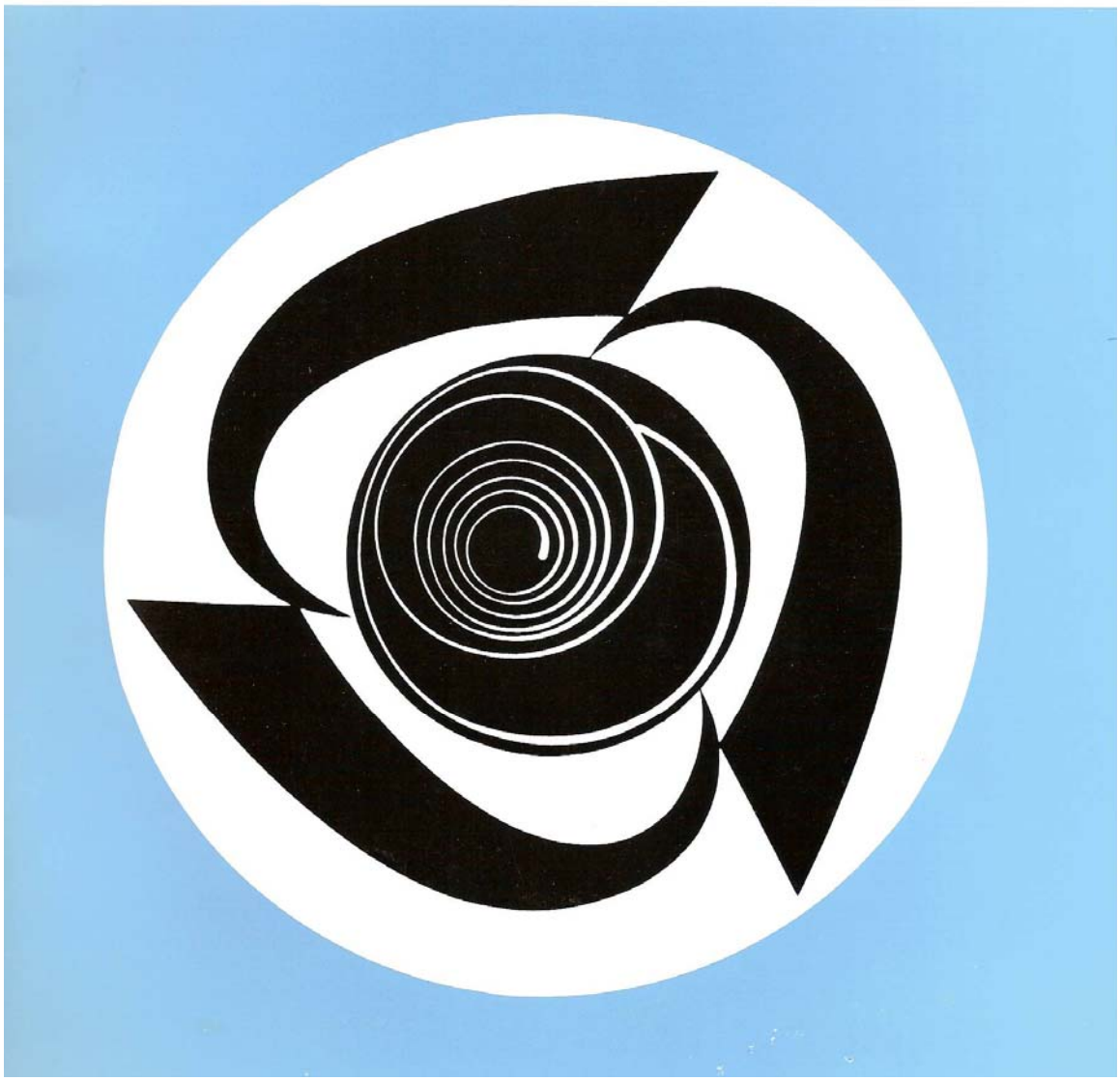
CACHE NEWS

NEWS ABOUT COMPUTERS IN CHEMICAL ENGINEERING EDUCATION.

No. 33 Fall 1991



The CACHE CORPORATION

WHAT IS CACHE?

CACHE is a not-for-profit organization whose purpose is to promote cooperation among universities, industry and government in the development and distribution of computer-related and/or technology-based educational aids for the chemical engineering profession.

CREATION OF THE CACHE CORPORATION

During the 1960s the rapid growth of computer technology challenged educators to develop new methods of meshing the computer with the teaching of chemical engineering. In spite of many significant contributions to program development, the transferability of computer codes, even those written in FORTRAN, was minimal. Because of the disorganized state of university-developed codes for chemical engineering, fourteen chemical engineering educators met in 1969 to form the CACHE (Computer Aids for Chemical Engineering) Committee. The CACHE Committee was initially sponsored by the Commission on Education of the National Academy of Engineering and funded by the National Science Foundation. In 1975, after several successful projects had been completed, CACHE was incorporated as a not-for-profit corporation in Massachusetts to serve as the administrative umbrella for the consortium activities.

CACHE ACTIVITIES

All CACHE activities are staffed by volunteers including both educators and industrial members and coordinated by the Board of Trustees through various Task Forces. CACHE actively solicits the participation of interested individuals in the work of its ongoing projects. Information on CACHE activities is regularly disseminated through CACHE News, published twice yearly. Individual inquiries should be addressed to:

CACHE Corporation
P. O. Box 7939
Austin, Texas 78713-7939
(512) 471-4933
CACHE@UTXVM.CC.UTEXAS.EDU.

CACHE NEWS

The CACHE News is published twice a year and reports news of CACHE activities and other noteworthy developments of interest to chemical engineering educators. Persons who wish to be placed on the mailing list should notify CACHE at the aforementioned address. Contributions from CACHE representatives are welcome. This issue was editied by D. M. Himmelblau with contributions from a number of CACHE members and representatives.

CACHE NEWS

No. 33 Fall 1991

CACHE Trustees

President:
Jeffrey J. Siirola, Eastman Kodak
(BITNET: SIIROLA@KODAK.COM)

Vice President
Michael B. Cutlip, University of Connecticut
(BITNET: MCUTLIP@UCONNVM)

Secretary
L. T. Biegler, Carnegie Mellon University
(BITNET: D101LB01@CMCCVB)

Executive Officer
David M. Himmelblau, University of Texas at Austin
(BITNET: CACHE@UTXVM.CC.UTEXAS.EDU)

Academic Trustees
Yaman Arkun, Georgia Institute of Technology
(BITNET: YAMAN_ARKUN@CHEMENG.GATECH.EDU)
Brice Carnahan, University of Michigan,
(BITNET:BRICE_CARNAHAN@UB.CC.UMICH.EDU)
James F. Davis, Ohio State University
(BITNET: KCGL1.ENG.OHIO-STATE.EDU)
Michael F. Doherty, University of Massachusetts
(BITNET: DOHERTY@UMAECS.BITNET)
Thomas F. Edgar, University of Texas at Austin
(BITNET: UTXCHE@UTCHPC.BITNET)
Bruce A. Finlayson, University of Washington
(BITNET: FINLAYSON@CHEVAX.CHEME.WASHINGTON.EDU)
H. Scott Fogler, University of Michigan
(BITNET: H_SCOTT_FOGLER@UM.CC.UMICH.EDU)
Ignacio Grossmann, Carnegie Mellon University
(BITNET: D391GR99@VB.CC.CMU.EDU)
Andrew N. Hrymak, McMaster University
(BITNET: HRYMAK@MCMASTER)

Academic Trustees, continued
Sangtae Kim, University of Wisconsin
(BITNET:KIM@CHEWI.CHE.WISC.EDU)
Richard S. H. Mah, Northwestern University
(BITNET: DICK_MAH@NUACC)
Manfred Morari, California Institute of Technology
(BITNET: MM@IMC.CALTECH.EDU)
Gintaras V. Reklaitis, Purdue University
(BITNET: REKLAITI@ECN.PURDUE.EDU)
Peter R. Rony, Virginia Polytechnic Institute and State University
(BITNET: RONY@VTVM1)
J. D. Seader, University of Utah
(BITNET: SEADER@UTAHCCA)
Warren D. Seider, University of Pennsylvania
(BITNET: SEIDER@CHEME.SEAS.UPENN.EDU)
John H. Seinfeld, California Institute of Technology
George Stephanopoulos, Massachusetts Institute of Technology

Industrial Trustees
Gary E. Blau, Dow Chemical Company
John C. Hale, E. I. DuPont de Nemours & Co.
J. J. Haydel, Shell Development Co.
Norman E. Rawson, IBM
(BITNET: NERAWSO@BETVMIC5.VNET.IBM.COM)
Edward M. Rosen, Monsanto Company
(BITNET: C00820ER@WUVMD)
Joseph D. Wright, Xerox
(BITNET: WRIGHT@MCMASTER)
or WRIGHT.XRCC-NS@XEROX.COM)

Advisory Committee
Lawrence B. Evans, Massachusetts Institute of Technology
James R. Fair, University of Texas at Austin
Thomas H. Lafferre, Monsanto Company
John J. McKetta, University of Texas at Austin

CACHE Task Forces and Committees

STANDING COMMITTEES

Publications
Prof. Brice Carnahan
Dept. of Chemical Engineering
University of Michigan
Ann Arbor, Michigan 48109
(313) 764-3366

Newsletter
Prof. David M. Himmelblau
Dept. of Chemical Engineering
University of Texas
Austin, Texas 78712
(512) 471-7445

Development
Dr. J. J. Haydel
Manager, Systems Development
Shell Development Company
Westhollow Research Center
P.O. Box 1380
Houston, Texas 77251-1380
(713)493-8141

Conferences
Prof. Richard S. H. Mah
Dept. of Chemical Engineering
Northwestern University
Evanston, Illinois 60201
(708) 491-5357

TASK FORCES

Artificial Intelligence
Prof. George Stephanopoulos
Dept. of Chemical Engineering
Massachusetts Institute of Technology
66-562
Cambridge, Massachusetts 02139
(617) 253-3904

Process Engineering
Dr. Edward M. Rosen
Monsanto Company - F2WK
800 N. Lindbergh Blvd.
St. Louis, Missouri 63167
(314) 694-6412

Case Studies
Prof. Manfred Morari
Dept. of Chemical Engineering
206-41
California Institute of Technology
Pasadena, California 91125
(818) 356-4186

Curriculum
Prof. Warren Seider
Dept. of Chemical Engineering
University of Pennsylvania
220 S. 33rd Street/D3
Philadelphia, Pennsylvania 19104
(215) 898-7953

Electronic Mail
Prof. Peter Rony
Dept. of Chemical Engineering
Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061
(703) 961-7658

NSF Development of Innovative Engineers
Prof. H. Scott Fogler
Dept. of Chemical Engineering
3074 Dow Building
University of Michigan
Ann Arbor, Michigan 48109
(313) 764-2384

NSF Simulated Laboratory Modules
Prof. Gintaras V. Reklaitis
School of Chemical Engineering
Purdue University
West Lafayette, Indiana 47907
(317) 494-4075

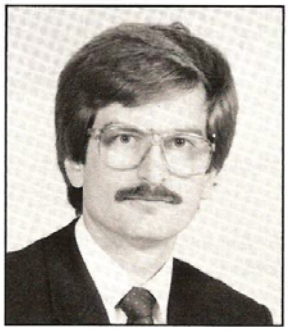
Graphical Computer Aids for ChE Education
Prof. Bruce A. Finlayson
Dept. of Chemical Engineering, BF-10
University of Washington
Seattle, Washington 98195
(206) 543-2250

CACHE Welcomes New Trustees

Sangtae Kim was born in Seoul, Korea, and emigrated with his family to Montreal, Canada, in 1965. He graduated from California Institute of Technology with concurrent B.S. and M.S. chemical engineering degrees in 1979, and from Princeton in 1982, with a Ph.D. in chemical engineering. In 1983, he joined the department of chemical engineering at the University of Wisconsin, and in 90 he was promoted to full professor. Also in 1990, he received a courtesy appointment in the department of computer science in recognition of his work with parallel computational algorithms.

