these two variables is to ensure that any design in which the predicted cold outlet temperature ***cold-outlet-temp*** is less than ***required-cold-out***, is infeasible. Thus the user need only assign a value to ***required-cold-out*** during a GNOSYS session for a minimum requirement to be placed on the cold outlet temperature.

The client placed a maximum pressure drop on each side of between 5 and 15 in. of water. To meet this requirement new variables ***hot-water-drop*** and ***cold-water-drop*** are introduced, together with a conversion factor from Pascals to inches of water. The user may modify the upper bounds of these variables during a GNOSYS session to specify particular limits.

Data is given for the packing material to be used: spheres of fixed diameter, density and voidage, and with a temperature-dependent specific heat.

The cost function is also specified at this juncture: the mass of the packing is to be minimized, while meeting both heat transfer and pressure drop requirements.

A typical design problem for the client can be stated as follows. Use the data from modules Reg2D, Packed-Bed, Denton, Ergun, and Client-1, and the following particular requirements:

Note that not all this data need be incorporated into subproblem modules, as constraints but may be defined interactively. Thus the user may easily change values of, say, the required minimum cold outlet temperature or the period lengths, and so perform case studies.

Figure A10 shows a GNOSYS session in which this design problem is solved for the case where the minimum cold outlet temperature is 950°C. Firstly, all the required GOLD modules are loaded. Then particular constraints (in the form of numerical constants) are assigned to more of the design variables, and finally the optimization routine is called.

Note that in the listing prior to optimization, there remains only two independent variables, the bed length and the bed radius, and that many dependent variables are already assigned numerical values. These are eliminated from optimization, as is indicated by the initial report of the optimization routine, given that their values are unaffected by any change in the independent variables. The solution delivered by the optimization routine clearly satisfies both pressure drop and heat transfer requirements (see the values of the variables ***cold-water-drop***, ***hot-water-drop***, ***cold-outlet-temp*** and ***cold-outlet-diff***) and the values of all other variables are listed for inspection.

To perform a case study, all that is necessary is to change the interactively-defined constraints (e.g. required cold outlet temperature). Larger changes may be made by replacing subproblem modules. The cost function itself can be changed to an arbitrary expression using a single command. A parameter study of this problem, in which the required cold outlet temperature and gas compositions are varied, and the effects of using different pressure drop correlations are compared, can be found in ref. [9]. By way of illustration, Fig. A11 shows the optimal sizes of regenerators for cold outlet temperatures ranging from 950 to 1350°C.

4.2. *Appraisal*

The GOLD/GNOSYS system has been developed to display some of the desirable features of a PSE for HED.

hot fluid inlet	= 1400°C	cold side gas	air
cold fluid inlet	= 20°C	hot side gas	N_2 72.2%
hot mass flow rate	= 1.246 kg s^{-1}		O_2 2.3%
cold mass flow rate	= 1.122 kg s^{-1}		CO_2 8.5%
hot period length	= 600 s		H_2O 17.0%
cold period length	= 600 s		
max. hot pressure drop	= 5 in. of water		
max. cold pressure drop	= 5 in. of water		
average cold outlet temp	= 950, 1000, . . . , 1350°C		
Cost function: *minimize the mass of the packing.*			

● The GOLD language allows sets of data modules to be defined from the very general to the highly specific, incorporating user-specific data, constraints and cost function. This requires no knowledge of, or modification to, the code of GNOSYS. Many of the modules can be used repeatedly, with requirements specific to each design problem being defined interactively.

● Although there is insufficient space to present another example, it is hopefully clear that entirely different HED problems may be solved using the package, and the two goals defined in Section 3 have been achieved.

● GNOSYS is highly robust. Optimization is possible in search spaces containing singularities or entire regions of exception-generating points.

It is also clear, however, that many features could be added to enhance the power and scope of the present system.

● The GOLD language is fairly restrictive: the only data type is the real value, whereas integer, vector and even array types would be useful. Constraints are currently limited to simple expressions, but IF/THEN and macro/procedural constructs would greatly enhance the language, as would automatic interpolation of tabular data (e.g. actual performance data used for tuning purposes).

● Although any simulation code may be interfaced to the package, currently only single real values are returned, whereas it would be useful to return vectors of temperature profile information. Also, it is difficult to communicate across to simulation routines data concerning nonlinearities (e.g. temperature-dependent properties).

● More accurate, efficient and less robust optimization methods must be added to the system, together with a corresponding extension of user commands to select, monitor and change the algorithms.

● The user-interface could benefit from several additional features: an intelligent editor, graphics, windowing and mouse-based commands are all applicable to the package. Larger listings might be broken down by module for easier comprehension, and sensitivity analysis could be provided.

Many of these items are a matter of implementation. For example, a variant of the package has been developed in which the user is able to state in equational form the non-linear behaviour of any variable, which is used in the simulation calculation. This system is described in ref. [9], and will be the subject of a future paper. Also, such features as the graphical display of temperature profile information returned from simulation routines are included in the specification of the next version of the GOLD/GNOSYS package.