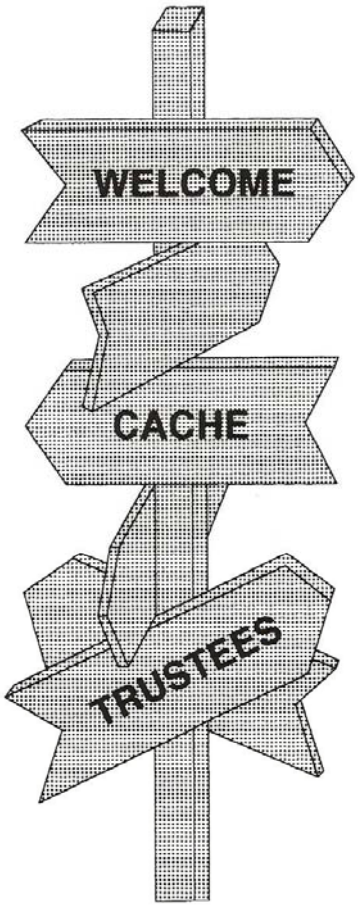
Sang's research program at Wisconsin has focused on the theoretical development for microstructure dynamics in viscous fluids, a discipline which encompasses applications ranging from stability of colloidal suspensions, rheology of polymer solutions and protein dynamics. His recent publications have lain the groundwork for large scale computations of the dynamics of complex microstructures in a viscous solvent, by algorithms well-suited for the next generation of parallel computers.

Sang and his wife Julie have two daughters, Denise 9, and Joyce 7.



Andy Hrymak received his B.Eng. degree from McMaster University in 1980, and a Ph.D. from Carnegie Mellon University in 1985. He joined the Department of Chemical Engineering at McMaster University in January, 1985, where he is currently Associate Professor. In the process systems area, he has co-supervised research projects in the area of process models for real-time optimization, large-scale linear programming algorithms for refinery scheduling, and expert systems for pulping processes. A major research effort is being undertaken in modelling the flow of polymeric materials for reaction injection molding, co-extrusion of polymer melts and reactive extrusion.

Andy has been involved in short courses and workshops in real-time optimization and polymer processing and has chaired technical sessions in conferences. Andy and his wife Cathy have a five year old son and three year old daughter.



Reflections from CPC IV

By Yaman Arkun, Georgia Institute of Technology

The fourth international Chemical Process Control Conference (CPC IV) took place in February 17-22, 1991 at South Padre Island, Texas. The conference was sponsored by CAST Division of AIChE and CACHE Corporation with support from NSF and industrial contributors:

Amoco Chemical
ChemShare
Dow Chemical
E. I. DuPont
Exxon Research & Engineering
Hoechst Celanese
Proctor & Gamble
Set Point
Shell Development
Tennessee Eastman
Weyerhaeuser

The technical program chairs were:

W. Harmon Ray, University of Wisconsin
Yaman Arkun, Georgia Institute of Technology

and the committee members were:

Thomas F. Edgar, University of Texas, Austin
D. Grant Fisher, University of Alberta
Christos Georgakis, Lehigh University
Sten Bay Jorgensen, Technical University of Denmark
Thomas McAvoy, University of Maryland
Manfred Morari, California Institute of Technology
James B. Rawlings, University of Texas, Austin
N. Lawrence Ricker, University of Washington
Dale E. Seborg, University of California, Santa Barbara
W. David Smith, Jr., E. I. DuPont

CPC IV presented the major advances that have taken place recently in the process control field under its theme *Future Needs and Challenges in Process Control.* The eight sessions of the conference focused on the modelling and identification, model-based process monitoring and control, control of nonlinear processes, learning systems, adaptive and AI control, and control technology in the year 2000.

It was evident from the presentations and the discussions that computers are playing an increasing role in process control education, research and plant automation. The industrial view papers indicated that the technological innovations in process automation are being shaped by new computer science paradigms. Several papers from academia illustrated the need for computer integration of rapidly diversifying real—time tasks including data analysis, pattern recognition, fault diagnostics, model building and optimization towards more intelligent implementation of complex control systems. The review paper by Wolfgang Marquardt focused on the state of the art and future challenges of dynamic process simulation with a survey of available software packages. Industry repeatedly expressed the need that dynamic simulation should be an integral part of process control education to help the engineer gain insight into process understanding to obtain better control solutions.

The Proceedings, *Chemical Process Control — CPCIV* (eds. Yaman Arkun and W. Harmon Ray), are now available from AIChE.

CACHE Corporation distributes worldwide.

An Introduction to Object-Oriented Programming

By Paul Rutz, MCC, Austin, Texas

Object-Oriented Programming Systems (OOPS) are growing rapidly in number and popularity. Smalltalk, C++, Objective C, Eiffel, and various object-oriented versions of Pascal and Lisp are now available on mini- and microcomputers.

Proponents of OOP claim the following advantages over non-OOP languages:

- Easier prototyping
- Increased programmer productivity
- Increased reusability of existing code
- Easier maintenance and extendibility
- More robust and reliable programs

Let's look at the tools that OOP provides to achieve these goals. Examples will be given in a Smalltalk-like syntax, but the meaning will be obvious and the concepts apply to all OOPL's (OOP Languages).

Objects, Classes, and Methods

Objects in OOP generally correspond to objects in the real world. A word processing program might have document, font, and ruler objects. An astronomy program might have star, planet, and comet objects. An OOP **class** defines the attributes and behavior for the set of objects of one type. OOP **methods** are procedures and functions that manipulate objects.

A **class** is defined by giving it a name, a set of variables that each object in the class will have, and a set of methods for manipulating those objects. For example:

```
class       Gauge
variables   value, min, max
methods     setValue,getValue,setMin, getMin,
            setMax, getMax
```

This creates a Gauge class with three variables and six methods. It does NOT create any gauges - the class declaration merely specifies what data and methods each gauge will have.

An **object** is one member, or instance, of a class. An object is created by giving it a name and specifying which class it belongs to:

```
gauge1 := Gauge new
```

This tells the Gauge class to create a new Gauge object called gauge1 which will have a current value, a minimum allowed value, a maximum allowed value, and our six methods to set and retrieve those variables.

A **method** is defined by specifying which class it belongs to, giving it a name and parameter list, and defining what the method does:

```
class       Gauge
method      setMin: newMin

min := newMin
```

This defines a Gauge method called setMin which takes one argument, newMin, and stores that argument into the min variable of a Gauge object. A similar method would be written for setMax. The setValue method needs to ensure that the new value is between the allowed min and max:

```
class       Gauge
method      setValue: newValue

if newValue < min
    value := min
else_if newValue > max
    value := max
else
    value := newValue
```

Calling a method is usually referred to as sending a message to an object. In non-OOP languages we call active procedures which manipulate passive data. In OOP we send messages to active objects which then manipulate themselves using their class methods in response to the message.

With the above definitions we could write:

```
gauge1 setMin: 0
gauge1 setMax: 10
gauge1 setValue: 4.5
```

This sends three messages to our gauge1 object requesting it to set its variables.

Inheritance

In an actual program we may need many specific kinds of

gauges with differing attributes or behavior. In OOP, we can borrow the existing Gauge definition to create new, similar, classes which vary or refine the generic Gauge definition as needed. *Inheritance allows us to derive new classes, called subclasses, from existing classes, called superclasses.* The subclasses inherit variables and methods from their superclasses while adding new variables and methods of their own.

Let's derive a Thermometer class from our Gauge class. For thermometers we may want to indicate whether the value is degrees Fahrenheit or degrees Celsius. We may also want our thermometer to be able to switch from one scale to the other during program execution. Here is a definition for the Thermometer subclass:

```
class          Thermometer
superclass     Gauge
variables      scale
methods        setScale
```

Each Thermometer object will have not only have a scale variable but also current, min, and max values, because a Thermometer is a Gauge. Here is the new setScale method:

```
class           Thermometer
method  setScale:newScale

if newScale = 'C' and scale = 'F'
     value :=  5/9 * (value - 32)
end if
if newScale = 'F' and scale = C'
     value :=  9/5 * value + 32
end if
scale := newScale
```

Now we are ready to create a thermometer:

```
t1 := Thermometer new
t1 setMin: -20
t1 setMax: 120
t1 setValue: 32
t1 setScale:'F'
```

Notice that some of the methods are defined in the Gauge class, others in the Thermometer class. When a message is sent to an object the object invokes a method from its own class if possible. If there is no such method then the object looks to its superclass for the method. Inheritance can be many levels deep, and the search for the method will continue up the superclass chain until the method is found or the top of the hierarchy is reached, in which case an error message results.

Some OOPL's allow **multiple inheritance**, in which a class may be derived from more than one superclass. In a windowing environment, where our thermometers could be displayed graphically, we could make class Thermometer a subclass not only of Gauge but also of a GraphicObject class. GraphicObjects would have variables for storing their bitmap image and coordinate position on the screen, methods for displaying and hiding the object, etc. The class hierarchy for a system using multiple inheritance can be depicted by an acyclic directed graph. See Figure 1.

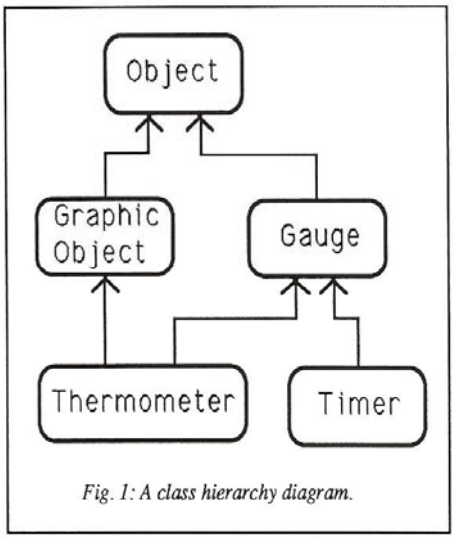


Fig. 1: A class hierarchy diagram.

Reuse of code through inheritance shortens programming time as the programmer can quickly create new classes from old. Program maintenance is easier because the behavior of many subclasses can be modified by modifying the superclass. For example, if we had many types of Gauges (thermometers, dials, etc.) and we decided to store additional data for each one (perhaps precision or accuracy) we could add the new variable to the Gauge class along with methods to manipulate it, and all the subclasses of Gauge would inherit the changes as well.

Composition vs. Inheritance

Suppose we are designing a device monitor which will contain several thermometers. Should we derive a DeviceMonitor class from our existing Thermometer class because the device monitor shares some of the behavior of a thermometer? Or perhaps the Thermometer class should be a subclass of DeviceMonitor because thermometers are found within the monitor? The answer is neither. Inheritance is used properly only when there exists an "is a" relationship between classes. Every Thermometer is a Gauge, so inheritance was appropriate there.

An alternate relationship is the "part of" relationship, or its converse, the "contains" relationship. A Thermometer is part of a DeviceMonitor. A DeviceMonitor contains Thermometers.

In this case we use composition to create a DeviceMonitor class that can contain Thermometers:

```
class       DeviceMonitor
variables   t1, t2 ...
methods     setT1, setT2 ...
```

Then we create a DeviceMonitor object and insert the Thermometer objects:

```
dm := DeviceMonitor new
dm setT1: Thermometer new
dm setT2: Thermometer new
```

A DeviceMonitor could additionally contain gauges, switches, and other objects we create.

Encapsulation

A class definition includes all the methods needed to manipulate objects in the class. These may include methods to create and initialize an object, set values within the object, retrieve those values, manipulate the object in arbitrarily complex ways, and dispose of the object. *In OOP, encapsulation is used to shield an object's data with methods such that the only way to reach the data is through the object's methods.* See Figure 2.

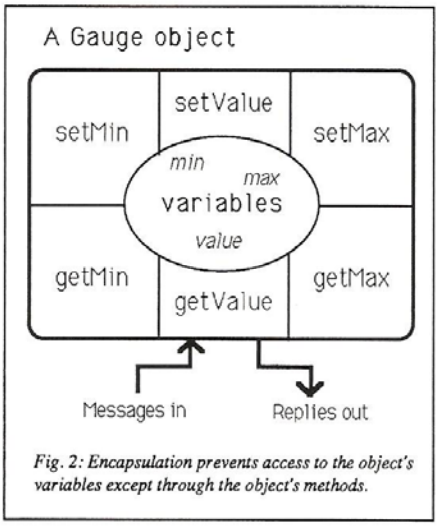


Fig. 2: Encapsulation prevents access to the object's variables except through the object's methods.