5. CONCLUSION

This paper has described the features of what is perceived as the growing software crisis in HED, and has proposed, and demonstrated, a declarative style of problem formulation. The successes and limitations of the GOLD/GNOSYS package have already been discussed. We conclude by considering the impact of introducing PSEs such as GOLD/GNOSYS into the HED domain.

If such systems can provide satisfactory design solutions for many users, then the potential benefits are enormous. Despite language limitations, the declarations in GOLD modules may certainly be viewed as concise definitions of HED knowledge, stated in a design manual style, and they can be applied, combined and modified with maximum flexibility by the GNOSYS program. The only aspects of HED theory still residing in the program itself are the simulation, optimization and thermal property codes which are best written by specialists and are relatively easy to test and verify in isolation. The user need only rarely resort to writing code, and then only to interface some specialized routine to the PSE. The focus therefore returns to the HED theory itself: if the reader wishes to criticize any aspect of the packed bed example, he is able to do so precisely because of the clarity with which the entire design calculation is stated, and it is likely that any criticism will be satisfied by a rewriting of some of the GOLD modules. The principle of communicating a complete design scheme in a technical paper, such that results can be replicated, has clearly been demonstrated, and the advantages this entails have already been discussed.

A PSE should enable users to share expertise and take advantage of advances in every area of HED theory and computer science. Advances in optimization, simulation and thermophysical property calculation can be incorporated into new versions of the PSE without inconveniencing the user, who need only be informed of some additional functions available in the declarative language or PSE. The declarative language itself is highly portable, GOLD/GNOSYS has been successfully installed on several manufacturers' computers and GOLD modules are entirely compatible between systems. If the expertise is available to port a PSE onto an advanced parallel architecture, then many users can benefit (indeed, the PHOENICS system has recently been ported onto PC-based transputer systems).

It is not anticipated that there will be a revolution in HED practice overnight, but rather a steadily growing acceptance that a declarative style of problem formulation represents a better way of preserving, exploiting and communicating HED expertise. However, the issues raised in this paper must be addressed with some urgency if the future of HED is to offer unparalleled opportunity and not merely daunting, and increasing, complexity.

REFERENCES

1. W. E. Kays and A. L. London, *Compact Heat Exchangers*. McGraw-Hill, New York (1964).
2. A. L. London, Low Reynolds number flow heat ex-

changers. In *Compact Heat Exchangers—Design Methodology* (Edited by S. Kakac, R. K. Shah and A. E. Bergles), pp. 815–844. Hemisphere/Springer, New York (1983).