Suppose that we decide to redesign our thermometers such that they actually store the current temperature in both Fahrenheit and Celsius, rather than just storing one and converting to the other as needed. In order to modify the existing design as little as possible we could designate the 'value' variable to always hold the temperature in Fahrenheit and declare a new variable, 'valueC', to hold the Celsius equivalent:

```
class       Thermometer
variables   scale, valueC
methods     setScale
```

Remember that the 'value' variable is inherited from the Gauge class and so is not declared here. The setScale method changes to:

```
class       Thermometer
method      setScale: newScale

scale := newScale
```

The getValue method now returns one of the two temperature values, depending on the current scale:

```
class       Thermometer
method      getValue

if scale = 'F'
    return value
else
    return valueC
```

Clearly the two temperature variables, value and valueC, must always be kept in sync, but in most non-OOP languages any part of the program could modify one variable without recalculating the other. If we were making this design change in an existing program we would have to find every line that changes the value and add a line to change valueC as well.

With encapsulation, the only way to modify the temperature would be to call the setValue method. setValue would be rewritten to set both temperature variables:

```
class       Thermometer
method      setValue: newValue

super setValue: newValue
valueC := 5/9 * (value - 32)
```

The 'super setValue' is a call to the setValue procedure in the Gauge superclass that we wrote earlier. It stores the new value, while ensuring that it is between the allowed min and max. Then valueC is recomputed based on the new value. Note that the entire redesign of the operation of Thermometers was done within the Thermometer class itself. Since our original design used encapsulation, any existing programs that manipulated thermometers using setValue, getValue, and setScale would not have to be modified at all as a result of this design change.

Encapsulation promotes data integrity and lower mainte-

nance by centralizing the knowledge and control of an object within the object itself. It creates more robust programs by confining programming changes to the objects being changed.

Abstraction

You can see that encapsulation creates a "black box" view of an object. The rest of the program can't know, and doesn't care, how the data within the object is actually stored. Programs that use thermometers do not know or care whether the getValue method is reporting the value of an internal Celsius variable or actually calculating the Celsius temperature based on the Fahrenheit. We can modify the internal design of our Thermometer class, change the class methods as appropriate, and our programs will continue to work without any further modification.

In concrete terms, our Thermometer either does or does not actually store the Celsius temperature. In abstract terms, the Thermometer always "knows" the Celsius temperature, and provides it via the getValue method. *Abstraction means designing the external interface to a class (the services it provides via its methods) without regard for the actual physical properties of its internal storage.* When designing classes the external interface should be designed first. For gauges, we would first decide that a gauge should be able to tell us what its current value and allowable range of values are, and we should be able to ask it to change those values. Then we decide on method names to support those services, e.g. getValue, setValue, etc. Next, the internal variables needed to support the methods are chosen. In non-OOP languages, procedures and functions exist to support the manipulation of data. In OOP, it is preferable to say that variables and data structures exist to support the methods (procedures and functions) that an object responds to.

Abstraction leads to easier maintenance and more robust programs because changes to the internal structure or operation of classes do not require changes outside of the class itself. It allows you to use a class written by someone else simply by understanding the interface to the class, without studying the actual code or internal operation.

You should now be able to understand, or at least ponder, one of the fundamental OOP principles: Objects are encapsulations of abstractions.

Polymorphism

Suppose that we decide to have three different types of thermometers: MercuryThermometers, which display an old-fashioned thermometer with the mercury level indicating the temperature, DialThermometers, which display a scale with an arrow pointing to the current temperature, and DigitalThermometers, which simply print the numeric value. In most programming languages we would add a variable to each thermometer record or structure to indicate what type of thermometer it was, then use a case statement to select the appropriate display procedure:

```
select thermometer_type
    case MercuryThermometer
        call display_mercury_thermometer
    case DialThermometer
        call display_dial_thermometer
    case DigitalThermometer
        call display_digital_thermometer
```

In fact, any time we wanted to treat the different types of thermometers in different ways we would have to include a similar case structure. If we decide to add a fourth type of thermometer we would have to find ALL the places in our program where that case structure occurs and modify it. This would be time-consuming and error-prone.

In an OOPL we would instead create a method named 'display' in each class:

```
class       MercuryThermometer
method      display

...code to display the MercuryThermometer...
```

The DialThermometer 'display' method would display the dial thermometer, and so forth. Now in place of the above case statement we can just write:

```
t1 display
```

This asks t1 to display itself using the 'display' method from its own class. The AnalogThermometer method will be called if t1 is an AnalogThermometer, the DialThermometer method will be called if t1 is a DialThermometer, etc. If we later added a new type of thermometer, the new class would merely need to include a 'display' method to display itself. The rest of the program would not need to be modified. If each of our Gauge subclasses had a 'display' method then we could write:

```
gauge1 display
```

where gauge1 could be a Thermometer or any other kind of gauge, and the appropriate method for displaying that gauge would be used.

Polymorphism allows methods in different classes to share the same name, and the actual method that is invoked depends on the class of the object for which it is called. Polymorphism creates more robust programs with easier extendibility by reducing the amount of change required to existing code when the program is enhanced.

Operator Overloading

If you have ever written a Basic program you are already familiar with one kind of polymorphism: the expression 1+2 performs addition, but a$ + b$ does string concatenation. The plus operator does double-duty; its operation depends on the operands that surround it. But what if you wrote 1 + a$? Or, if A and B were numeric arrays and you wrote A + B? In most languages such expressions are meaningless and would generate compiler errors.

In OOP we can use operator overloading to redefine the meaning of operators based on their operands. The plus operator when used with numeric arrays could be defined to add corresponding elements of the arrays and return another array. When used with a number and a string variable, plus could be defined to convert the string to a number and then do addition. Operator overloading is a notational convenience that prevents the need to declare new function names for each type of addition (e.g. add_string_to_num(a$,x), add_arrays(A,B), etc.)

Object-Oriented vs. Top-Down

Since the mid-60's top-down design has been the prevalent program design method. Top-down design is based on functional decomposition, in which we envision our program as a hierarchy of functions. The main program sits at the top of the hierarchy and is divided into several subordinate functions, each of which consists of other functions, and so on. Knowledge of the program's data structures and how to manipulate them is distributed among the program's functions. Over the years, however, programmers have made an interesting observation: In the long run a program's functionality tends to change as new or enhanced features are added, but the objects dealt with by the program tend to remain the same. In a word processing program, for instance, the data objects will always be documents, paragraphs, sentences, words, etc., while the functionality will vary as new features are added, or the display is changed from character-based to graphics-based, or the user-interface is changed from keyboard-driven to mouse-driven. With top-down design these frequent changes in functionality result in a continual reshuffling at the highest design level, the functional hierarchy. This, in turn, can require considerable recoding.

Object-Oriented Design is based on data structure decomposition. We begin not by asking what our program will do (its functionality), but what it will do it to (its objects). Instead of designing a functional hierarchy, we design a data class hierarchy. For each class we design an interface, then create the data and methods needed to support that interface. Once a class is complete it is placed into a class library from which it can be used and reused in many applications. OOP places a heavy emphasis on building class libraries that can later be quickly assembled into a prototype or a complete program.

Objections to OOP

"Isn't OOP just for windowing systems?"

This is a common misconception. Certainly OOPL's have been used to write windowing systems, and are often used in a windowing environment. But OOP is a general-purpose design method, and OOPL's have been used successfully in a wide range of applications.

"Won't OOPL's run slower than other languages?"

Many programmers' first exposure to OOP is with a Smalltalk interpreter. Like any interpreter, these Smalltalk systems can seldom match the speed of a compiled application. Compilers are available for most OOPL's, however, and the first Smalltalk compiler was released recently. There is still a speed penalty to be paid for some OOP features. Nevertheless, as computer time continues to become cheaper and programmer time more expensive, it is imperative that we use the best available tools to increase programmer productivity, while relying on ever-increasing hardware capabilities to keep up with our new software's demands. Today, a tightly-coded assembly language program can outperform the code produced by a Pascal or Fortran compiler, but few programmers are willing to give up the ease and productivity gains of the higher level languages.

"OOP has not proved itself in large, long-term application development."

This is probably the most serious objection. Although some very impressive systems have been developed, it is difficult to quantify the claimed gains in productivity and reusability, and impossible to predict the actual effects on the 10, 20, and even 30-year lifetimes and required maintenance that some large systems require. The initial results, however, are encouraging enough to convince many large software vendors, including Apple, IBM, DEC, and Microsoft, to commit to OOP in a large way.

"Well, I've really been doing it this way all along."

Good for you! Many programmers evolved an object-oriented style of their own long before OOP became a buzzword. The concepts of OOP are not brand new or unique to OOP languages. What distinguishes OOP is its endorsement of data-driven design using encapsulation, abstraction, inheritance, and polymorphism as THE fundamental principles of design and implementation. If you find yourself already using some of these techniques in your own programming, you may enjoy using a language that provides explicit support for them.

AIRSPILL - A Graphic Toxic Material Dispersion Teaching Program

By William Resnick
Department of Chemical Engineering
Isreal Institute of Technology
Haifa, Isreal

Introduction

The possibility of the release of hazardous or toxic materials and the possible health hazards in the event that a release does occur are topics of prime importance to the chemical engineers who design and operate chemical plants as ~ell as to the public at large. Both those involved with the chemical industry and the general public are aware that occasional releases of toxic or hazardous materials to the atmosphere do occur during plant operation. These releases which are the result of loss of containment can occur on the plant site due to equipment failure or as a result of accident or explosion. Loss of containment, however, can also occur outside the plant boundaries in locations well-removed from the plant due to accidents involving rail tank cars, highway tankers and ships. Pipe line failures have also occurred in the past and releases of hazardous or toxic materials have resulted.

As a consequence of these problems it is important that the chemical engineer be well-schooled in the design of plant so that the probability of loss of containment on the plant site will be minimized. The evaluation and analysis of the consequences of emissions of toxic materials in the event that they do occur must also be within the province of the chemical engineer. This analysis must consider the possible consequences not only to personnel on the plant site but also to the community adjacent to the plant site or to the location of an off-site release. Such an analysis is imperative for emergency planning. In addition, the consequences of the loss of containment may be of importance in site selection.

The AIRSPILL program was developed to allow the chemical engineering student to study the release of a toxic material and its subsequent dispersion in the atmosphere. The results are presented in graphical form as a ground-level toxicity map superimposed on a map of the plant site and its immediate environs. AIRSPILL allows the student to study the effects of atmospheric and climatic conditions on the dispersion, permits an estimate of the quantity of toxic material which would be released in the event of a failure or explosion, allows a study of the effect of the elevation of point of toxic material emission, and presents the results in an easily understandable manner.

Modelling Procedure

A two-step procedure is required to model the consequences of a toxic material release. In the first step the rate of release of the toxic material into the atmosphere is estimated. The second step involves the calculation of the toxic material concentration as a function of downwind and crosswind distance from the release point. AIRSPILL calculates the concentration at ground level and presents the results as isopleths of constant concentration for several toxicity levels.

Emission Rates

If the rate of emission of toxic material is known or can be assumed AIRSPILL permits the direct entry of this information. If not known the program can estimate the rate of emission for catastrophic tank failure or for several cases of equipment or component failure. The equations used to model the emission rates are well-known to chemical engineers from their studies of transport phenomena and thermodynamics. Three scenarios for loss of containment are considered for the failure cases - liquid phase discharge through a hole in a vessel followed by flashing of part of the liquid; gas phase discharge through a break in a line or vessel and two-phase emission with flashing occurring during discharge. Only vapor emissions are considered because the program is designed to calculate atmospheric dispersion with the goal of presenting the user with an easily understandable graphics display of the toxicity results. A high degree of accuracy is not required for this purpose so subsequent vaporization of the discharged liquid is not considered nor is the possible boiling or vaporization of a liquid pool.