3. H. Hausen, *Heat Transfer in Counterflow, Parallel Flow and Crossflow.* McGraw-Hill, New York (1976).
4. R. K. Shah, B. T. Kulakowski, A. J. Willmott, P. J. Heggs and I. L. Maclaine-Cross, Advances in regenerator design theory and technology. ASME Winter Annual Meeting, Boston, November (1983).
5. C. E. Iliffe, Thermal analysis of the contra-flow regenerative heat exchanger, *J. Inst. Mech. Engng* **159**, 363–372 (1948).
6. P. Razelos and V. Paschkis, The thermal design of blast furnace regenerators, *Iron Steel Engr* 81–116 (August 1968).
7. P. Butterfield, J. S. Schofield and P. A. Young, Hot blast stoves (part III), *J. Iron Steel Inst.* 497–508 (June 1963).
8. H. Hausen, Improved calculations for heat transfer in regenerators, *Ver. Deut. Ing.* **2**, 31–43 (1942).
9. M. P. Henry, Design methodology: regenerative heat exchangers. Doctoral Thesis, Department of Computer Science, University of York (1987).
10. T. J. Lambertson, Performance factors of a periodic-flow heat exchanger, *Trans. ASME* **159**, 586–592 (1958).
11. J. W. Palen, T. P. Cham and J. Taborek, Optimisation of shell-and-tube heat exchangers by case study method, *CEP Symp. Ser.* **70**, 205–214 (1974).
12. Heat Transfer and Fluid Flow Service, *THERMECH—Thermal and Mechanical Design of Shell and Tube Heat Exchangers.* Harwell Laboratories, Oxford (1987).
13. H. Hausen, Developments of theories on heat transfer in regenerators. In *Compact Heat Exchangers—History, Technological Advancements and Mechanical Design Problems* (Edited by R. K. Shah and D. E. Metzger), pp. 79–89. ASME (1980).
14. A. Hill and A. J. Willmott, A robust method for regenerative heat exchanger calculations, *Int. J. Heat Mass Transfer* **30**, 241–249 (1987).
15. R. K. Shah, K. A. Afimiwala and R. W. Mayne, Heat exchanger optimisation, *Heat Transfer* **4**, 185–191 (1978).
16. C. P. Hedderich, M. D. Kelleher and G. N. Vanderplaats, Design and optimisation of air-cooled heat exchangers, *J. Heat Transfer* **104**, 683–690 (1982).
17. F. L. Rubin, Design and optimisation of air-cooled heat exchangers (comment), *ASME J. Mech. Transmissions Aut. Des.* **106**, 256 (1984).
18. D. B. Spalding, Editorial, *PHOENICS J. Comput. Fluid Dyn. Applic.* **1**(1), 1–7 (January 1988).
19. B. T. Kulakowski and F. W. Schmidt, Explicit design of balanced regenerators, *Heat Transfer Engng* **3**, 25–37 (1982).
20. J. W. Palen, Designing heat exchangers by computer, *Chem. Engng Prog.* 23–27 (July 1986).
21. D. Q. Kern, *Process Heat Transfer.* McGraw-Hill, New York (1950).
22. J. Taborek, Evolution of heat exchanger design techniques, *Heat Transfer Engng* **1**, 15–29 (1979).
23. K. J. Åstöm, A Simnon tutorial, Department of Automatic Control, Lund Institute of Technology (1985).
24. W. Nye and A. Tits, Delight for beginners, Memo UCB/ERL M82/55, Electronic Research Lab, University of California (1982).
25. Systems Control Technology, Inc., *The Ctrl-C Users Guide.* Palo Alto, California (1986).
26. W. A. Martin and R. J. Fateman, The MACSYMA system. *Proc. Second Symp. Symbolic and Algebraic Manipulation,* Los Angeles, California, pp. 59–75 (1971).
27. K. J. Åstöm, Private communication (May 1988).
28. M. J. Box, A new method of constrained optimisation and a comparison with other methods, *Comput. J.* **8**, 42–52 (1965).
29. F. W. Schmidt and A. J. Willmott, *Thermal Energy Storage and Regeneration.* McGraw-Hill, New York (1981).
30. S. Kobayashi, T. Umeda and A. Ichikawa, Synthesis of optimal heat exchanger systems, *Chem. Engng Sci.* **26**, 1367–1380 (1971).
31. M. A. Townsend and C. E. Zarak, Accelerating flexible polyhedron searches for nonlinear minimisation, *ASME J. Mech. Transmissions Aut. Des.* **105**, 197–200 (1983).
32. A. Hill and A. J. Willmott, Accurate and rapid thermal regenerator calculations, *Int. J. Heat Mass Transfer* **32**, 465–476 (1989).
33. N. G. Cutland, The unsteady-state performance of an experimental thermal regenerator, D. Phil. Thesis, Department of Computer Science, University of York (1984).
34. W. H. Denton, C. H. Robinson and R. S. Tibbs, The heat transfer and pressure loss in fluid flow through randomly packed spheres, Harwell Report (UKAEA) AERE-R 4346 (1963).
35. S. Brooks, Computer simulation of high temperature thermal regenerators. D. Phil. Thesis, Department of Computer Science, University of York (1984).
36. Heat Transfer and Fluid Flow Service, Design Report DR40—Physical Properties. Harwell Laboratories, Oxford (1987).

APPENDIX

```
PROBLEM Reg2D;

{This problem module defines a 2D linear Thermal Regenerator
Modelling Package. Given the dimensionless parameters and the
inlet temperatures of the Regenerator, it calculates the mean
outlet temperatures. The model uses the improved Hill-Willmott (1989)
method of regenerator simulation }

USE
abs_zero            =  -273.16;            {minimum temperature - Centigrade}

VARIABLES

Hot_Reduced_Length   [0 : 100];            {Dimensionless Parameters}
Cold_Reduced_Length  [0 : 100];

Hot_Reduced_Period   [0 : 30];
Cold_Reduced_Period  [0 : 30];

Gamma                [0.001 : 1000];       {Degree of Imbalance}

Hot_Eta_Reg          [0 : 1];              {Hot side thermal effectiveness, dimensionless}
Cold_Eta_Reg         [0 : 1];

Hot_Inlet_Temp       [abs_zero : 3000];    {Centigrade}
Cold_Inlet_Temp      [abs_zero : 3000];    {Centigrade}

Hot_Outlet_Temp      [abs_zero : 3000];    {Centigrade}
Cold_Outlet_Temp     [abs_zero : 3000];    {Centigrade}

CONSTRAINTS

Gamma               =  Hot_Reduced_Period/Hot_Reduced_Length
                       *Cold_Reduced_Length/Cold_Reduced_Period;

Hot_Eta_Reg         =  Cold_Eta_Reg/Gamma;

Cold_Eta_Reg        =  Hill_Willmott(Hot_Reduced_Length, Cold_Reduced_Length,
                       Hot_Reduced_Period, Cold_Reduced_Period);

Hot_Outlet_Temp     =  Hot_Inlet_Temp - (Hot_Inlet_Temp - Cold_Inlet_Temp)
                       *Hot_Eta_Reg;

Cold_Outlet_Temp    =  Cold_Inlet_Temp + (Hot_Inlet_Temp - Cold_Inlet_Temp)
                       *Cold_Eta_Reg;

PROBLEM DEFINED.
```