Modelling of the vapor release due to loss of containment by catastrophic failure or explosion is based on energy balance considerations. The model assumes that all of the vessel s contents are released and an energy balance is made to calculate the fraction of the material which flashes to vapor. AIRSPILL does not consider the possibility of aerosol formation and the subsequent vaporization of the liquid droplets.

Dispersion Models

Hanna and Drivas (1987a) recently presented a comprehensive review of many of the vapor cloud dispersion models which have been developed. The three-dimensional Gaussian plume equation for neutrally-buoyant gas was chosen to model emission dispersion for AIRSPILL. This equation was selected for several reasons - the parameters required for its use are readily available, it is relatively simple and it is the foundation for most of the models recommended by the EPA for neutrally buoyant gas releases. In addition, Hanna and Drivas (1987b) point out that the Gaussian plume equation appears to work as well as complex, three-dimensional numerical models. Although most toxic materials are dense gases the concentrations in the plume decreases rapidly due to dispersion and dilution so that the use of the neutrally buoyant model does not introduce significant error for a scoping analysis of the AIRSPILL type. In addition, the toxic concentrations of hazardous materials generally are very low, <1 g/m^3, so the gas density can be considered to be close to that of the ambient air.

The Gaussian formula is

$$C=[Q/2\pi u\sigma_y\sigma_z)]\exp[-y^2/2\sigma_y^2]\{\exp[-(h-z)^2/2\sigma_z^2]+\exp[-(-h-z)^2/2\sigma_z^2]\}$$

where C = concentration, kg/m3
Q = source emission rate, kg/s
u = wind speed, m/s
σ_y, σ_z = lateral and vertical dispersion coefficients, m
h = elevation of the emission, m
y = crosswind position from plume center line, m
z = elevation above ground, m

The numerical values of the dispersion coefficients are functions of the atmospheric stability and of the downwind distance from the emission source, x. Six atmospheric stability classes, A to F, have been defined. They are dependent upon wind speed, solar insolation and cloudiness. Stability class A is the most unstable and occurs under daytime conditions with high solar radiation intensity. Class F, the most stable category, would occur at night if there were low wind speeds and little or no cloud cover. Tables which define the stability categories are presented in sources such as Hanna and Drivas (1987a), Stern et al, (1984) and Pasquill (1974). Values for dispersion coefficients are available in Gifford (1961) and also in the same sources cited above for the stability classes. The numerical values for the coefficients can obtained either from graphs, from correlating equations or from tabulations. AIRSPILL uses the equations and tabular data from Stern et al (1984).

The Gaussian plume model as shown in equation (l) is known as the Pasquill-Gifford model when used with the Gifford dispersion coefficients. It permits the calculation of the concentration of the emitted material as a function of the downwind distance, x, and the crosswind location from the plume centerline, y. AIRSPILL calculates the concentration at ground level, z = 0, which would represent the concentration of toxic material to which personnel on the plant site or persons in the environs would be exposed.

Presentation of Results

The concept of Acute Toxicity Concentration (ATC) was adopted for the presentation of the toxicity calculations. The ATC is defined as the concentration of a toxic substance that will result in acute health effects on the exposed population with expected fatality results of one fatality out of twenty persons exposed during a one hour period (Costanza et al, 1987). Values for the ATC for extraordinarily hazardous substances as proposed for the New Jersey Toxic Catastrophe Prevention Act (TCPA 1987) are presented by Baldini and Komosinsky (1988). Data for LC_{LO} and for LC_{50} from animal tests and values for IDLH were used to select the ATC values*. Generally the value selected for the ATC was the lowest from among the value for LC_{LO} the value for O.l*LC_{50} and the value for IDLH.

Although AIRSPILL calculates the concentration of toxic material in the atmosphere downwind of the release point these results can be presented to the user in a more meaningful way if they are shown in terms of a toxicity level rather than as a concentration. When the ATC results are presented in a graphical form on a plot of the plant site and its immediate surroundings the user can immediately see the effects of the emission on plant personnel and on the people adjacent to the site in a dramatic form. The importance of plant design for safety and integrity is emphasized and the necessity for emergency planning is dramatized. In the event that controlled releases during normal plant operations are required AIRSPILL demonstrates the effects of emission height and atmospheric conditions so that the design and operation would be done in a manner which would not put human life in jeopardy.

The data supplied with AIRSPILL includes information concerning a small plant and its surroundings. The sample plant is shown in Figure 1 and the toxicity results are superimposed on this plot. The plot includes six units within the plant boundaries and fourteen buildings outside the boundaries. The largest building outside the plant is a school which is about 150m due west of the Engineering structure. The other buildings outside of the plant boundaries include five houses and eight apartment buildings.

* LC_{LO} is the lowest concentration that will cause at least one fatality out of a test population, usually 20 animals, for test durations of 10 minutes to 8 hours. LCso is the lowest concentration that will cause at least 50% fatalities of the test population for test durations of 10 minutes to 8 hours but usually of 4 hours. IDLH (immediate danger to life and health) is the maximum concentration for which one should not suffer any escape-impairing symptoms or health impairing effects within 30 minutes.

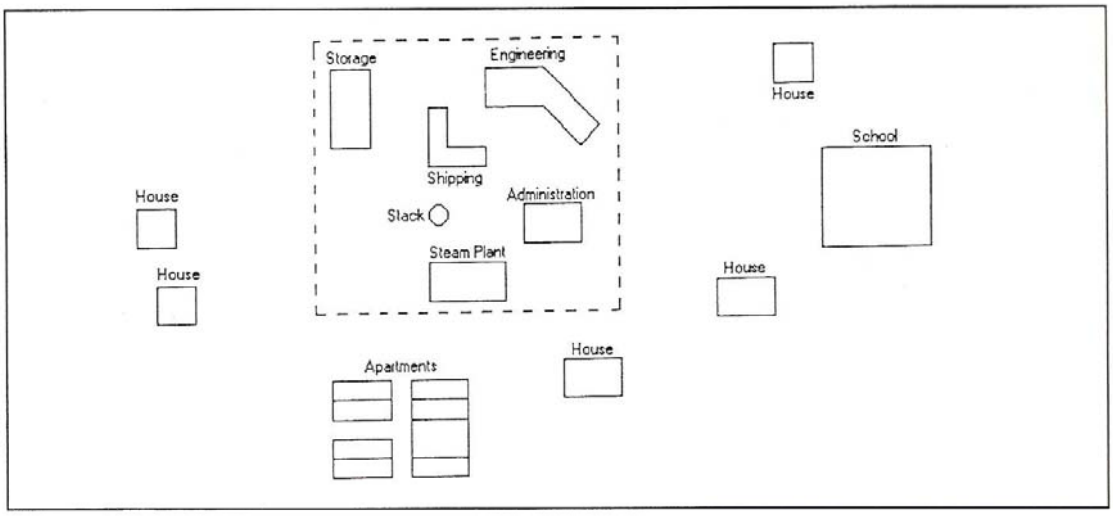


Figure 1 - Sample Plant and Surrounding Area

AIRSPILL presents its toxicity results as a nest of isopleths superimposed on the plot plan as shown in Figure 2. The outermost isopleth bounds the region for which the the toxicity level is <0.2*ATC; the next isopleth bounds the region for a toxicity concentration which is <0.4*ATC followed by isopleths for toxicity levels which are <0.6*ATC and <0.8*ATC with the innermost isopleth bounding the region for which the toxicity levels are equal to or greater than the ATC. Obviously, emergency measures would be in order for toxic material releases which would result in persons being exposed to concentrations which are greater than the ATC and particularly susceptible persons would be put at risk at concentrations which are less than the ATC.

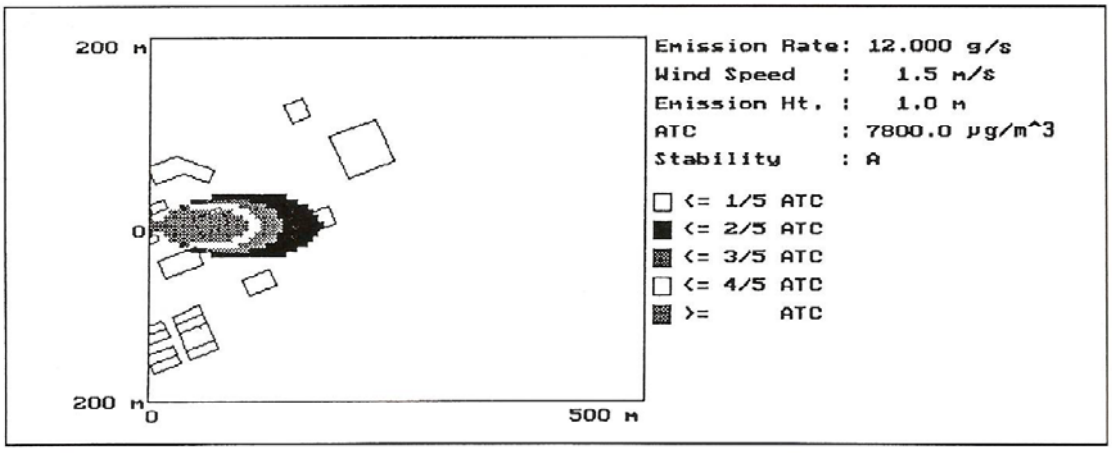


Figure 2 - Graphic Output

The initial graphical presentation of the calculation results for an emission shows an area whose dimensions are 400 by 500 meters. If necessary a more complete picture of the graphical results can be obtained by zooming the output in the range of 50% to 200~ of the initial output. The zooming can also be repeated so that, in principle, the effects of the emission can be evaluated over any distance desired.

Databases

AIRSPILL uses two databases, a chemical database and a plot plan database. The data presented in the chemical database, DATABASE.DAT, includes the compound name, the CAS ID number, molecular weight, liquid density, heat capacity ratio, heat capacity parameters, Antoine equation parameters and the ATC. The database presently available with AIRSPILL comprises complete data for 14 compounds which permit the use of the explosion and failure models to calculate the emission rates. ATC data are provided for an additional 95 compounds but the user must supply the emission rate as part of the input data for these compounds.

The file PLAN.DAT contains information about the location of the buildings and also includes the number of occupants in each building. The data concerning the number of occupants is not used in the present version of AIRSPILL but the program is now being expanded to permit an estimate of the number of fatalities which could result in the event of a toxic material release. Such information is necessary for a complete consequence analysis.

The manual which accompanies AIRSPILL provides information on the data file structure and how to modify it. From this information the user can easily add compounds to the chemical database. The plot plan database can be easily modified to suit plant topographies other than the one produced for AIRSPILL.

AIRSPILL in Teaching

AIRSPILL was developed by several students as their term project in an elective course entitled Pollution Control Technology which was presented by the author, then on a visiting appointment at the Chemical Engineering Department, University of California, Los Angeles. A number of lectures within the framework of this course were devoted to the pollution resulting from loss of containment of various chemicals and the health hazards in the event that the release of toxic chemicals occurred. The background material which was presented included lectures and reading assignments on the physics of the atmosphere, effects of pollution on the atmosphere, removal mechanisms and models for the transport and dispersion of emissions. Although puff and plume dispersion models were considered the author suggested to the students that only a plume model should be considered for AIRSPILL because emissions due to loss of containment usually occur over a period of time of at least several minutes and sometimes much longer in the case of an off-site accident. Thus, the use of a steady-state plume model such as the Gaussian equation would be justified, especially for a scoping or study operation such as those performed by the students. The instructor should point out the possible effects of the various simplifying assumptions which are made in the AIRSPILL modelling.

AIRSPILL could be used within the framework of a number of courses in the usual chemical engineering curriculum. An obvious candidate for AIRSPILL use would be the capstone design course which usually includes some safety considerations. Appropriate assignments could deal with mapping of release consequences, selection of release vent elevation, selection of storage quantities for hazardous and toxic materials, study of possible loss of containment scenarios, emergency planning, etc.