FIG. A1. GOLD problem module Reg2D.

```
gnosys> load Reg2D
gnosys> list
Solution Register 0

Independent vars :
.. Cold_Inlet_Temp             =  [No value assigned.]
.. Cold_Reduced_Length         =  [No value assigned.]
.. Cold_Reduced_Period         =  [No value assigned.]
.. Hot_Inlet_Temp              =  [No value assigned.]
.. Hot_Reduced_Length          =  [No value assigned.]
.. Hot_Reduced_Period          =  [No value assigned.]

Dependent vars :
.. Cold_Eta_Reg                =  Hill_Willmott(Hot_Reduced_Length,
                                  Cold_Reduced_Length, Hot_Reduced_Period,
                                  Cold_Reduced_Period)
.. Gamma                       =  Hot_Reduced_Period/Hot_Reduced_Length*
                                  Cold_Reduced_Length/Cold_Reduced_Period
.. Cold_Outlet_Temp            =  Cold_Inlet_Temp + (Hot_Inlet_Temp -
                                  Cold_Inlet_Temp)*Cold_Eta_Reg
.. Hot_Eta_Reg                 =  Cold_Eta_Reg/Gamma
.. Hot_Outlet_Temp             =  Hot_Inlet_Temp - (Hot_Inlet_Temp -
                                  Cold_Inlet_Temp)*Hot_Eta_Reg

Cost Function : None.
gnosys>
```

FIG. A2. GNOSYS session loading Reg2D.

```
SUBPROBLEM Packed_Bed OF Reg2D;

                {Regenerator model based on University of York Rig}

VARIABLES

                {Bed Geometry}

length          [0 : 10];              {length of bed - m}
radius          [0 : 5];               {radius of bed - m}
X_area          [0 : 40];              {X-sectional area - m**2}
S_area_bed      [0 : 1E4];             {total bed surface area for heat transfer - m**2}

                {Gas Properties - Both Periods}

av_temp         [abs_zero : 3000];     {'average' gas temp - Centigrade}

                {Gas Properties - Hot Period}

hot_mfr         [0 : 100];             {mass flow rate kg/s}
hot_v           [0 : 1000];            {linear velocity of gas - m/s}
hot_vfr         [0 : 2000];            {volumetric flow rate- m**3/s}
hot_G           [0 : 100];             {mass flow rate per unit X-sectional area - kg/s*m**3}
hot_k_visc      [0 : 1E-3];            {kinematic viscosity - m**2/s}
hot_C_gas       [0 : 1E4];             {specific heat - J/kg*K}
hot_rho_gas     [0 : 100];             {density - kg/m**3}
hot_k           [0 : 1];               {thermal conductivity - W/m*K}
hot_delta_P     [0 : 1E5];             {pressure drop, Pascals}
```

FIG. A3. GOLD subproblem module Packed-Bed.

```
                          {Gas Properties – Cold Period}

cold_mfr                  [0 : 100];              {mass flow rate kg/s}
cold_v                    [0 : 1000];             {linear velocity of gas – m/s}
cold_vfr                  [0 : 2000];             {volumetric flow rate– m**3/s}
cold_G                    [0 : 100];              {mass flow rate per unit X–sectional area – kg/s*m**3}
cold_k_visc               [0 : 1E–3];             {kinematic viscosity – m**2/s}
cold_C_gas                [0 : 1E4];              {specific heat – J/kg*K}
cold_rho_gas              [0 : 100];              {density – kg/m**3}
cold_k                    [0 : 1];                {thermal conductivity – W/m*K}
cold_delta_P              [0 : 1E5];              {pressure drop, Pascals}

                          {Packing Properties}

D_p                       [0 : 0.1];              {diameter of packing particles – m}
voidage                   [0 : 1];                {voidage fraction of bed – dimensionless}
C_solid                   [0 : 1E4];              {specific heat – J/kg*K}
rho_solid                 [0 : 2E4];              {density of solid – kg/m**3}
M_bed                     [0 : 1E5];              {mass of packing – kg}
hot_htc                   [0 : 1E4];              {heat transfer coefficient – W/m**2*K}
cold_htc                  [0 : 1E4];              {heat transfer coefficient – W/m**2*K}

                          {Period Lengths}

hot_P                     [0 : 1E5];              {length of hot period – s}
cold_P                    [0 : 1E5];              {length of cold period – s}

CONSTRAINTS
                          {Properties of Bed}

av_temp                   =  (hot_inlet_temp + cold_inlet_temp)/2;
X_area                    =  pi*sqr(radius);
S_area_bed                =  6/D_p*(1 – voidage)*X_area*length;
M_bed                     =  (1.0 – voidage)*X_area*length*rho_solid;

                          {Gas – Hot Side}

hot_vfr                   =  hot_mfr/hot_rho_gas;
hot_G                     =  hot_mfr/X_area;
hot_v                     =  hot_vfr/X_area;

                          {Gas – Cold Side}

cold_vfr                  =  cold_mfr/cold_rho_gas;
cold_G                    =  cold_mfr/X_area;
cold_v                    =  cold_vfr/X_area;

                          {Regenerator Parameters}

hot_reduced_length        =  hot_htc * S_area_bed /(hot_mfr * hot_C_gas);
hot_reduced_period        =  hot_htc * S_area_bed * hot_P/(M_bed * C_solid);
cold_reduced_length       =  cold_htc * S_area_bed /(cold_mfr * cold_C_gas);
cold_reduced_period       =  cold_htc * S_area_bed * cold_P/(M_bed * C_solid);

SUBPROBLEM DEFINED.
```

FIG. A3.—*Continued.*

```
gnosys> {...continued }
gnosys> load packed_bed
gnosys> list
Solution Register 0

Independent vars :
.. Cold_Inlet_Temp          =  [No value assigned.]
.. Hot_Inlet_Temp           =  [No value assigned.]
.. cold_C_gas               =  [No value assigned.]
{etc}

Dependent vars :
.. av_temp                  =  (Hot_Inlet_Temp + Cold_Inlet_Temp)/ 2
{etc}
.. Cold_Reduced_Length      =  cold_htc*S_area_bed/(cold_mfr*cold_C_gas)
.. Cold_Reduced_Period      =  cold_htc*S_area_bed*cold_P/(M_bed*C_solid)
.. Hot_Reduced_Length       =  hot_htc*S_area_bed/(hot_mfr*hot_C_gas)
.. Hot_Reduced_Period       =  hot_htc*S_area_bed*hot_P/(M_bed*C_solid)
{etc}
```

FIG. A4. GNOSYS session loading module Packed-Bed.

```
SUBPROBLEM Denton OF Packed_Bed;

                    {Calculates Heat Transfer Coefficients for a Packed Bed,
                    using Denton's Analysis}

VARIABLES

                    {Gas Properties - Hot Period}

hot_Re              [0 : 1E5];          {hot side Reynolds Number}
hot_Pr              [0 : 100];          {hot side Prantl Number}

                    {Gas Properties - Cold Period}

cold_Re             [0 : 1E5];          {cold side Reynolds Number}
cold_Pr             [0 : 100];          {cold side Prantl Number}

CONSTRAINTS

hot_Re              =  hot_v*D_p/hot_k_visc;
hot_Pr              =  hot_C_gas*hot_k_visc/hot_k;
hot_htc             =  0.57*hot_C_gas*hot_G*hot_Re↑(-0.3)*hot_Pr↑(-0.72);

cold_Re             =  cold_v*D_p/cold_k_visc;
cold_Pr             =  cold_C_gas*cold_k_visc/cold_k;
cold_htc            =  0.57*cold_C_gas*cold_G*cold_Re↑(-0.3)*cold_Pr↑(-0.72);

SUBPROBLEM DEFINED.
```

FIG. A5. Subproblem module Denton.

```
SUBPROBLEM Ergun OF Packed_Bed;

                      {Uses Ergun's equations for the calculation of pressure drop}

USE
g_c              =  1.0;                 {gravitational constant – dimensionless}

VARIABLES
hot_abs_visc     [0 : 1];                {absolute viscosity – kg/ms}
cold_abs_visc    [0 : 1];                {absolute viscosity – kg/ms}

CONSTRAINTS

hot_abs_visc     =  hot_k_visc*hot_rho_gas;
cold_abs_visc    =  cold_k_visc*cold_rho_gas;

hot_delta_P      =  length*hot_v*(1.0 – voidage)/(g_c*D_p*voidage↑3)*
                    ( 1.75*hot_G + 150*(1.0 – voidage)*hot_abs_visc/D_p);

cold_delta_P     =  length*cold_v*(1.0 – voidage)/(g_c*D_p*voidage↑3)*
                    ( 1.75*cold_G + 150*(1.0 – voidage)*cold_abs_visc/D_p);

SUBPROBLEM DEFINED.
```