Many departments present elective courses with titles such as Pollution Control; Loss Prevention and Waste Minimization; Environmental Aspects of Chemical Plant Operations; Process Safety and Process Design and Analysis. Courses of this nature would be candidates for the use of AIRSPILL in the appropriate points in the course syllabus.

System Requirements

AIRSPILL runs on any IBM or compatible personal computer. It requires at least 256 kbytes of system memory and DOS operating system version 2.0 or higher. A color monitor is essential but graphics can be CGA, EGA or VGA. Neither the toxicity level isopleths nor the building plot plan will appear on a monochrome display. The program is written in Turbo-Pascal.

The output of the present version of AIRSPILL is limited to a graphics display on the monitor screen of the ATC isopleths superimposed on the on-and-offsite buildings plot. A future version will provide hard copy output of the ATC isopleths as a function of the x and y coordinates.

AIRSPILL is a useful tool for teaching purposes but, as it deals only with the atmospheric dispersion of vapors, it has limited capabilities. A number of commercial software packages are available which possess the capability of analyzing almost any conceivable hazard situation. They are able to deal with vapor discharge, liquid discharge, pool boiling, boiling liquid evaporating vapor explosions (BLEVE) and many more. Some of them also are user-friendly. The only disadvantage of these packages is that they are rather expensive for university use. Acquisition of a commercially available package should be considered if the financial resources are available.

Acknowledgements

The author would like to express his thanks to Jonathan Deeds who did the graphic implementation of AIRSPILL, to Kyle Fowler who developed the failure emission rate portion of the program and to Eric Hopkins who prepared the chemical data base. They gave unstintingly of their time and AIRSPILL is really their child.

The author would like to express his thanks to the Chemical Engineering Department, UCLA, which produced a visiting appointment that allowed the author to present Pollution Control Technology, the latest version of a course he has been developing under several names and versions during the past few years. Presenting the course also allowed the author to meet and interact with a stimulating, highly motivated and enthusiastic group of students.

References

Baldini, R. and Komosinsky, P., 1988, J. Loss Prev. Process Ind., 1, 147.

Costanza, P., Hagen, G., Kalagnanam, R. and Komosinsky, P., 1987, Plant/Operations Progress, 6(4), 215.

Gifford, F.A., 1961, Nuclear Safety, 2, 47-55.

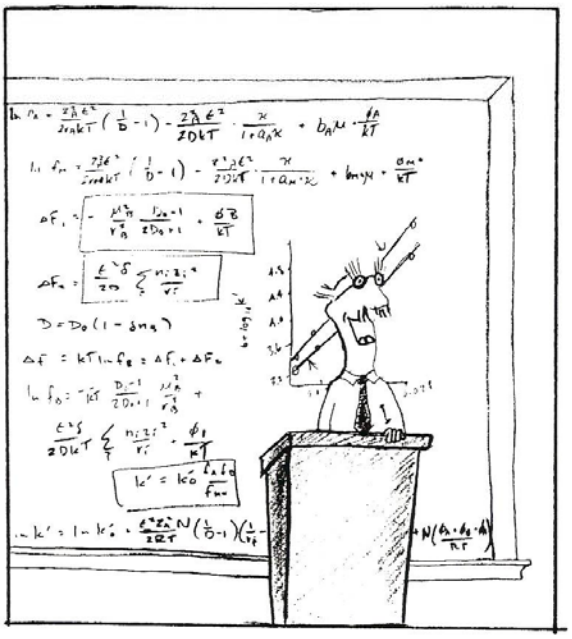
Hanna, S.R. and Drivas, P.J., 1987a, Guidelines for Use of Vapor Cloud Dispersion Models, Center for Process Safety, A.I.Ch.E., New York

Hanna, S.R. and Drivas, P.J., 1987b, op cit. p. 125.

New Jersey Department of Environmental Protection, 1987, Basis and Background Document for Proposed New Rule N.J.A.C. 7:31, Toxic Catastrophe Prevention Act Program, Bureau of Release Prevention, September, 1987.

Pasquill, F., 1974, Atmospheric Dispersion, 2nd ed., Halsted Press, London, 368.

Stern, A.C., Boubel, R.W., Turner, D.B. and Fox, D.L., 1984, Fundamentals of Air Pollution, 2nd ed., Academic Press, Orlando, 282-287.



NONE OF THIS MEANS A THING

WHAT DO YOU THINK OF THAT? HA!

CACHE Departmental Representatives Electronic Mail Database

by Peter R. Rony, Virginia Tech,
H. Scott Fogler, University of Michigan,
and Yaman Arkun, Georgia Institute of Technology

In early May 1991, CACHE sent the following request to CACHE member institutions around the world:

> "Dear Colleague:
>
> "In order to strengthen our interaction and the lines of communication between CACHE and Chemical Engineering Departments, CACHE is updating the address list and generating a new e-mail network for its departmental representatives. We want to make sure that we personally know all our representatives, and that they are well informed of CACHE activities, meetings, and educational products. This way we believe that our departmental representatives will play a more salient role in our activities, and CACHE will better serve its supporting departments.
>
> "We would appreciate it if you would provide us with the name, address, phone number, and e-mail address of the current CACHE representative in your department. The form is listed below.
>
> "Thank you for your time."
>
> (Signed) Yaman Arkun, H. Scott Fogler, Peter R. Rony
>
> Name of CACHE representative:
> Address:
> Phone number:
> E-mail address:

Approximately 80 responses were received. In early August 1991, they were entered into a database with the following characteristics:

- CACHE Database Name (draft): REP_01.DBF (August 24, 1991)
- Database Fields: CACHE_REP, NICKNAME, USERID, NODE, UNIVERSITY, CITY, STREET, STATE, COUNTRY, ZIPCODE, PHONE
- Database Software: DBASE IV version 1.1, with MousePatch
- Validation of email userids?: Not validated yet as of Aug 24, 1991

CACHE News is printing selected fields (UNIVERSITY, CITY, CACHE_REP, NICKNAME, USERID, NODE, in the order given) of the responses received to date. Please check your entry, if listed below, for accuracy or add your department CACHE representative to our growing list. Please send all additions in writing to CACHE Main Office, P.O. Box 7939, Austin, TX 78713-7939 rather than electronically to a CACHE trustee.

The selected departmental "nickname" is in upper-case letters. Some departments (4-line entries) do not yet have a USERID or NODE. If prefer a different "nickname," please communicate such a request to the CACHE Main Office.

Thank you for your participation and support of this activity.

Auburn University
Auburn University
Mahmoud El-Halwagi
AUBURN
mahmoud
eng.auburn.edu

Alberta, University of
Edmonton, Alberta
R. E. Hayes
ALBERTA
userhaye
hts.ucs.ualberta.ca

Arkansas, University
Fayetteville
Roy W. Penney
ARKANSAS

Arkansas, University of
Tuscaloosa
Leon Y. Sadler
ALABAMA

British Columbia, University of
Vancouver, British Columbia
Bruce Bowen
BRITISH_COLUMBIA
bruce_bowen
mtsg.ubc.ca

Brigham Young University
Provo
Michael J. Beliveau
BRIGHAM_YOUNG
zitwit
bill.byu.edu

Bucknell University
Lewisburg
Michael E. Hanyak, Jr.
BUCKNELL
hanyak
bucknell.edu

Concepcion, University of
Concepcion 3
Ricardo Reich
CONCEPCION
rreich%udec
uchcecvx

Clarkson University
Potsdam
Angelo Lucia
CLARKSON

Cornell University
Ithaca
Thansis Panagiotopoulos
CORNELL
panagiot
nemesis.cheme.cornell.edu

\Cleveland State University
Cleveland
E. Earl Graham
CLEVELAND_STATE
r0174
csuohio

California-Davis, University of
Davis
Ahmet Palazoglu
CALIFORNIA_DAVIS
anpalazoglu
ucdavis.edu

Clemson University
Clemson
Stephen S. Melsheimer
CLEMSON
ssmls
clemson

Colorado, University of
Boulder
Angelo Lucia
COLORADO
ramirez_f
cubldr.colorado.edu

Dartmouth University
Hanover
Mark Franklin
DARTMOUTH
mark.franklin
dartmouth.edu

Denmark, Technical University of
DK-2800 Lyngby
Rafiqul Gani
DENMARK
ketrg211
vm.uni-c.dk

E.T.S.E.I.B.
0828 Barcelona
Luis Puigjaner Corbella
ETSEIB
aec
eq.upc.es

Georgia Institute of Technology
Atlanta
F. Joseph Schork
GEORGIA_TECH
joseph_schork
chemeng.gatech.edu

Houston, University of
Houston
Richard Willson
HOUSTON
willson
uh.edu

Institut D'Automatique
1015 Lausanne
Dominique Bonvin
INSTITUT_AUTOMATIQUE
bonvin
elia.epfl.ch

Iowa State University
Ames
Deqn L. Ulrichson
IOWA_STATE
dlulrich
iastate.edu

Idaho, University of
Moscow
David C. Drown
IDAHO
cheadmin
idui1.csrv.uidaho.edu

Illinois Institute of Technology
Chicago
Ali Cinar
ILLINOIS_INST_TECH
checinar

Iowa, University of
Iowa City
Victor G. S. Rogers
IOWA
vgrodgers
icaen.uiowa.edu

Kansas, University of
Lawrence
Colin S. Howat
KANSAS

Johns Hopkins University
Baltimore
Joseph L. Katz
KENTUCKY
chf
jhuvms.hcf.jhu.edu

Kyoto University
Kyoto, 606
Dr. Iori Hashimoto
KYOTO
a52165
jpnkudpc.bitnet

Kentucky, University of
Lexington
Tate T. H. Tsang
KENTUCKY
che159
ukcc.uky.edu

Lakehead University
Thunder Bay, Ontario
David Cooper
LAKEHEAD
dcooper
flash.lakeheadu.ca

Louisville, University of
Louisville
Dermot J. Collins
LOUISVILLE
djcoll01
ulkyvm

Laval, Universite
Ste-Foy, Quebec
Jules Thibault
LAVAL
jules
lavalm1

Louisiana State University
Baton Rouge
Armando B. Corripio
LSU
corripio
lsuche.bitnet

Lehigh University
Bethlehem
William E. Schiesser
LEHIGH
wes1
lehigh

Michigan, University of
Ann Arbor
H. Scott Fogler/Brice Carnahan
MICHIGAN
h_scott_fogler
um.cc.umich.edu

McMaster University
Hamilton, Ontario
Cameron M. Crowe
MCMASTER
crowe
mcmaster.ca

Michigan State University
East Lansing
K. Jayaraman
MICHIGAN_STATE
jayaraman
msuegr.bitnet

Middle East Technical University
Ankara 06531
Turker Gurkan
MIDDLE_EAST_TECHUNIV
a-o-3597
tr.metu

Maribor, University of
YU-62000 Maribor
Peter Glavic
MARIBOR

Maryland, University of
College Park
Tom J. McAvoy
MARYLAND
mcavoy
eng.umd.edu

McGill University
Montreal
M. R. Kamal
MCGILL
kamal
chemeng.lan.mcgill.ca

University of Minnesota, Duluth
Duluth
Dianne Dorland
MINNESOTA_DULUTH
ddorland
ub.d.umn.edu

Maine, University of
Orono
John C. Hassler
MAINE
hassler
maine.bitnet

Massachusetts, University of
Amherst
Philip R. Westmoreland
MASSACHUSETTS
westm
umaecs.bitnet

New Brunswick, University of
Fredericton, New Brunswick
J. J. C. Picot
NEW_BRUNSWICK

Northwestern University
Evanston
Richard S. H. Mah
NORTHWESTERN
dick_may
nuacc.nwu.edu

New Mexico, University of
Albuquerque
H. Eric Nuttall
NEW_MEXICO
nuttall
unmb.unm.edu

North Carolina A&T State University
Greensboro
Franklin G. King
NORTH_CAROLINA_AT

Ottawa, University of
Ottawa, Ontario
David D. McLean
OTTAWA
j24pc
uottawa

Oklahoma, University of
Norman
Lloyd L. Lee
OKLAHOMA
lloydlee
uokmax.ecn.uoknor.edu