FIG. A6. Subproblem module Ergun.

```
SUBPROBLEM Hot_Gas_Comp OF Packed_Bed;

                 {Allows Hot Side Gas Composition to vary}

VARIABLES
Hot_CO2          [0 : 1];                {relative proportion, dimensionless}
Hot_H2O          [0 : 1];                { " " }
Hot_N2           [0 : 1];                { " " }
Hot_O2           [0 : 1];                { " " }

CONSTRAINTS

Hot_N2           =  1.0 – Hot_CO2 – Hot_O2 – Hot_H2O;

Hot_C_Gas        =  Specific_Heat( 0,     {proportion of 'air' is zero}
                    Hot_N2,
                    Hot_O2,
                    Hot_CO2,
                    Hot_H2O,
                    av_temp – abs_zero);

Hot_rho_gas      =  (–abs_zero)/(av_temp – abs_zero)*
                    Gas_density(0,
                    Hot_N2,
                    Hot_O2,
                    Hot_CO2,
                    Hot_H2O);

Hot_k            =  Gas_Conductivity( 0,
                    Hot_N2,
                    Hot_O2,
                    Hot_CO2,
                    Hot_H2O,
                    av_temp – abs_zero);

Hot_k_visc       =  Gas_Abs_Viscosity(0,
                    Hot_N2,
                    Hot_O2,
                    Hot_CO2,
                    Hot_H2O,
                    av_temp – abs_zero)/Hot_rho_gas;

SUBPROBLEM DEFINED.
```

FIG. A7. Subproblem module Hot-Gas-Comp.

```
SUBPROBLEM Cold_Air OF Packed_Bed;

                    { Fixes Cold Side Gas Properties to be those of air}

CONSTRAINTS
Cold_C_Gas          =  Specific_Heat(1.0,       {proportion of air is 1}
                       0,                       {other gases zero}
                       0,
                       0,
                       0,
                       av_temp - abs_zero);

Cold_rho_gas        =  1.2928*(-abs_zero)/(av_temp - abs_zero);

Cold_k              =  Gas_Conductivity(1.0,
                       0,
                       0,
                       0,
                       0,
                       av_temp - abs_zero);

Cold_k_visc         =  Gas_Abs_Viscosity(1.0,
                       0,
                       0,
                       0,
                       0,
                       av_temp - abs_zero)/Cold_rho_gas;

SUBPROBLEM DEFINED.
```

FIG. A8. Subproblem module Cold-Air.

```
SUBPROBLEM Client_1 OF Packed_Bed;

USE
convert_fact_1      =  4180;              {btu/lbF - > J/kgK}
convert_fact_2      =  4.014e-3;          {Pascals - > inches of water}

VARIABLES
hot_water_drop      [0 : 15];             {inches of water pressure drop}
cold_water_drop     [0 : 15];             {inches of water pressure drop}
required_cold_out   [0 : 1500];           {Minimum Average Cold Temperature, Centigrade}
cold_out_diff       [0 : 1500];           {Check to ensure minimum is achieved}

CONSTRAINTS
hot_water_drop      =  convert_fact_2 * hot_delta_p;
cold_water_drop     =  convert_fact_2 * cold_delta_p;

cold_out_diff       =  cold_outlet_temp - required_cold_out;

                                          {Packing data supplied by client:}
D_p                 =  0.019;             {diameter of spheres}
C_solid             =  convert_fact_1*(9.996e-5*av_temp + 0.1947);{specific heat}
rho_solid           =  3644;              {density}
voidage             =  0.45032;           {voidage}

MINIMISE M_bed;        {required cost function}

SUBPROBLEM DEFINED.
```

FIG. A9. Subproblem module Client-1.

```
gnosys>
gnosys> {first load GOLD modules defining the problem}
gnosys>
gnosys> load Reg_2D
gnosys> load Packed_Bed
gnosys> load Denton
gnosys> load Ergun
gnosys> load Cold_Air
gnosys> load Hot_Gas_Comp
gnosys> load Client_1
gnosys>
gnosys> {Now interactively set requirements for this problem}
gnosys>
gnosys> fix{i.e. input set of contraints}
fix> cold_inlet_temp = 20
fix> hot_inlet_temp = 1400
fix> hot_P = 600
fix> cold_P = 600
fix> hot_mfr = 1.246
fix> cold_mfr = 1.122
fix> hot_water_drop.ub = 5.0{set upper bound on this variable}
fix> cold_water_drop.ub = 5.0
fix> hot_H2O = 0.17
fix> hot_CO2 = 0.085
fix> hot_O2 = 0.023
fix> required_cold_out = 950
fix> quit {exit fix mode}
gnosys>


gnosys> list {show final problem definition before optimisation}
Solution Register 0

Independent vars :
.. length                  =  [No value assigned.]
.. radius                  =  [No value assigned.]

Dependent vars :
.. required_cold_out       =  950
.. Hot_O2                  =  0.023
.. Hot_CO2                 =  0.085
.. Hot_H2O                 =  0.17
.. cold_mfr                =  1.122
.. hot_mfr                 =  1.246
.. cold_P                  =  600
.. hot_P                   =  600
.. Hot_Inlet_Temp          =  1400
.. Cold_Inlet_Temp         =  20
.. D_p                     =  0.019
.. rho_solid               =  3644
.. voidage                 =  0.45032
.. av_temp                 =  710
.. X_area                  =  3.14159*sqr(radius)
.. Hot_N2                  =  0.722
.. cold_C_gas              =  1137.77
.. cold_G                  =  1.122/X_area
.. cold_k                  =  6.67264E-2
.. cold_rho_gas            =  0.35919
.. C_solid                 =  1110.51

.. hot_C_gas               =  1293.19
.. hot_G                   =  1.246/X_area
.. hot_k                   =  6.69957E-2
.. hot_rho_gas             =  3.26298E-1
.. M_bed                   =  0.54968*X_area*length* 3644
.. S_area_bed              =  173.583*X_area*length
.. cold_k_visc             =  1.15098E-4
```