Oulu, University of
907570 Oulu
Veikko Pohjola
OULU
po-vp
finou.oulu.fi

Ohio State University
Columbus
Jim Davis
OHIO_STATE
davis
kcgl1.eng.ohio-state.edu

Oregon State University
Corvallis
Keith Levien
OREGON_STATE
levienk
ccmail.orst.edu

Purdue University
West Lafayette
Joseph F. Pekny
PURDUE
pekny
ecn.purdue.edu

Penn State University
University Park
Ali Borhan
PENN_STATE
axb20
psuvm

Princeton University
Princeton
Christodoulos A. Floudas
PRINCETON
floudas
zeus.princeton.edu

Pennsylvania, University of
Philadelphia
Lyle H. Ungar
PENNSYLVANIA
ungar
cis.upenn.edu

Queen's University
Kingston, Ontario
T. J. Harris
QUEENS
harrist
qucdn

Rutgers University
Piscataway
Alkis Constantinides
RUTGERS
constantinid
zodiac.rutgers.edu

Royal Military College of Canada
Kingston, Ontario
Ron D. Weir
ROYAL_MILITARY
weirr
rmc.ca

South Carolina, University of
Columbia
Andrew Farell
SOUTH_CAROLINA
farell
charlie.ece.scarolina.edu

Sherbrooke, Universite de
Sherbrooke, Quebec
Normand Therien
SHERBROOKE
fg00
udesma

South Alabama, University of
Mobile
B. Keith Harrison
SOUTH_ALABAMA
fchar
usouthal

South Florida, University of
Tampa
Aydin K. Sunol
SOUTH_FLORIDA
sunol
sunset.ec.usf.edu

Saskatchawan, University of
Saskatoon, Saskatchawan
G. A. Hill
SASKATCHAWAN
hillg
sask.usask.ca

Toledo, University of
Toledo
Steven E. LeBlanc
TOLEDO
fac0518
uoft01

Toronto, University of
Toronto, Ontario
Will Cluett
TORONTO
cluett
ecf.utoronto.ca

Texas A&M University
College Station
Michael Nikolaou
TEXAS_AM
m0n2431
sigma.tamu.edu

Tennessee Tech
Cookeville
C. P. Kerr
TENN_TECH

Tennessee, University of
Knoxville
Fred E. Weber
TENNESSEE
WEBER
UTKVX1.UTK.EDU

Texas, University of
Austin
James B. Rawlings
TEXAS
jbraw
che.utexas.edu

Vanderbilt University
Nashville
Thomas M. Godbold
VANDERBILT
godboltm
vuctrvax

Virginia, University of
Charlottesville
Peter T. Cummings
VIRGINA
ptc
virginia.edu

Virginia Polytechnic Institute and SU
Blacksburg
Peter R. Rony
VIRGINIA_TECH
rony
vtvm1

Worcester Polytechnic Institute
Worcester
Anthony G. Dixon
WORCESTER_POLY
agdixon
wpi.wpi.edu

Widener University
Chester
John W. Hoopes, Jr.
WIDENER
pfjwhoopes
cyber.widener.edu

Washington University at St. Louis
St. Louis
Rudolphe L. Motard
WASHINGTON_ST_LOUIS
motard
cape1.wustl.edu

Wyoming, University of
Laramie
Michael A. Matthews
WYOMING

West Virginia College of Grad Studies
Institute
F. William Kroesser
WEST_VA_COLLEGE

Washington State University
Pullman
James N. Petersen
WASHINGTON_STATE
scef0003
wsuvm1

West Virginia University
Morgantown
Eugene V. Cilento
WEST_VIRGINIA
cilento
wunvms

Waterloo, University of
Waterloo, Ontario
A. Penlidis
WATERLOO
penlidis
watacs

Youngstown State University
Youngstown
Dilip K. Singh
YOUNGSTOWN_STATE

Microcomputer Chemical Engineering Programs (develped by Professors)

Edited by Bruce A. Finlayson, University of Washington

Have you wondered what microcomputer programs are being used in other chemical engineering curricula? This column provides a mechanism for University professors to let others know about the programs they've developed and are willing to share on some basis.

The program should be described by a 250 word description, machine requirements, and ordering information. These programs should be ready to be shipped out the door, and should have been tested by students at more than one University. It would be helpful if the specific Chemical Engineering course were identified in which the program is useful. The programs will not be reviewed by Professor Finlayson, nor will they be certified by CACHE.

In order to edit the column efficiently, the submissions must be made to Finlayson via BITNET, address FINLAYSON@MAX or on a diskette in ASCII. He will acknowledge receipt of the submission via BITNET and will send the edited column to the CACHE office via BITNET. Letters cannot be accepted.

The column can only be successful if professors submit their writeups. Let us hear from you!

BIODESIGNER

By Demetri Petrides
New Jersey Institute of Technology

BioDesigner is a software tool designed to enhance the effectiveness and productivity of engineers and scientists engaged in design and development of individual or integrated biochemical processes. For a given flowsheet, BioDesigner calculates the material and energy balances, estimates the size and cost of equipment, and carries out a detailed economic evaluation. The program has scheduling capabilies and has the ability to handle batch and semicontinuous processes. A design case in BioDesigner can have any number of unit operations, material streams, and chemical components because memory for those objects is allocated dynamically at runtime. BioDesigner is intended to be used at the early stages of process development and during scale-up. It enables chemical and biochemical engineers to quickly carry out a conceptual design for a new project idea and evaluate it from an economic point of view.

BioDesigner makes use of advanced graphics to facilitate the human/computer interaction and minimize the learning period. It is equipped with an advanced "Help" facility that minimizes the need for manuals.

The current version of BioDesigner runs on any Apple Macintosh computer with at least 1 MB of RAM (more memory is required for large flowsheets). It is written in THINK C and is distributed in object code and parts of source code if desired.

BioDesigner was developed as a part of the Ph.D. thesis of D. Petrides at the Biotechnology Process Engineering Center (BPEC) of the Massachusetts Institute of Technology. Peter Klenk, an undergraduate computer science student, contributed significantly in the development of the interface and a number of other features.

BioDesigner has been used in biochemical engineering courses at MIT during the last two years.

More information may be obtained from:

Prof. Demetri Petrides
Dept. of Chemical Engineering
New Jersey Institute of Technology
Newark, NJ 07102
(201) 596-3614

Internet: dxp6129@tesla.njit.edu

REACT! A CHEMICAL EQUILIBRIUM CALCULATOR

By James A. O'Brien
Yale University

REACT! calculates chemical reaction equilibria by minimizing the Gibbs free energy of the reacting mixture, subject to conservation of material. It requires Microsoft(R) Windows 3.0 and at least 420k of free memory at startup. It is entirely menu driven.

Chemicals and initial mole numbers are chosen from a list, and initial temperature and pressure are selected through dialog boxes. REACT! then solves for the equilibrium mole numbers. In addition, REACT! can calculate and display a set of independent chemical reactions for the mixture of chemicals and an equilibrium constant for each reaction. Options are provided for adiabatic and/or constant volume reaction equilibria, in addition to the more standard isothermal isobaric conditions. The current version assumes immiscible solids and ideal gases. Thermodynamic information is built in for 90 chemicals (33 solids and 57 gases). REACT! is suitable for use in a ChE Thermodynamics or Reactor Design Course.

For more information, contact:

Professor James A. O'Brien
Department of Chemical Engineering
Yale University
New Haven, CT 06520-2159

OBRIEN@YALEVMS (BITNET)
OBRIEN%OBRIEN@VENUS.YCC.YALE.EDU (INTERNET)

The following programs have been listed in prior editions of the CACHE News.

1. Vapor compression refrigeration cycle, No. 24 and 25 Stanley Sandler, University of Delaware
2. Compression of an ideal gas, No. 24 and 25, Stanley Sandler, University of Delaware
3. Computer Aided Analysis for Process Systems, No.24 and 25, Ted Cadman, University of Maryland
4. Discounted Cash Flow Analysis (and Present Worth), No. 24 and 25, Bruce A. Finlayson, University of Washington
5. Short-cut Distillation and Flash Calculations, No. 24 and 25, Bruce A. Finlayson, University of Washington
6. Convective Diffusion Equation (CDEQN), No. 25 and 26, Bruce A. Finlayson, University of Washington
7. Engineering Plot (ENGNPLOT), No. 25 and 26, Bruce A. Finlayson, University of Washington
8. Educational Software for Teaching Process Dynamics and Control, No. 26 and 27, Patrick Richard and Jules Thibault, Laval University
9. MIDAS - Microcomputer Integrated Distillation Sequences, No. 26 and 27, Andrew Hrymak, McMaster University
10. A Rigorous Multicomponent Multistage Steady-State Distillation Rating Program, No. 27 and 28, E.C. Roche, Jr., New Jersey Institute of Technology
11. RESIM. A Reactor Design Teaching Tool, No. 27 and 28, B.W. Wojciechowski, Queen's University
12. Real-time Multiloop Computer Control Program, UC ONLINE , No. 27 and 28, by Alan Foss, University of California at Berkeley
13. Real-time Dynamic Distillation Simulation and Relative Gain Program , No. 27 and 28, by Alan Foss , University of California, Berkeley
14. The Kinetics and Selectivity of Consecutive Reactions, No. 29 and 30, by Alvin H. Weiss and Reynold Dodson, Worcester Polytechnic Institute.
15. Equations of State, No. 30 and 31, by Kenneth R. Jolls, Iowa State University.
16. Thermal Design of Shell and Tube Heat Exchangers, No. 31 and 32, by Nurcan Bac and Ilker Ozal, Worcester Polytechnic.
17. Optimum Series Bioreactor Design, No. 31 and 32, by Gordon Hill, University of Saskatchewan.

Announcements

Cray Research Sponsors University Licenses for SPEEDUP

By Dr. Stephen E. Zitney, Cray Research, Inc.

SPEEDUP is a software system used to solve problems in chemical process engineering. Over 100 major process industry companies and 30 universities around the world are using this software system. Based on original research work conducted at London's Imperial College, SPEEDUP can perform the widest spectrum of simulations:

- dynamic simulation, design and optimization
- steady-state simulation, design and optimization
- parameter estimation
- data reconciliation

Cray Research, Inc. and Prosys Technology Ltd. (ProsysTech), of Cambridge, England, are jointly developing a high-performance version of ProsysTech's SPEEDUP software for use on Cray Research supercomputers. Product availability is scheduled for the first quarter of 1992.

As part of the agreement with ProsysTech, Cray Research is sponsoring a limited number of five-year, royalty free educational licenses for the Cray Research version of SPEEDUP. (The standard fee for a five-year academic license is $5000.) Product updates are also being handled at a lower cost for these Cray Research sponsored licenses. ProsysTech will provide the updates for a nominal fee intended to cover the cost of distribution of such updates.

With this approach, Cray Research and ProsysTech will assist engineering students in gaining experience with dynamic simulation methods for solving chemical engineering problems. In addition, we believe that allowing students to work with high-performance computers firsthand will not only improve their productivity, but also capture their attention and ultimately motivate them to pursue the use of supercomputing in their future scientific and engineering endeavors.

To apply for a SPEEDUP academic license on Cray Researcsystems, a proposal must be received by November 1, 1991. Notification of acceptance will be given by January 1, 1991. Your proposal must include the following:

1. Sponsor(s) name, affiliation, mailing address, email address, telephone number, and fax number.

2. Describe sponsor's current position.

3. Describe how the Cray Research version of SPEEDUP will be used in education and research. If SPEEDUP is currently used for research, please attach a recent research article or abstract. Any use of SPEEDUP for commercial purposes shall not be permitted.

4. Indicate the site where the Cray Research version of SPEEDUP will be used; include a description of the CRAY supercomputer (system type: CRAY Y-MP,CRAY X-MP, CRAY-2, number of processors, and memory). Universities can be authorized to install and use the SPEEDUP code on a Cray Research system at a national supercomputer center. Each license sponsored by Cray Research permits SPEEDUP to be used only by a single university affiliated with such a center. Accordingly, each university accessing SPEEDUP on a Cray Research system at a supercomputer center must have an academic license for the Cray Research version of SPEEDUP.