FIG. A10. Heat exchanger design using GNOSYS.

```
.. cold_vfr                  =  3.12369
.. hot_k_visc                =  1.21231E-4
.. hot_vfr                   =  3.8186
.. cold_v                    =  3.12369/X_area
.. hot_v                     =  3.8186/X_area
.. cold_Pr                   =  1.96257
.. hot_Pr                    =  2.34008
.. cold_abs_visc             =  4.13422E-5
.. hot_abs_visc              =  3.95575E-5
.. cold_delta_P              =  length*cold_v* 0.54968/ 1.73507E-3*( 1.75*
                                cold_G + 1.79408E-1)
.. hot_delta_P               =  length*hot_v* 0.54968/ 1.73507E-3*( 1.75*
                                hot_G + 1.71663E-1)
.. cold_Re                   =  cold_v* 0.019/ 1.15098E-4
.. cold_htc                  =  648.528*cold_G*cold_Re↑-0.3* 6.15412E-1

.. hot_Re                    =  hot_v* 0.019/ 1.21231E-4
.. hot_htc                   =  737.12*hot_G*hot_Re↑-0.3* 5.42193E-1
.. cold_water_drop           =  4.014E-3*cold_delta_P
.. hot_water_drop            =  4.014E-3*hot_delta_P
.. Cold_Reduced_Length       =  cold_htc*S_area_bed/ 1276.58
.. Cold_Reduced_Period       =  cold_htc*S_area_bed* 600/(M_bed* 1110.51)
.. Hot_Reduced_Length        =  hot_htc*S_area_bed/ 1611.32
.. Hot_Reduced_Period        =  hot_htc*S_area_bed* 600/(M_bed* 1110.51)
.. Cold_Eta_Reg              =  Hill_Willmott(Hot_Reduced_Length,
                                Cold_Reduced_Length, Hot_Reduced_Period,
                                Cold_Reduced_Period)
.. Gamma                     =  Hot_Reduced_Period/Hot_Reduced_Length*
                                Cold_Reduced_Length/Cold_Reduced_Period
.. Cold_Outlet_Temp          =  20 + 1380*Cold_Eta_Reg
.. Hot_Eta_Reg               =  Cold_Eta_Reg/Gamma
.. Hot_Outlet_Temp           =  1400 - 1380*Hot_Eta_Reg
.. cold_out_diff             =  Cold_Outlet_Temp - 950

Cost Function :
Minimise : M_bed

Value : M_bed

[Non-optimal point]


gnosys> detail 4 {set high reporting level}
gnosys> optimise


Start Optimisation: Complex Method.

Searching for the optimal point.
Checking problem definition.

Independent vars : 2
length, radius.

Dependent vars : 25
X_area, cold_G, hot_G, M_bed, S_area_bed, cold_v, hot_v, cold_delta_P,
hot_delta_P, cold_Re, cold_htc, hot_Re, hot_htc, cold_water_drop,
hot_water_drop, Cold_Reduced_Length, Cold_Reduced_Period,
Hot_Reduced_Length, Hot_Reduced_Period, Cold_Eta_Reg, Gamma,
Cold_Outlet_Temp, Hot_Eta_Reg, Hot_Outlet_Temp, cold_out_diff.

Problem is well-defined.

Must search for a feasible starting point
Found feasible starting point.
[ 4.22888, 1.95801 ]
Checked Feasible Cost Function : 102022
```

FIG. A10.—*Continued.*

```
Generating New Complex.
Cycle number : 1 Best Value : 30784.5 Convergence Index : 0
This Vertex : 30784.5

{etc...}


Complex has converged.


Best Value :
[ 4.40724E-1, 5.65181E-1 ]
Checked Feasible Cost Function : 885.89


End of Optimisation.


gnosys> list
Solution Register 0

Independent vars :
.. length                  =  4.40724E-1
.. radius                  =  5.65181E-1

Dependent vars :
.. required_cold_out       =  950
.. Hot_O2                  =  0.023
.. Hot_CO2                 =  0.085
.. Hot_H2O                 =  0.17
.. cold_mfr                =  1.122
.. hot_mfr                 =  1.246
.. cold_P                  =  600
.. hot_P                   =  600
.. Hot_Inlet_Temp          =  1400
.. Cold_Inlet_Temp         =  20
.. D_p                     =  0.019
.. rho_solid               =  3644
.. voidage                 =  0.45032
.. av_temp                 =  710
.. X_area                  =  1.00352
.. Hot_N2                  =  0.722
.. cold_C_gas              =  1137.77
.. cold_G                  =  1.11807
.. cold_k                  =  6.67264E-2
.. cold_rho_gas            =  0.35919
.. C_solid                 =  1110.51
.. hot_C_gas               =  1293.19
.. hot_G                   =  1.24163
.. hot_k                   =  6.69957E-2
.. hot_rho_gas             =  3.26298E-1
.. M_bed                   =  885.89

.. S_area_bed              =  76.7713
.. cold_k_visc             =  1.15098E-4
.. cold_vfr                =  3.12369
.. hot_k_visc              =  1.21231E-4
.. hot_vfr                 =  3.8186
.. cold_v                  =  3.11275
.. hot_v                   =  3.80522
.. cold_Pr                 =  1.96257
.. hot_Pr                  =  2.34008
.. cold_abs_visc           =  4.13422E-5
.. hot_abs_visc            =  3.95575E-5
.. cold_delta_P            =  928.346
.. hot_delta_P             =  1245.64
.. cold_Re                 =  513.841
```

FIG. A10.—*Continued.*

```
.. cold_htc                   =  68.5985
.. hot_Re                     =  596.373
.. hot_htc                    =  72.9507
.. cold_water_drop            =  3.72638
.] hot_water_drop             =  5
.. Cold_Reduced_Length        =  4.1254
.. Cold_Reduced_Period        =  3.21191
.. Hot_Reduced_Length         =  3.47574
.. Hot_Reduced_Period         =  3.41569
.. Cold_Eta_Reg               =  6.73913E-1
.. Gamma                      =  1.26222
.. Cold_Outlet_Temp           =  950
.. Hot_Eta_Reg                =  5.33912E-1
.. Hot_Outlet_Temp            =  663.201
.[ cold_out_diff              =  8.82989E-8

Cost Function :
Minimise : M_bed

Value : 885.89

[Optimal Solution]

gnosys> quit

End of GNOSYS Session.
```

FIG. A10.—*Continued.*

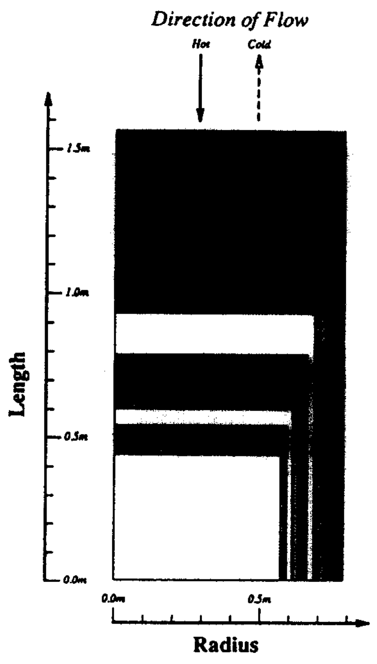

FIG. A11. Parameter study. Optimal designs with cold outlet temperatures varying from 950°C (smallest bed) to 1350°C (largest) in steps of 50°C.

UN LANGAGE DECLARATIF POUR LA CONCEPTION THERMIQUE DES ECHANGEURS REGENERATEURS

Résumé—On décrit quelques difficultés associées à la conception des échangeurs de chaleur industriels et on plaide pour une nouvelle approche pour la formulation du problème. Par une voie d'illustration, on décrit un petit langage déclaratoire et un problème exemplaire est défini dans le langage et résolu en utilisant la correspondance problème-environnement.

EIN DIALOG-SYSTEM FÜR DIE THERMISCHE AUSLEGUNG VON REGENERATIVEN WÄRMETAUSCHERN

Zusammenfassung—Einige Probleme bei der thermischen Auslegung industrieller Wärmetauscher werden beschrieben. Dies führt zum Wunsch nach einem neuen Verfahren mit einfacher Umschreibung des Problems. Unter Verwendung von Illustrationen wird eine eigens entwickelte kleine "Sprache" beschrieben. Ein Beispiel für die Umschreibung eines Problems wird gegeben, die Lösung wird vorgeführt.

ДЕКЛАРАТИВНЫЙ ЯЗЫК ДЛЯ ТЕПЛОВОГО РАСЧЕТА РЕГЕНЕРАТИВНЫХ ТЕПЛООБМЕННИКОВ

Аннотация—Описываются некоторые трудности, связанные с тепловым расчетом промышленных теплообменников, и предложен новый подход с использованием декларативного стиля постановки задач. Иллюстрируется малый декларативный язык, и в качестве примера формулируется расчетная задача, решаемая при соответствующих условиях.