5. Send proposals to Stephen E. Zitney, Cray Research, Inc., 655-E, Lone Oak Drive, Eagan, MN 55121. Phone (612-683-3690). E-mail (sez@cray.com).

EURECHA Teaching Program Project

The EURECHA Teaching Program Project collects and distributes chemical engineering programs to teaching institutions. The modifications make the programs suitable for IBM-PC machines and can be used interactively.

Programs Available

1. CHEMCOSET: data bank
2. UNICORN: flowsheet programs
3. DISTILSET: distillation programs
4. SOPS: state oriented property system
5. CAPCOS: capital cost estimation package
6. TACS: simulation of control systems
7. VLESET: comprehensive VLE analysis package

8.	SYNSET:	heat exchanger network synthesis and column sequencing
9.	BATCHDIST:	batch distillation design program
10.	INTERN:	the designs of column internals
11.	REACTEX:	11 reactor models suitable for running alone or as UNICORN subroutines. All have been used as the basis for student design projects.
12.	ENISYN:	Synthesis of energy integrated distillation systems
13.	OPTIMISER:	Non-linear optimization in chemical engineering
14.	REPROCHE:	Non-linear regression program
15.	CHEQUUS:	Chemical equilibrium calculation
16.	KINET:	A kinetic examination, data acquisition and regression program
17.	SENSKIN:	Sensitivity analysis of chemical source terms in reactors
18.	UCTNET:	Analysis and design of heat exchanger networks

The normal price for one source code is 300 Swiss francs. As a special offer, the UCTNET program and the new version of ENISYN program, are available for EURECHA members for 100 Swiss francs.

The next EURECHA Annual General Meeting will be during the COPE-91 "European Symposium on Computer Applications in Chemical Engineering" in Barcelona, Spain, 13-16 October 1991.

For further information, contact:

Dr. Z. FONYO
ETH-Zentrum
Technisch-Chemisches Laboratorium
CH-8092 Zurich, Switzerland
Telefax: +41 1 252 0975
Electronic Mail Address: fonyo@ezzus.vmsmail.ethz.CH

Conceptual Design Program PIP-CPS

From University of Massachusetts at Amherst and Simulation Sciences

An updated version of PIP, the Process Invention Procedure from UMass is now available on request from CACHE. The new version is the result of a joint effort by Professor James Douglas at the University of Massachusetts at Amherst and Simulation Sciences, Inc. in Fullerton, California. It is now called PIP-CPS to reflect the contributions from the CPS, Chemical Process Synthesis, a project at Simulation Sciences.

The PIP-CPS program is an aid for conceptual design and an excellent tool for teaching conceptual design. It closely follows the concepts and structures of the textbook "Conceptual of Chemical Processes Design" by Professor James Douglas, McGraw Hill, c. 1988.

The new version of PIP-CPS is significantly different from the version of PIP that was distributed by CACHE in 1988. The current version is much more robust, and includes functionality for complex flowsheets, solid components, and steady=state control analysis. It comes with data files for five flowsheets:

HDA1	Hydrodealkylation of toluene
POTASH	Solids separation of KCl from a KCl & NaCl mixture
STYREN	Styrene from toluene
TPN	Terephthalonitrile to BHET
XYLENE	Xylene from toluene

The computer requirements are as follows:

VAX VMS Version 5
GKS run time graphics package
Approximately 15000 blocks of disk space

DEC GKS for VMS, Version 4.1 (Product SSA 26.20.09) must be licensed from Digital Equipment.

Announcements (continued)

New CACHE Design Case Study

By Ignacio Grossmann, Carnegie Mellon,

VOLUME 6 - "Chemical Engineering Optimization Models with GAMS"

This case study has been prepared by Professors Larry Biegler (Carnegie Mellon), Chris Floudas (Princeton), Iftekhar Karimi (Northwestern), and Ignacio E. Grossmann (Carnegie Mellon) with their research students. GAMS Development Corporation, the Licensing Technology Office at Stanford University, XMP Optimization Software and Sunset Systems have donated the computer software for this case study.

This case study describes the formulation and solution of 22 chemical engineering optimization problems with the modelling system GAMS. While in the past optimization courses have been largely confined to small analytical problems, GAMS offers the possibility of solving a variety of meaningful optimization problems where the students can concentrate on problem formation. The objective of this case study is to provide a set of chemical engineering problems to supplememt optimization courses at both the undergraduate and graduate level. This case study should also be useful in other courses to learn about GAMS and its applications.

This case study covers applications at various levels of complexity in the following areas: (a) Planning and scheduling of batch and continuous processes, (b) Chemical and phase equilibrium, (c) Design of heat exchanger networks, distillation columns and batch processes, (d) Synthesis of reaction paths, heat exchanger networks and distillation sequences, (e) Optimization of dynamic and distributed parameter models. These problems are modelled as linear, nonlinear and mixed-integer optimization problems.

This case study describes in detail the formulation and solution of a total of 22 optimization problems that cover the different areas cited above. Exercises are also given for each problem. In addition, the case study includes:

a) Special student version of GAMS for IBM-PC and compatibles in 3 1/2" diskettes. The MINOS, ZOOM and DICOPT++ codes are included in this GAMS version which can handle problems with up to 1000 nonzero elements in the Jacobian matrix.
b) GAMS input files for all the problems; these are extensively documented.
c) GAMS User's Guide.

Status of Flowtran Load Modules for University Computers

By J. D. Seader, University of Utah

As part of a continuing program of support to education, Monsanto Company announced on August 19, 1982, that load modules for the FLOWTRAN simulation program would be made available on magnetic tape to chemical engineering departments to install on their in-house computers. Thus departments would be able to run FLOWTRAN at no additional charge.

CACHE continues to supervise the preparation of FLOWTRAN load modules for some mainframe, supermini, and supermicro type digital computers and the distribution of the modules on magnetic tape to departments that order them. A new optimization feature is now included, and the instructional FLOWTRAN is in its third edition. Please see the order form at the end of this newsletter.

FLOWTRAN tapes are now available for the following computers:

Apollo workstations running AEGIS operating system (program on floppy disks).

CDC Cyber mainframe computers with the NOS operating system and a FORTRAN V compiler.

DEC VAX computers running with either the VMS or ULTRIX operating system.

IBM and IBM-plug compatible mainframe computers such as the 370, 30XX, and 43XX with the following operating system and FORTRAN compiler combinations:

Version	Operating System	FORTRAN Compiler
a	VM/CMS	VS
b	OS/VS2 MVS	VS

Sun workstations running UNIX

Encore Multimax APC computer.

Announcements (continued)

Each FLOWTRAN tape contains either load and/or relocatable code, test problems and solutions, and installation instructions. The FLOWTRAN program may be used for educational purposes, but not for consulting. A total of 176 FLOWTRAN tapes, cartridges, and floppy disks have already been distributed.

If you would like to obtain a FLOWTRAN tape for your computer and have not already contacted CACHE, complete and submit the FLOWTRAN TAPE form in the back of this issue of CACHE News. You will be required to sign a User's Agreement that must be approved by Monsanto. The cost of the tape, payable to CACHE, is $250. The charge to CACHE-supporting departments is $175.

FLOWTRAN TAPE ORDER FORM

_____ I am interested in preparing a FLOWTRAN tape.

_____ I am interested in obtaining a FLOWTRAN tape.

If you have checked either of the above, please complete the following information. If you have two computers you want to consider, duplicate this form and submit both completed forms and your preference.

1. Computer make and complete model number: _____
2. Operating system version: _____
3. FORTRAN compiler version: _____
4. Magnetic tape facility: _____
5. No. of tracks: _____
6. Drive speed in bits/inch: _____

Send form to:
Professor J. D. Seader/CACHE
3290 MEB
University of Utah
Salt Lake City, UT 84112

FLOWTRAN BOOKS ORDER FORM

Please send me:
Flowtran Simulation - An Introduction, 3rd edition; Pauls.

No. of copies: _____ at $16.95/copy

Name: _____
Address: _____
_____
_____
_____

Send form to: ULRICH'S Bookstore
ATTN: Heather Senior
549 E. University Avenue
Ann Arbor, MI 48109

Add $1.50 per copy for mailing and handling in the U.S.
A check must accompany all orders by individuals. Also available in quantity at regular quantity discounts to established book retailers. Make checks payable to Ulrich's Bookstore.

ORDER FORM

CACHE Process Design Case Study Vol. 4
Alternative Fermentation Processes for Ethanol Production

The objective of this case study is the preliminary technical and economic evaluation of a fermentation process for the production of ethanol from a molasses feedstock. The intent is to expose the student to some non-traditional chemical engineering processes and to the rapidly expanding field of biotechnology. Groups of 2-3 students should be able to complete the design in about 30 days. The major focus of this design study is the creation and rational development of a suitable process flowsheet, simulation of the flowsheet by a commercial process simulator (in this case FLOWTRAN), and economic evaluation and cost minimization of the final process.

The problem begins with the specification of the plant operating requirements. The type of fermentor to be used as well as plant operating conditions are left open. Suggested fermentors include batch, CSTR, CSTR with cell recycle as well as a novel extractive fermentor based on the use of hollow fiber membranes, the Hollow Fiber Extractive Fermentor (HFEF). The choice of fermentor will affect the nature of the flowsheet and lead to several design alternatives. Given the time constraints, the student will have to rationally screen these alternatives before arriving at a workable flowsheet ready for simulation. A kinetic expression describing the production of ethanol from glucose is provided as well as a 5 1/4" floppy disk (IBM format) with a BASIC program for evaluating the performance of the CSTR fermentors. Performance characteristics are also provided for the batch fermentor and the HFEF. Detailed explanations and graphics are included to explain the results and the FLOWTRAN program for the suggested design.

The problem statement was posed by Professors Steven E. LeBlanc and Ronald L. Fournier and prepared under their supervision, by Mr. Samer F. Naser, all of the University of Toledo.

CACHE PROCESS DESIGN CASE STUDY VOLUME 4
"Alternative Fermentation Processes for Ethanol Production"

# of copies:

_____ $15 for initial copy for CACHE-supporting department ..... _____

_____ $35 for additional copies ..... _____

_____ $35 for non-supporting departments ..... _____

TOTAL _____

Name: _____

Address: _____

_____

_____

Please mail order form to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

Make check payable to CACHE Corporation.

ORDER FORM

CACHE Process Design Case Study Vol. 5
Retrofit of a Heat Exchanger Network and Design of a Multiproduct Batch Plant

This volume contains two short design projects that can be developed by groups of 2-3 students in about two weeks. As opposed to the large projects that are commonly used in a design course, the objective of the case study is to expose students to a greater variety of problems and which are of current industrial significance.

The first problem deals with the retrofit of a heat exchanger network consisting of 8 exchangers with 5 hot and 3 cold processing streams as well as steam and cooling water. The layout of the network and areas of the exchangers are also given. The objective is to determine a retrofit design that can reduce the energy consumption within specified limits for the capital investment and payout times. This problem requires examination of alternatives for the level of energy recovery, matching of streams, addition of area, and removal or reassignment of existing exchangers and piping. This problem can be used to illustrate basic concepts of heat integration, as well as the application of computer software such as Target II, THEN, MAGNETS and RESHEX. The second design problem deals with the design of a batch processing plant that has to manufacture 4 different products, all of which require 5 similar processing steps (reaction, product recovery, purification, crystallization and centrifuge). An important aspect of this problem is that the production schedule and inventory must be anticipated at the design stage. Furthermore, this problem also requires analyzing alternatives for merging processing tasks into single units, and using parallel units with and without intermediate storage. The use of Gantt charts is emphasized to examine some of these alternatives. The case study also includes two sets of homework problems with solutions that can be used to provide the basic background for the two problems.

This case study has been prepared by the students Richard Koehler and Brenda Raich at Carnegie Mellon University under the supervision of Professor Grossmann who developed the problem statements and educational material.

CACHE PROCESS DESIGN CASE STUDY VOLUME 5
"Retrofit of a Heat Exchanger Network and Design of a Multiproduct Batch Plant"

# of copies:		Total
_____	$20 for initial copy for CACHE-supporting department	_____
_____	$40 for additional copies	_____
_____	$40 for non-supporting departments	_____
	TOTAL	_____

Name: _____

Address: _____

_____

_____

Please mail order form to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

Make check payable to CACHE Corporation.

ORDER FORM

CACHE Design Case Study Volume 6
Chemical Engineering Optimization Models with GAMS

The objective of this case study is to provide a set of chemical engineering problems to supplememt optimization courses at both the undergraduate and graduate level. This case study should also be useful in other courses to learn about GAMS and its applications.

GAMS is an algebraic modelling system in which the user need not be concerned with details of providing the interfaces with various optimization codes. Instead, the GAMS environment allows the user to *concentrate on the modelling of problems*, which ultimately is the main skill that is required for the successful application of optimization in practice. This case study covers applications at various levels of complexity in the following areas: (a) Planning and scheduling of batch and continuous processes, (b) Chemical and phase equilibrium, (c) Design of heat exchanger networks, distillation columns and batch processes, (d) Synthesis of reaction paths, heat exchanger networks and distillation sequences, (e) Optimization of dynamic and distributed parameter models. These problems are modelled as linear, nonlinear and mixed-integer optimization problems.

This case study describes in detail the formulation and solution of a total of 22 optimization problems that cover the different areas cited above. Exercises are also given for each problem. In addition, the case study includes:

a) Special student version of GAMS for IBM-PC and compatibles in 3 1/2" diskettes. The MINOS, ZOOM and DICOPT++ codes are included in this GAMS version which can handle problems with up to 1000 nonzero elements in the Jacobian matrix.
b) GAMS input files for all the problems; these are extensively documented.
c) GAMS User's Guide.

This case study has been prepared by faculty and students from Carnegie Mellon University, Northwestern University and Princeton University under the coordination of Ignacio E.. Grossmann. GAMS Development Corporation, the Licensing Technology Office at Stanford University, XMP Optimization Software and Sunset Systems have donated the computer software for this case study.

CACHE PROCESS DESIGN CASE STUDY VOLUME 6
"Chemical Engineering Optimization Models with GAMSt"

# of copies:		Total
_____	$55 per copy for CACHE-supporting departments	_____
_____	$80 for non-supporting departments	_____
	TOTAL	_____

Name: _____

Address: _____

_____

_____

Make check payable to CACHE Corporation.

Please mail order form to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

POLYMATH Order Form

If you decide to obtain POLYMATH for testing (see form below), please be aware of the following:

1. **You may reproduce the program as many times as you like for students and other faculty.**
2. **Your department chairman will be informed of the testing.**
3. **If you decide to use POLYMATH in your department after 3 months, your department will be billed for $125.00, and $75.00 for each successive year thereafter. This fee covers any updates or new versions.**
4. **If you decide not to use POLYMATH after 3 months, you must return (or certify you have erased) all copies made.**
6. **Educational supporting material will be available from CACHE later in the year at $50.00 per copy.**

Please send me a copy of POLYMATH for the IBM/PC. I have read and understood the conditions described above.

Date: _____

Requested Disk:

[ ] 5 1/4" Double Density

[ ] 5 1/4" High Density

[ ] 3 1/2" High Density

Name _____

Address _____

_____

_____

Machine _____

Make check payable to CACHE Corporation and send to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

A SERIES OF MONOGRAPHS ON AI IN CHEMICAL ENGINEERING

A series of monographs have been and are being written for use as main or supplementary material in advanced undergraduate and graduate courses addressing the application of expert systems or as a working introduction to AI by practicing engineers. Three monographs in the series are available. The purpose of these first three monographs is to provide detailed discussions on the principles, ideas, techniques, methodologies and issues of AI as they apply to chemical engineering. Later monographs will address approaches to specific problems of direct interest to chemical engineers such as fault diagnosis, design, etc. Currently available are:

Volume I, "Knowledge-Based Systems in Process Engineering: An Overview", is authored by George Stephanopoulos of MIT. This volume serves as an introduction to the monograph series and provides a broad perspective on AI. Specifically, this volume addresses the scope, history and market of AI and defines the need and role of knowledge-based systems in chemical engineering. Particular attention is paid to describing the general issues surrounding software and hardware environments.

Volume II, "Rule-Based Expert Systems in Chemical Engineering", is authored by James F. Davis and Murthy S. Gandikota of Ohio State University. This monograph focuses specifically on the implementation of knowledge-based systems in rule-based languages. The emphasis is not on the mechanics of rule-based programming environments, but on the issues which impact the implementation and performance of a system. While the focus is on rule-based implementations, many of the issues discussed cut across all general purpose implementation language. Using specific examples, the monograph covers these issues in detail. As a stand alone chapter, several of the most popular methods for various kinds of uncertainty handling are discussed and compared.

Volume III, "Knowledge Representation", is authored by Lyle Ungar of the University of Pennsylvania and V. Venkatasubramanian of Purdue University. The content of this monograph is directed at two distinct aspects of knowledge representation. In the first part of the monograph, the problem-independent issues and features of a variety of knowledge representations are presented. Included are discussions on semantic networks, frames, scripts and object-oriented programming. The second part addresses the subject of qualitative physics applied in chemical engineering. The issues of representing structure and behavior are discussed in detail. Examples demonstrating two philosophies are used to illustrate advantages and limitations.

The price for non-supporting departments and for extra copies to supporting departments will be $15 each or $35 for set of 3. To order, please complete the Standard Order Form on the following page.

STANDARD ORDER FORM

Description of Item	Quantity	Unit Price		Total
		Supporting	Non-supporting	
AI Monograph - Volume 1		$15		
AI Monograph - Volume 2		$15		
AI Monograph - Volume 3		$15		
Set (1, 2, and 3)		$35		
AI Case Study - Volume 1		$10	$17	
AI Case Study - Volume 2		$10	$17	
AI Case Study - Volume 3		$10	$17	
Set (1, 2, and 3)		$20	$35	
GPSS		$15		
PIP		$50	$75	

Name: _____

Address: _____

_____

_____

Please make checks payable to CACHE Corporation
in U.S. funds and drawn on a U. S. Bank and return to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

For a complete listing of CACHE products, send in for a CACHE catalog.

STANDARD ORDER FORM

Description of Item	Quantity	Unit Price		Total
		Supporting	Non-supporting	

Name: _____

Address: _____

_____

_____

Please make checks payable to CACHE Corporation
in U.S. funds and drawn on a U. S. Bank and return to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

For a complete listing of CACHE products, send in for a CACHE catalog.

STANDARD ORDER FORM

Description of Item	Quantity	Unit Price		Total
		Supporting	Non-supporting	
AI Monograph - Volume 1		$15		
AI Monograph - Volume 2		$15		
AI Monograph - Volume 3		$15		
Set (1, 2, and 3)		$35		
AI Case Study - Volume 1		$10	$17	
AI Case Study - Volume 2		$10	$17	
AI Case Study - Volume 3		$10	$17	
Set (1, 2, and 3)		$20	$35	
GPSS		$15		
PIP		$50	$75	

Name: _____

Address: _____

_____

_____

Please make checks payable to CACHE Corporation
in U.S. funds and drawn on a U. S. Bank and return to:

CACHE Corporation
P.O. Box 7939
Austin, TX 78713-7939

For a complete listing of CACHE products, send in for a CACHE catalog.

INDUSTRIAL CONTRIBUTORS TO CACHE

The following companies have recently contributed financial support to specific CACHE activities:

Chevron Research and Technology Company

DuPont Committe on Educational Aid

Monsanto Chemical Company

Rust International Corporation

Shell Companies Foundation

Simulation Sciences

Tektronix

Xerox Foundation

List of Chemical Engineering Departments Supporting CACHE

CACHE annually solicits universities for funds to carry out on-going CACHE activities and nurture new projects. The following is a list of our generous supporters:

Auburn University
Tuskegee University
University of Alabama, Tuscaloosa
University of South Alabama
Arizona State University
University of Arizona
University of Arkansas
University of California, Berkeley
University of California, Davis
University of California, Los Angeles
University of California, San Diego
University of California, Santa Barbara
California Institute of Technology
California State Polytechnic University, Pomona
University of Southern California
University of Colorado
Colorado School of Mines
Colorado State University
University of Connecticut
Yale University
University of Delaware
Howard University
Florida A&M University
Florida Institute of Technology
University of South Florida
Georgia Institute of Technology
University of Idaho
University of Illinois, Chicago
University of Illinois, Urbana
Illinois Institute of Technology
Northwestern University
Purdue University
University of Notre Dame
Rose-Hulman Institute of Technology
Tri-State University
University of Iowa
Iowa State University of Sci. & Tech.
University of Kansas
Kansas State University
University of Kentucky
University of Louisville
Louisiana Technical University
Louisiana State University
University of Southwestern Louisiana
University of Maryland
University of Lowell
University of Maine
Johns Hopkins University
Massachusetts Institute of Technology
University of Lowell
University of Massachusetts, Amherst
Northeastern University
Tufts University
Worcester Polytechnic Institute
University of Michigan
Michigan State University
Michigan Technological University
Wayne State University
University of Minnesota, Duluth
University of Minnesota, Minneapolis
Mississippi State University
University of Missouri, Columbia
University of Missouri, Rolla
Washington University
University of Nebraska, Lincoln
University of Nevada, Reno
University of New Hampshire
Dartmouth College
New Jersey Institute of Technology
Princeton University
Rutgers, The State University
Stevens Institute of Technology
University of New Mexico
Clarkson University
Cornell University
Manhattan College
Polytechnic University
Rensselaer Polytechnic Institute
University of Rochester
State U. of New York at Buffalo
Syracuse University
N. Carolina State University, Raleigh
N. Carolina A&T State University
University of North Dakota
University of Akron
Case Western Reserve University
University of Cincinnati
Cleveland State University
University of Dayton
Ohio State University
University of Toledo
Youngstown State University
University of Oklahoma
Oklahoma State University
Oregon State University
Bucknell University
Carnegie Mellon University
Drexel University
Lafayette College
Lehigh University
University of Pennsylvania
Pennsylvania State University
University of Pittsburgh
Widener University
University of Rhode Island
Clemenson University
University of South Carolina
S. Dakota School of Mines & Tech.
University of Tennessee, Knoxville
Vanderbilt University
Lamar University
University of Texas at Austin
University of Houston
Rice University
Texas A&M University
Texas Tech University
University of Utah
Brigham Young University
Hampton University
University of Virginia
Washington State University
University of Washington
University of West Virginia College of Graduate Studies
W. Virginia Institite of Technology
W. Virginia University
University of Wisconsin
University of Wyoming
University of Adelaide, South Australia
University of Sidney, Australia
University of Alberta
University of British Columbia
Technical University of Nova Scotia
University of Ottawa
Ecole Polytechnique—U. of Montreal
Queen's University
Royal Military College of Canada
University of Calgary
Lakehead University
Laval University
McGill University
McMaster University
University of New Brunswick
University of Saskatchewan
University of Sherbrooke
University of Toronto
University of Waterloo
University of Concepcion, Chile
University of Santiago de Chile
Institut for Kemiteknik, Denmark
University of Bath, England
University of Oulu, Finland
University of Stuttgart, Germany
Imperial College, Great Britain
South Bank Polytechnic, G. Britain
Indian Institute of Technology
University College, Dublin, Ireland
Kyoto University, Japan
KAIST, Korea
Pohang Institute of Science & Technolgy, Korea
Institute Tech. de Celaya, Mexico
Norwegian Institute of Technology
King Abdulaziz U., Saudi Arabia
Escuela de Ingenieros, Spain
University of Barcelona, Spain
University Politecnica Catalunya, Spain
Institute D'Automatique, EPFL, Switzerland
Middle East Tech. University, Turkey
University of Maribor, Yugoslavia

Topics in this issue of the CACHE Newsletter:

CACHE Corporation
P. O. Box 7939
Austin, TX 78713-7939

Nonprofit Organization
U.S. Postage Paid
Austin, TX
Permit No. 699

Chairman Thomas F. Edgar
Department of Chemical Engineering
University of Texas
Austin TX 78712-1062